SRA: Spatial Reasoning Adapter via Evolving Social Interaction Graphs for Trajectory Prediction

Jaewoo Jeong[1*] **Seonkyu Song**[1*] **Hyeonwoo Park**[2] **Jegyeong Cho**[1]
Yujin Bae[1] **Giwon Lee**[1] **Daehee Park**[2,3] **Kuk-Jin Yoon**[1]
[1]KAIST [2]DGIST [3]UNIVA
{jeong207, sok9854, j2k0618, yujinbae, dlrldnjs, kjyoon}@kaist.ac.kr
{hyeonu4995, dhpark}@dgist.ac.kr

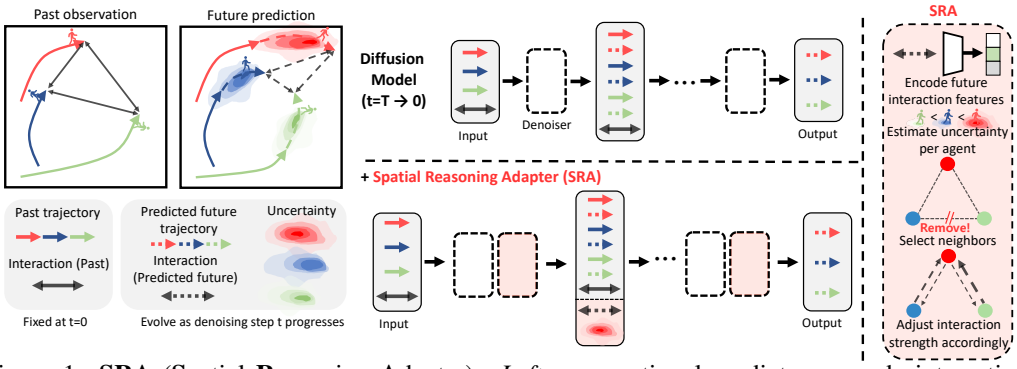

Figure 1: **SRA** (**S**patial **R**easoning **A**dapter). *Left:* conventional predictors encode interactions (black arrows) once from the past trajectories and keep them fixed during denoising, whereas SRA infers interactions among the *predicted* futures, together with a per-agent uncertainty, and lets them evolve as denoising progresses. *Middle:* a host diffusion model (top) and the same model with SRA inserted at every denoising step (bottom, red blocks). *Right:* at each step, SRA encodes future interaction features, estimates per-agent uncertainty, selects neighbors (removing irrelevant edges), and adjusts interaction strength by uncertainty, refining spatial relations for coherent scene-level forecasts.

Abstract

Predicting where multiple agents will move next is fundamental to autonomous driving, robotics, and any system that must share space with people safely. Modern stochastic predictors read off inter-agent interactions once from the observed past and keep that view fixed throughout the generative rollout, so the reciprocal geometric constraints that emerge as the joint future sharpens are not re-examined. We address this gap with **SRA** (**S**patial **R**easoning **A**dapter), a modular graph denoiser that is integrated into existing diffusion and flow-matching predictors without redesigning their generative cores. At every reverse step, SRA reads the host's current future estimate, builds a sparse Euclidean interaction graph on the predicted trajectories, and returns a residual correction to the host's decoding path. Our adapter consists of three core components: a sparse neighborhood retrieved with a learned query-key score and a geometric bias, a future-conditioned relational feature recomputed on the evolving prediction, and uncertainty-weighted message passing where confident agents act as anchors for uncertain ones. We instantiate SRA on three structurally different hosts (LED, MID, and MoFlow) across four multi-agent benchmarks: SDD, NBA, Soccer, and Football. SRA improves ADE for every host on every dataset, with average gains of 5.8% in ADE

*Equal contribution.

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

and 7.0% in FDE. A cumulative ablation shows that the three components act as one mechanism: only together do they improve every host on every sports dataset, indicating that spatial reasoning over the model's own evolving future, weighted by how far that future can be trusted, is broadly useful across denoising architectures. Code is available at `https://github.com/Jaewoo97/SRA`.

1 Introduction

Trajectory prediction aims to forecast the future motions of dynamic agents in a scene [1, 68, 9, 42, 17, 11, 29, 55, 43, 5]. As a core component of autonomous driving [35, 39, 47, 33, 16, 74, 46, 44, 45], robotics [51], and surveillance systems [31], it underpins safe motion planning in dynamic environments [26, 27]. Recent studies cast trajectory prediction as a generative problem rather than a deterministic regression task, focusing on uncertainty [57, 37, 56, 53] and multi-modality in agent behaviors [30, 22, 54]. Within this view, *stochastic denoisers* have emerged as the dominant recipe for multi-agent forecasting, spanning diffusion predictors such as MID [15] and its leapfrog-initialized variant LED [37], as well as flow-matching predictors such as MoFlow [14].

Despite their generative strength, these models share a structural limitation: interactions among agents are encoded *once*, from observed histories, and held fixed throughout the rollout [76, 64, 3]. However, the interactions that decide whether a joint future is plausible are interactions between *future* trajectories, such as whether two predicted paths cross or which attacker a defender ends up marking. These relations do not exist in the observed past: they appear only once the futures are predicted, and they differ from one sampled future to another. A context computed once from the past cannot represent them, so each agent's future is refined without being checked against the futures of the others. While some works inject future-aware interactions via cross-operations in a latent manifold [20, 14, 23], such implicit coupling lacks spatial precision and offers limited interpretability for enforcing geometric relations (e.g., gaps, right-of-way, group motion) [36]. Rollouts therefore remain individually plausible yet globally inconsistent, exhibiting near-collisions or incoherent group behaviors that degrade prediction accuracy and downstream safety [19, 49, 10].

We argue that what limits these models is not how well they generate each trajectory, but how well they model the interactions among the future trajectories they generate. To this end, we introduce **S**patial **R**easoning **A**dapter (SRA), a *modular graph denoiser* that models these future interactions and is integrated into an existing stochastic denoiser without redesigning its diffusion or flow-matching core. At each denoising step, SRA takes the host's own current future estimate $\hat{\mathbf{y}}$, builds a sparse agent-pair graph directly in Euclidean space, and emits a residual correction that the host injects back into its denoising update (Fig. 1). Because the graph is recomputed on each sample's *evolving* estimate, the K modes receive K different interaction structures that sharpen as the prediction does: a defender reacts to an attacker's projected run, not to the last two seconds of history. SRA is modular but not training-free: it is trained jointly with, or fine-tuned on, the host.

SRA is built around three components. First, *sparse neighborhood selection*: for each agent we select K_{nbr} neighbors via a retrieval-style score that combines a learned semantic term with a lightweight geometric bias, so that all subsequent encoding and message passing run only over relevant neighbors. Second, *agent-pair relational future encoding*: for each selected pair we build a pair-specific edge feature directly on the host's current future estimate $\hat{\mathbf{y}}^0$. Third, *uncertainty-weighted message passing*: a predicted per-agent, per-timestep σ, calibrated via an NLL regularizer [40, 4], modulates routing so that confident agents act as anchors for uncertain ones. We instantiate SRA on three hosts that span modern stochastic predictors: the leapfrog-initialized DDPM LED [37], the vanilla DDPM MID [15], and the flow-matching model MoFlow [14]. We evaluate on NBA [72], Soccer [52], Football [38], and SDD [48]. SRA improves ADE for every host on every dataset, with average gains of 5.8% in ADE and 7.0% in FDE.

Our contributions are threefold:

- We propose **SRA**, a modular graph denoiser for iterative stochastic trajectory predictors that rebuilds an explicit interaction graph on the host's current future estimate at every reverse step and injects a residual correction without redesigning the generative core.
- We design three components (sparse neighborhood selection, agent-pair relational future encoding, and uncertainty-weighted message passing) and show with a cumulative abla-

tion on three sports benchmarks that they act as one mechanism: only their combination improves all nine host–dataset pairs.

- We instantiate SRA on three structurally different hosts (LED, MID, MoFlow) and show consistent ADE/FDE improvements on four benchmarks.

2 Related Work

2.1 Trajectory prediction

Trajectory prediction underpins autonomous driving [77], mobile robotics [25], and human-robot interaction [17, 2]. Multimodal human behavior has shifted the field from deterministic regression [55, 11, 42] to generative formulations: GANs and VAEs [17, 28, 29], then diffusion- and flow-based models [57, 37, 30, 15, 14, 34, 70]. A complementary line models inter-agent interaction via pooling [1, 17], attention [76, 41, 66, 58], equivariant [69] and graph/hypergraph reasoning [67, 51, 59, 61, 12, 21], higher-order modules [7, 25], and uncertainty-aware diffusion [4].

A structural limitation persists across both threads: the inter-agent signal is almost always derived from *observed histories* and held fixed as the predictor unrolls [76, 64, 3]. SocialMOIF [7] distils a future-conditioned posterior into a history-only one at training time, but at inference still reduces to history cues. Scene-aware stochastic denoisers [24, 3, 4, 70, 34] inject agent context during denoising through cross-attention over latent features rather than explicit geometric relations on the predicted futures, leaving no handle for spatial constraints such as gaps, right-of-way, or group cohesion. Methods that do condition on predicted futures do so in latent space: GameFormer [20] attends to the latent futures of other agents through level-k reasoning, MoFlow [14] mixes agents with latent attention in its decoder, and the coarse-to-fine approach [23] refines along the temporal axis; none builds an explicit graph on the predicted coordinates or weighs a relation by the reliability of its endpoints (compared in Supp. E). To our knowledge, no prior method explicitly reasons over *future-future* agent interactions in Euclidean space *at inference* for scene coherence.

We close this gap with a modular graph denoising adapter on diffusion- and flow-based predictors, whose iterative denoising chain naturally supports recomputing future-trajectory interactions on the model's own evolving estimate. At every reverse step the adapter builds an explicit Euclidean interaction graph over the current future estimate and runs uncertainty-weighted message passing, injecting scene-level coherence as a residual correction without altering the host's generative core.

2.2 Spatial reasoning with diffusion models

A recent line of work treats the denoising chain as a vehicle for *structured reasoning* rather than plain sample refinement [63]. One thread induces causal or temporal structure into the noise schedule itself: Diffusion Forcing [6], AR-Diffusion [65], Rolling Diffusion [50], and TEDi [73] use position- or time-dependent schedules to enforce sequential dependencies, and SRM [63] frames denoising as ordered structured reasoning over continuous variables.

A parallel thread injects explicit *spatial* structure into diffusion: LayoutDiffusion [75], SceneGenie [13], R3CD [32], and CommonScenes [71] guide image and 3D-scene diffusion with layouts, scene-graph priors, or relational constraints. For relation prediction itself, DIFFVSGG [8] casts video scene-graph generation as iterative denoising over object-pair embeddings, SPADE [18] folds spatial relation reasoning into a panoptic scene-graph denoiser, and Wang et al. [60] analyze how reasoning structures emerge across the video-generation chain. Across these settings, diffusion-style refinement serves as an effective spatial-reasoning mechanism when the generated entities have explicit geometric structure, particularly when reasoning is performed directly in the output domain.

Multi-agent trajectory prediction is a natural fit: each reverse step's $\hat{\mathbf{y}}^\tau$ already defines a temporary future scene in which Euclidean distances, relative motions, and group configurations, the quantities that govern social coherence, can be measured directly on the predicted coordinates. Our adapter accordingly treats denoising as progressive spatial reasoning: it rebuilds an explicit Euclidean interaction graph on $\hat{\mathbf{y}}^\tau$ at each step, runs uncertainty-weighted message passing over a sparse neighborhood, and returns a residual correction. Unlike scene-graph or layout diffusion, we operate directly in trajectory space, and in contrast to prior trajectory predictors, we enable future-conditioned in-

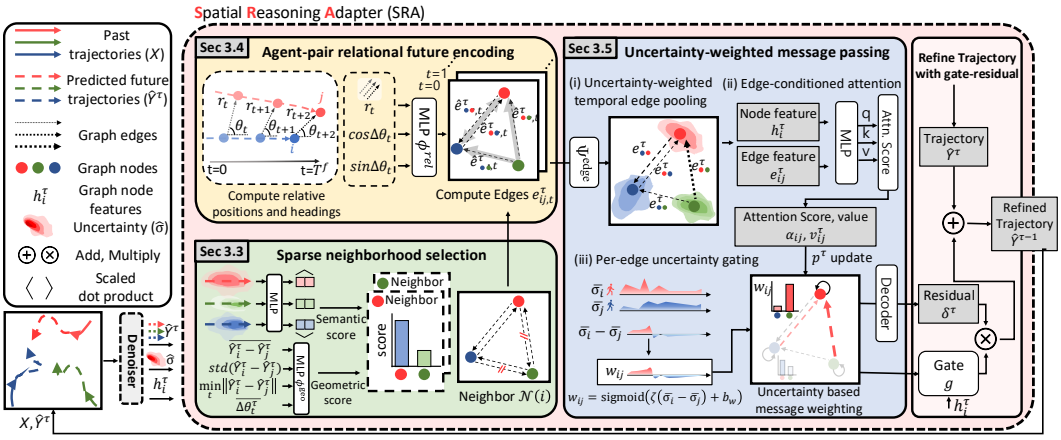

Figure 2: Overview of SRA at one reverse step τ. From the host's current future estimate $\hat{Y}^\tau$, node features h_i^τ, and uncertainty $\bar{\sigma}$ (bottom left), SRA **(green, Sec. 3.3)** scores every pair with a semantic and a geometric term and keeps the top-K_{nbr} neighbors $\mathcal{N}(i)$; **(yellow, Sec. 3.4)** encodes the per-timestep relative positions and headings of each selected pair into edge features $\hat{e}^\tau_{ij,t}$; **(blue, Sec. 3.5)** pools them over time with uncertainty-biased weights, uses them in edge-conditioned attention, and gates each message by $\bar{\sigma}_i - \bar{\sigma}_j$; and **(right)** decodes the aggregated message into a residual δ^τ that is added through the gate g to obtain the refined trajectory.

teractions at inference, where ground-truth futures are unavailable, by building the graph on the model's own estimate.

3 Method

3.1 Problem definition

Trajectory forecasting aims to learn a mapping from the observed trajectories of N agents $X = \{x_{i,t}\}_{i=1,\dots,N,\, t=T_p,\dots,0} \in \mathbb{R}^{N\times(T_p+1)\times 2}$ to a set of M plausible future trajectories $\hat{Y} = \{\hat{Y}^{(m)}\}_{m=1}^{M}$, $\hat{Y}^{(m)} = \{\hat{y}^{0,(m)}_{i,t}\}_{i=1,\dots,N,\, t=1,\dots,T_f} \in \mathbb{R}^{N\times T_f\times 2}$. Following recent diffusion and flow-matching formulations, the clean future y^0 is progressively noised across levels τ^k (from the fully noisy τ^K down to the clean $\tau^0 = 0$) and the denoiser is trained to reverse that process. In addition, we predict a per-agent, per-timestep uncertainty, parameterized as the log-variance $\mathbf{s}^\tau = \log\left((\sigma^\tau)^2\right) \in \mathbb{R}^{N\times T_f}$, which is used as a runtime signal that modulates how inter-agent messages are routed. We write $\hat{y}^\tau$ (or $\hat{y}^0$ when the step is clear) for the host's clean-future estimate at reverse step τ. All symbols are summarized in the notation table of Supp. A.

We treat the scene as a directed interaction graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with $|\mathcal{V}| = N$ nodes. The neighborhood $\mathcal{N}(i) = \{j \mid (j,i) \in \mathcal{E}\}$ is *dynamic*: it is rebuilt at every denoising step from the model's own current estimate of the future, so the graph's topology evolves as the prediction sharpens.

Fixed versus evolving interaction context. A history-based predictor computes its interaction context once, $c = f(X)$, and conditions every reverse step and every mode on it. (1) All M modes then share the same c, so interactions that differ between modes (a defender engaging attacker A in one mode and attacker B in another) must be represented with mode-independent features; (2) c is never revisited, so it cannot reflect the interaction structure that emerges as the futures are resolved. SRA instead builds $\mathcal{G}^{\tau,(m)}$ from each mode's current estimate $\hat{Y}^{\tau,(m)}$ at every reverse step, so interaction structure and trajectories sharpen together.

3.2 Overview: a modular graph denoiser

Our module takes the "current best guess" of the future that the host already produces, builds a sparse agent–pair graph on it, and emits a residual correction that the host injects back into its own denoising step. Concretely, at each reverse step τ the host denoiser produces (i) a current future

estimate $\hat{y}_i^\tau$ and (ii) a per-agent uncertainty σ_i^τ. Our module ingests these and returns $\delta_i^\tau \in \mathbb{R}^{T_f \times 2}$, a per-agent residual that is added as a gated correction to the host's prediction:

$$\tilde{y}_i^\tau = \hat{y}_i^\tau + g_i\, \delta_i^\tau, \qquad g_i = \mathrm{sigmoid}\big(W_g\,[h_i, \bar{m}_i]\big)\,, \tag{1}$$

where $g_i \in [0,1]$ is a learned scalar gate, $h_i \in \mathbb{R}^D$ denotes the host-provided agent feature of width D, W_g is a learned projection, and $\bar{m}_i$ denotes the temporally pooled graph message produced by our module. The refined estimate $\tilde{y}_i^\tau$ is passed back to the host for the reverse update. The module is a *retrieve-then-encode* pipeline of three components shared across hosts: **(a)** sparse neighborhood selection (Sec. 3.3) picks the few agents relevant to each target, **(b)** agent-pair relational future encoding (Sec. 3.4) builds edge features only for them, and **(c)** uncertainty-weighted message passing (Sec. 3.5) aggregates messages with confidence-aware weights. More details, including the host-specific integration, the residual decoder, and the training and inference algorithms for each host, are provided in the supplementary material (Supp. B).

3.3 Sparse neighborhood selection

A cheap score over all pairs selects a small relevant set, so that the expensive per-timestep encoder avoids both the quadratic cost of $N(N-1)$ pairs and the noise from irrelevant ones.

Future-conditioned graph topology. Instead of a topology fixed by the observed past, we build the graph on the clean future estimates $\hat{y}_i^0$; for ε-prediction hosts they are recovered from the noisy sample y^τ via the standard DDIM inversion

$$\hat{y}_i^0 = \big(y_i^\tau - \sqrt{1-\bar{\alpha}_\tau}\,\varepsilon_\theta(y^\tau, x)\big)\big/\sqrt{\bar{\alpha}_\tau}, \tag{2}$$

Two-term retrieval score. Based on the constructed topology, we select a sparse top-K_{nbr} neighborhood $\mathcal{N}(i)$ per target i using a score that combines a *semantic* (learned) affinity with a *geometric* (spatial) bias. Let $r_{ij,t}^\tau = \hat{y}_{j,t}^\tau - \hat{y}_{i,t}^\tau$ denote the per-pair, per-timestep relative position and $\Delta\theta_{ij,t}^\tau$ the corresponding heading difference (formalized in Eqs. 5–6). The score is

$$\mathrm{score}_{ij}^\tau = \underbrace{\frac{\langle W_q\, u_i^\tau,\, W_k\, u_j^\tau\rangle}{\sqrt{d_s}}}_{\text{semantic}} + \underbrace{\phi^{\mathrm{geo}}\big([\,\bar{r}_{ij}^\tau,\, \mathrm{std}(r_{ij}^\tau),\, \min_t \|r_{ij,t}^\tau\|,\, \overline{\Delta\theta_{ij}^\tau}\,]\big)}_{\text{geometric bias}}, \tag{3}$$

$$u_i^\tau = W_u\,[\hat{y}_{i,\cdot}^\tau,\, \sigma_{i,\cdot}^\tau], \tag{4}$$

where W_u, W_q, and W_k are learned projections, d_s is the score dimension, $\bar{\cdot}$ is the temporal mean, $\mathrm{std}(\cdot)$ is the per-axis temporal standard deviation, and ϕ^{geo} is an MLP.

The **semantic term** is a query–key match between *what i needs* (u_i^τ: its future plan and uncertainty) and *what j offers* (u_j^τ), so it can find relevance beyond proximity, e.g., two agents converging toward the same region while still far apart. The **geometric bias** supplies the explicit spatial prior it lacks through four cheap statistics of the relative trajectory: mean offset, motion variability, closest-approach distance, and mean heading difference. The top-K_{nbr} candidates by score_{ij}^τ form $\mathcal{N}(i)$ and the sparse edge set $\mathcal{E} = \{(j,i) \mid j \in \mathcal{N}(i)\}$; we use $K_{\mathrm{nbr}}{=}5$ throughout.

3.4 Agent-pair relational future encoding

For each selected pair (j,i), $j \in \mathcal{N}(i)$, we extract the relation in i's agent-centric frame. Let R_i be the rotation matrix that aligns i's heading at $t=0$ with the local x-axis, and let $\theta_{i,t}^0 = \mathrm{atan2}\big((\hat{y}_{i,t}^0 - \hat{y}_{i,t-1}^0)_y, (\hat{y}_{i,t}^0 - \hat{y}_{i,t-1}^0)_x\big)$ be the instantaneous heading, where $\hat{y}_{i,0}^\tau$ is anchored at the last observed position $x_{i,0}$. The per-pair, per-timestep ego-centric features are

$$\tilde{r}_{ij,t}^0 = R_i^\top\big(\hat{y}_{j,t}^0 - \hat{y}_{i,t}^0\big), \tag{5}$$

$$\Delta\theta_{ij,t}^0 = \theta_{j,t}^0 - \theta_{i,t}^0. \tag{6}$$

$\tilde{r}_{ij,t}^0$ is the ego-centric counterpart of $r_{ij,t}^0$ in Eq. 3, so interactions look the same regardless of scene orientation. A small MLP $\phi_\theta^{\mathrm{rel}}$ embeds $[\tilde{r}_{ij,t}^0, \cos\Delta\theta_{ij,t}^0, \sin\Delta\theta_{ij,t}^0]$ into a per-timestep future relation feature that we use as edge feature between agent nodes,

$$\hat{e}_{ij,t}^\tau = \phi_\theta^{\mathrm{rel}}\big([\tilde{r}_{ij,t}^0, \cos\Delta\theta_{ij,t}^0, \sin\Delta\theta_{ij,t}^0]\big) \in \mathbb{R}^D, \quad t = 1, \ldots, T_f, \tag{7}$$

one D-dimensional feature for every selected pair at every future timestep.

3.5 Uncertainty-weighted message passing

We add a log-variance head that predicts the uncertainty σ^τ. A confident agent carries reliable evidence about the joint future that uncertain neighbors can lean on, so we modulate messages by uncertainty to make confident agents anchors for uncertain ones. The graph-attention layer operates on host-provided agent features $h_i^\tau \in \mathbb{R}^D$ and pooled edge features $e_{ij}^\tau \in \mathbb{R}^D$ (edge $j \to i$), and uncertainty enters at three points.

(i) Uncertainty-weighted temporal edge pooling. The per-timestep edge features $\{\hat{e}_{ij,t}^\tau\}_{t=1}^{T_f}$ from Eq. 7 are collapsed into a single pooled edge vector $e_{ij}^\tau \in \mathbb{R}^D$ used by message passing. The encoder Ψ^{edge} first applies a temporal self-attention to mix information across the T_f frames, producing $p_{ij,t}^\tau = \mathrm{SelfAttn}(\hat{e}_{ij,\cdot}^\tau)_t$. A learned attention-pooling head then computes per-timestep weights, into which we inject the per-timestep relative uncertainty $\Delta\sigma_{ij,t}^\tau = \sigma_{i,t}^\tau - \sigma_{j,t}^\tau$ as an additive bias before softmax,

$$p_{ij,1:T_f}^\tau = \Psi^{\text{edge}}\left(\hat{e}_{ij,1:T_f}^\tau\right), \quad w_{ij,t}^\tau = \mathrm{softmax}_t\left(\phi^{\text{pool}}(p_{ij,t}^\tau) + \Delta\sigma_{ij,t}^\tau\right), \quad e_{ij}^\tau = \sum_{t=1}^{T_f} w_{ij,t}^\tau p_{ij,t}^\tau. \tag{8}$$

so frames in which the target i is uncertain and the source j is confident receive larger pooling weight.

(ii) Edge-conditioned attention. Queries are formed from the target node i and keys/values from each source neighbor $j \in \mathcal{N}(i)$, with the future-trajectory edge feature e_{ij}^τ from Eq. 8 entering keys and values but not queries:

$$q_i^\tau = W_q\, h_i^\tau + b_q(\psi^\tau), \quad k_{ij}^\tau = W_k\, h_j^\tau + b_k(\psi^\tau) + W_k^{(e)} e_{ij}^\tau, \quad v_{ij}^\tau = W_v\, h_j^\tau + b_v(\psi^\tau) + W_v^{(e)} e_{ij}^\tau, \tag{9}$$

where $W_{\{q,k,v\}}$ and $W_{\{k,v\}}^{(e)}$ are learned projections, $b_{\{q,k,v\}}(\cdot)$ are learned biases conditioned on the noise level, and ψ^τ is the noise-level embedding.

(iii) Per-edge uncertainty gating. The per-timestep $\sigma_{i,t}^\tau$ is too jittery to drive a per-edge gate, so we average it over the horizon, $\bar{\sigma}_i^\tau = \frac{1}{T_f}\sum_t \sigma_{i,t}^\tau$, and derive a scalar message gate

$$w_{ij}^\tau = \mathrm{sigmoid}\big(\zeta(\bar{\sigma}_i^\tau - \bar{\sigma}_j^\tau) + b_w\big), \tag{10}$$

where $\zeta{=}4$ is a fixed temperature and $b_w{=}0.5$ a fixed offset, so w_{ij}^τ grows when target i is more uncertain than source j and equals $\mathrm{sigmoid}(0.5) \approx 0.62$ for equally confident agents.

Aggregation. Standard multi-head attention weights $\alpha_{ij}^\tau = \mathrm{softmax}_{j\in\mathcal{N}(i)}\left((q_i^\tau)^\top k_{ij}^\tau/\sqrt{d}\right)$, with per-head dimension d, are combined with the confidence gate w_{ij}^τ from Eq. 10 and the values v_{ij}^τ from Eq. 9 into a message:

$$m_i^\tau = \sum_{j\in\mathcal{N}(i)} w_{ij}^\tau\, \alpha_{ij}^\tau\, v_{ij}^\tau, \qquad h_i^{\tau,(l+1)} = h_i^{\tau,(l)} + m_i^\tau + \phi_\theta^m\big(\mathrm{LN}\big(h_i^{\tau,(l)} + m_i^\tau\big)\big). \tag{11}$$

where ϕ_θ^m is a feed-forward block and LN layer normalization. After L layers, a host-specific decoder ϕ_θ^{dec} (Supp. B.2) maps the refined embedding to the residual δ_i^τ of Eq. 1.

3.6 Instantiation in host denoisers

Host requirements. SRA assumes that the host (i) iteratively refines a continuous future estimate, (ii) exposes a prediction or intermediate representation into which a correction can be injected, and (iii) provides agent features from which an uncertainty head can be trained (it need not be native to the host). A reverse chain satisfies these naturally, since it denoises all agents' futures jointly and exposes them at every step. SRA is *modular*, as it does not redesign the host's generative core, but *not training-free*: it needs a host-specific injection point and is trained with the host (Sec. 3.7). Regression- and classification-based predictors are left to future work.

Table 1: **Main results.** ADE / FDE (best of $K=20$ modes; lower is better) of three hosts with and without SRA on NBA, Soccer, Football, and SDD. Percentages are the relative improvement of SRA over its host; the only negative value (MID, FDE on SDD) is shown in black.

	NBA		Soccer		Football		SDD	
Method	ADE	FDE	ADE	FDE	ADE	FDE	ADE	FDE
MID [15]	1.037	1.472	0.451	0.783	0.274	0.459	8.85	15.92
MID + SRA	0.957 (+7.7%)	1.346 (+8.6%)	0.370 (+18.0%)	0.595 (+24.0%)	0.265 (+3.3%)	0.438 (+4.6%)	8.53 (+3.6%)	16.61 (−4.3%)
LED [37]	0.839	1.204	0.372	0.821	0.289	0.580	8.14	13.95
LED + SRA	0.778 (+7.3%)	1.161 (+3.6%)	0.364 (+2.2%)	0.710 (+13.5%)	0.253 (+12.5%)	0.505 (+12.9%)	7.91 (+2.8%)	13.31 (+4.6%)
MoFlow [14]	0.703	0.884	0.406	0.593	0.251	0.421	5.44	8.30
MoFlow + SRA	0.695 (+1.1%)	0.870 (+1.6%)	0.374 (+7.9%)	0.522 (+12.0%)	0.245 (+2.4%)	0.412 (+2.1%)	5.37 (+1.3%)	8.21 (+1.1%)

Hosts. We use three hosts: **MID** [15], a vanilla DDPM with a pure transformer denoiser and no built-in inter-agent reasoning; **LED** [37], a leapfrog-initialized DDPM that collapses the reverse chain to five fine-grained ε-prediction steps; and **MoFlow** [14], a flow-matching model whose Motion-Transformer decoder directly regresses clean trajectories with factorized K-attention (modes) and A-attention (agents). Implementation details for each host are in Supp. B.

3.7 Training

SRA leaves the host's generative objective $\mathcal{L}_{\text{gen}}$ (e.g., DDPM ε-MSE or flow matching) unchanged and only adds a single heteroscedastic NLL term that trains the log-variance head alongside it. Given the host's prediction residual r_θ and its predicted variance σ_θ^2, we minimize

$$\mathcal{L}_\sigma = \tfrac{1}{2}\mathbb{E}\big[\log \sigma_\theta^2 + \|r_\theta\|^2/\sigma_\theta^2\big]\,, \tag{12}$$

where r_θ is the noise residual $\varepsilon - \varepsilon_\theta$ for noise-prediction hosts or the trajectory residual $y^0 - \hat{y}_\theta^0$ for flow-matching hosts. The total objective is $\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{gen}} + \lambda_\sigma \mathcal{L}_\sigma$ with $\lambda_\sigma{=}10^{-2}$. We detach r_θ from the NLL gradient path so $\mathcal{L}_\sigma$ trains only the log-variance head and leaves the host's denoising parameters untouched.

4 Experiments

4.1 Dataset

We use 4 datasets to evaluate the prediction performance: SDD [48], NBA [72], Soccer [52], Football [38]. While SDD is widely adopted as a benchmark for trajectory prediction, its patterns are monotonous as it only includes pedestrians walking. On the other hand, sports datasets (NBA, Soccer, Football) consist of dynamic motion and interaction, greatly increasing the difficulty of predicting future motion. Therefore, we focus on the sports datasets: the component analysis is run on all three sports datasets, and the finer-grained ablations on NBA, the most demanding of them, where the court is small relative to the number of agents, nearly every player is within interaction range of several others, and configurations turn over quickly across possessions. For sports datasets, we vary the frame rate (5Hz for NBA, 5Hz for football, 3.33Hz for soccer) with 10 past frames to predict 20 frames. For SDD, we follow the standard protocol of 8 past frames (3.2 s) and 12 future frames (4.8 s) at 2.5 Hz.

4.2 Metrics

We report Average Displacement Error (ADE) and Final Displacement Error (FDE), computed as the minimum over $K{=}20$ predicted modes following the standard protocol of prior work [37, 15, 14]. We do not use collision rate: physical contact between players is a normal part of basketball, soccer, and football, so a collision counter would penalize predictions that are in fact correct.

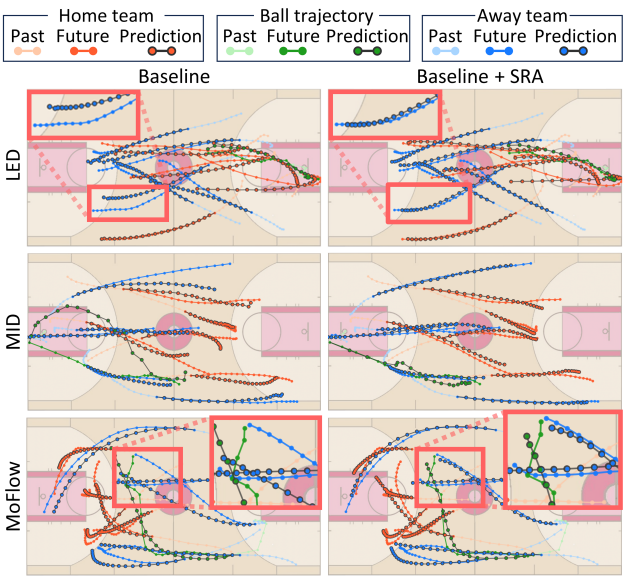

Figure 3: Qualitative comparison on NBA. Rows: LED, MID, MoFlow; left: host, right: host + SRA. Orange, green, and blue are home players, the ball, and away players; light, medium, and dark dots are the observed past, the ground-truth future, and the prediction (best of $K=20$ modes by ADE). Red boxes mark where the predictions differ.

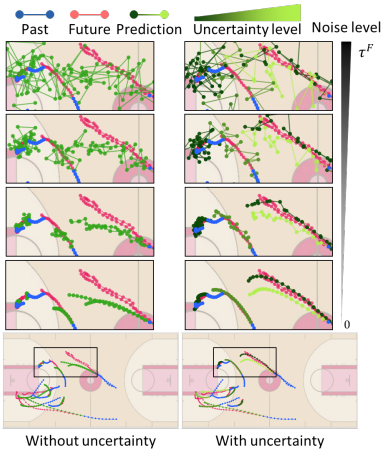

Figure 4: Denoising without (left) and with (right) uncertainty, LED on NBA. Bottom: full scene; above: the boxed region from high noise (top) to the final step. Blue: past; pink: ground truth; green: prediction, shaded by predicted σ (right).

5 Results

5.1 Quantitative results

Table 1 compares SRA against generative baselines on sports (NBA, Soccer, Football) and pedestrian (SDD) benchmarks. SRA improves ADE for every host on every dataset and FDE in 11 of 12 host–dataset pairs, by 5.8% ADE and 7.0% FDE on average, with the largest gains on sports scenes: up to 18.0% ADE and 24.0% FDE over the host (MID on Soccer). Sports scenes involve dense, fast, ball-coupled interactions where future-conditioned coherence matters most, whereas pedestrian dynamics are comparatively monotone and leave less headroom for graph corrections.

Further experiments are provided in the supplementary material: an analysis with the joint metrics JADE/JFDE [62] (Supp. D), a comparison with future-aware interaction modules (Supp. E), and the inference latency of SRA (Supp. G).

5.2 Qualitative results

Figure 3 compares SRA's predictions against the baseline on three representative NBA scenes. In the first row (LED), SRA correctly predicts the defender's return-to-defense motion that the baseline misses. In the second row (MID), SRA captures a tight ball-carrier coupling, so the ball's trajectory tracks its carrier rather than drifting independently. In the third row (MoFlow), SRA produces an accurate offensive sequence in which the ball is passed cleanly along the attacking players.

5.3 Ablations

Cumulative component analysis. Table 2 adds the three components one at a time on all three sports datasets. Leveraging interaction from the predicted future is what produces the gain: relational encoding on a dense graph already improves ADE in 6 of the 9 host–dataset pairs. Using that signal well, however, requires the other two components. Sparsifying without uncertainty is close to a coin flip (better than the dense graph in 4 pairs and worse in 5) because top-K_{nbr} selection commits hard to a still-noisy estimate, and a wrong choice discards the neighbors that mattered. Uncertainty tells the module how far to trust that estimate, and only with all three components does SRA improve all nine pairs in both metrics. The components are therefore one mechanism rather than independent

Table 2: **Cumulative component analysis** (ADE / FDE). Starting from the unmodified host, we add one SRA component at a time: future relational encoding over all neighbors (Enc.), sparse top-K_{nbr} neighbor selection (Sel.), and uncertainty-weighted message passing (Unc.). The last row of each block is the full SRA reported in Table 1. Best per dataset and host in **bold**.

Dataset	Configuration	Enc.	Sel.	Unc.	MID	LED	MoFlow
NBA	Host baseline				1.037 / 1.472	0.839 / 1.204	0.703 / 0.884
	+ encoding (all neighbors)	✓			1.005 / 1.417	**0.776** / 1.176	0.698 / 0.876
	+ sparse selection	✓	✓		0.963 / 1.367	0.886 / 1.238	0.705 / 0.881
	+ uncertainty (full SRA)	✓	✓	✓	**0.957 / 1.346**	0.778 / **1.161**	**0.695 / 0.870**
Soccer	Host baseline				0.451 / 0.783	0.372 / 0.821	0.406 / 0.593
	+ encoding (all neighbors)	✓			0.438 / 0.648	0.384 / 0.748	0.386 / **0.520**
	+ sparse selection	✓	✓		0.455 / 0.736	0.383 / 0.822	0.382 / 0.533
	+ uncertainty (full SRA)	✓	✓	✓	**0.370 / 0.595**	**0.364 / 0.710**	**0.374** / 0.522
Football	Host baseline				0.274 / 0.459	0.289 / 0.580	0.251 / 0.421
	+ encoding (all neighbors)	✓			0.269 / 0.454	0.305 / 0.682	0.251 / **0.395**
	+ sparse selection	✓	✓		0.296 / 0.475	0.306 / 0.643	0.247 / 0.422
	+ uncertainty (full SRA)	✓	✓	✓	**0.265 / 0.438**	**0.253 / 0.505**	**0.245** / 0.412

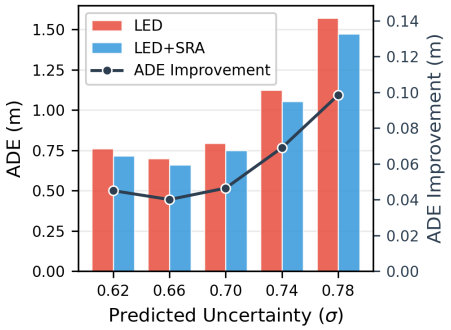

Figure 5: ADE of LED vs. LED+SRA stratified by predicted uncertainty σ on the NBA dataset (bars, left axis), and the ADE improvement of LED+SRA over LED (line, right axis). The baseline error grows with σ, and SRA's gain appears in every bin and is largest in the highest-σ bin.

additions: the future interaction signal, the selection that concentrates it, and the confidence estimate that makes the selection safe. This also explains why ablating a single design choice of the full model (Tables 3 and 4) changes the result by a smaller margin than the full gain over the host: it measures a broken dependency rather than the size of an independent contribution.

Use of uncertainty. The last two rows of each block in Table 2 isolate uncertainty-weighted message passing, which improves ADE and FDE for every host on every sports dataset, with the largest relative ADE reductions for MID on Soccer ($0.455 \rightarrow 0.370$, 18.7%) and LED on Football ($0.306 \rightarrow 0.253$, 17.3%). Each message is modulated by the *difference* $\sigma_i - \sigma_j$ (Eqs. 8 and 10), so what matters is which of the two agents the host currently localizes better: the more certain agent in each pair becomes the anchor that the uncertain one leans on, and relations propagate in the direction in which they can be trusted. Figure 5 stratifies NBA scenes into five bins by predicted σ. The LED baseline ADE grows monotonically with σ, so σ is well-calibrated; the improvement of LED + SRA appears in every bin and rises with σ, reaching more than $2\times$ the lowest-bin gain in the highest bin, so SRA concentrates its corrections where the host is least reliable. Figure 4 shows the effect qualitatively: SRA assigns the highest σ to the ball carrier and its defender, the two most coupled agents, and routes messages from more confident players to them, so the predicted paths follow the actual pass to the corner; without uncertainty (left), the ball drifts into an implausible trajectory that misses the pass.

Relational features. Table 3 compares four configurations of pairwise relational features used in the future relational feature $\hat{e}^{\tau}_{ij,t}$. Using only L2 distance, the simplest geometric cue, already yields gains over the base denoisers, confirming that *any* explicit future-trajectory interaction improves over latent-only attention. Relative position adds directional information and improves FDE in MID and LED, while heading direction alone provides comparable ADE but stronger FDE gains in MoFlow. Combining relative position with heading direction (our default) achieves the best balance across all

Relational feat.			MID		LED		MoFlow	
L2	Pos.	Head.	ADE	FDE	ADE	FDE	ADE	FDE
✓			0.968	1.376	0.781	1.164	0.699	0.878
	✓		0.966	1.360	**0.778**	**1.153**	0.698	0.880
		✓	0.967	1.353	0.784	1.163	0.698	0.872
	✓	✓	**0.957**	**1.346**	**0.778**	1.161	**0.695**	**0.870**

Table 3: **Relational features** used in $\hat{e}^{\tau}_{ij,t}$ (Eq. 7) on NBA: L2 distance (L2), relative position (Pos.), heading difference (Head.), or relative position + heading (ours, last row). ADE / FDE, best of 20; best per column in **bold**.

Rule-based	Ours		MID		LED		MoFlow	
L2	Geo.	Sem.	ADE	FDE	ADE	FDE	ADE	FDE
✓			0.963	1.367	0.780	1.182	0.701	0.881
	✓		0.959	1.348	0.782	1.172	0.697	0.873
		✓	0.960	1.351	0.781	1.169	0.696	0.872
	✓	✓	**0.957**	**1.346**	**0.778**	**1.161**	**0.695**	**0.870**

Table 4: **Neighbor scoring** on NBA. Rule-based: the K_{nbr} closest agents by mean future L2 distance (L2). Ours: the geometric (Geo.) and semantic (Sem.) terms of our learned retrieval score (Eq. 3), used alone or together (full SRA, last row). ADE / FDE, best of 20.

three backbones, as it captures both where agents are heading and how they are oriented relative to each other.

Neighbor selection method. Table 4 compares our learned two-term retrieval score (Eq. 3) against a rule-based selection of the K_{nbr} agents closest in mean future L2 distance, and against each of its two terms alone. Either learned term alone already improves FDE over the rule-based selection for every host, and the two perform almost identically, but their combination is best in every cell. The terms are complementary: the geometric bias supplies a proximity prior that the semantic term has no explicit access to, while the semantic query–key match captures relevance that distance misses, such as two agents converging on the same region while still far apart. The gaps are modest, indicating that the graph is fairly robust to the selection rule.

6 Conclusion

We presented SRA, a modular graph denoiser that is integrated into iterative stochastic trajectory predictors and reasons about inter-agent interaction directly in the Euclidean future trajectory space. At every reverse step, SRA rebuilds an explicit interaction graph on the host's current future estimate, runs uncertainty-weighted message passing over a sparse K_{nbr}-neighborhood, and emits a residual correction that the host injects back into its denoising update without redesigning the generative core. Our cumulative ablation shows that its three components (sparse neighborhood selection, agent-pair relational future encoding, and uncertainty-weighted message passing) work as one mechanism, and only their combination improves every host on every sports dataset. Instantiated on three structurally different hosts (MID, LED, MoFlow), SRA improves every host on every dataset across four multi-agent benchmarks (NBA, Soccer, Football, SDD) in ADE, at a cost of a few milliseconds per scene.

Limitations. SRA is modular but not training-free: it requires a host-specific injection point and joint training or fine-tuning with the host. It assumes an iterative host that refines continuous future estimates, so extending it to regression- or classification-based predictors remains future work. Its graph is built on the host's own estimate and inherits that estimate's errors, and our evaluation does not cover mixed-class scenes with map-dependent interactions. Supp. H discusses these points in detail.

Acknowledgements

This work was supported by the Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (IITP-2026-RS-2026-25617720, AI Star Fellowship Support Program, Contribution Rate: 40%), and by the Culture, Sports and Tourism R&D Program, funded by the Ministry of Culture, Sports and Tourism, through the Korea Culture Technology Planning and Evaluation Institute, an affiliated institute of the Korea Creative Content Agency in 2026 (Project Name: Development of Outpainting AI and spatial expansion technology for utilizing archive video contents, Project Number: RS-2026-25519568, Contribution Rate: 50%), and by the Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (RS-2025-25420118, Contribution Rate: 10%).

References

[1] Alexandre Alahi, Kratarth Goel, Vignesh Ramanathan, Alexandre Robicquet, Li Fei-Fei, and Silvio Savarese. Social LSTM: Human trajectory prediction in crowded spaces. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 961–971, 2016.

[2] Inhwan Bae, Jin-Hwi Park, and Hae-Gon Jeon. Non-probability sampling network for stochastic human trajectory prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6477–6487, 2022.

[3] Inhwan Bae, Young-Jae Park, and Hae-Gon Jeon. SingularTrajectory: Universal trajectory predictor using diffusion model. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 17890–17901, 2024.

[4] Guillem Capellera, Antonio Rubio, Luis Ferraz, and Antonio Agudo. Unified uncertainty-aware diffusion for multi-agent trajectory modeling. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 22476–22486, 2025.

[5] Guillem Capellera, Luis Ferraz, Antonio Rubio, Alexandre Alahi, and Antonio Agudo. JointDiff: Bridging continuous and discrete in multi-agent trajectory generation. In *International Conference on Learning Representations*, 2026.

[6] Boyuan Chen, Diego Martí Monsó, Yilun Du, Max Simchowitz, Russ Tedrake, and Vincent Sitzmann. Diffusion forcing: Next-token prediction meets full-sequence diffusion. *Advances in Neural Information Processing Systems*, 37:24081–24125, 2024.

[7] Kai Chen, Xiaodong Zhao, Yujie Huang, Guoyu Fang, Xiao Song, Ruiping Wang, and Ziyuan Wang. SocialMOIF: Multi-order intention fusion for pedestrian trajectory prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 22465–22475, 2025.

[8] Mu Chen, Liulei Li, Wenguan Wang, and Yi Yang. DIFFVSGG: Diffusion-driven online video scene graph generation. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 29161–29172, 2025.

[9] Pranav Singh Chib and Pravendra Singh. Pedestrian trajectory prediction with missing data: Datasets, imputation, and benchmarking. *Advances in Neural Information Processing Systems*, 37:124530–124546, 2024.

[10] Patrick Dendorfer, Sven Elflein, and Laura Leal-Taixé. MG-GAN: A multi-generator model preventing out-of-distribution samples in pedestrian trajectory prediction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 13158–13167, 2021.

[11] Liangji Fang, Qinhong Jiang, Jianping Shi, and Bolei Zhou. TPNet: Trajectory proposal network for motion prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6797–6806, 2020.

[12] Zilin Fang, David Hsu, and Gim Hee Lee. Neuralized Markov random field for interaction-aware stochastic human trajectory prediction. In *The Thirteenth International Conference on Learning Representations*, 2025. URL `https://openreview.net/forum?id=r3cEOVj7Ze`.

[13] Azade Farshad, Yousef Yeganeh, Yu Chi, Chengzhi Shen, Björn Ommer, and Nassir Navab. SceneGenie: Scene graph guided diffusion models for image synthesis. In *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops*, pages 88–98, 2023.

[14] Yuxiang Fu, Qi Yan, Lele Wang, Ke Li, and Renjie Liao. MoFlow: One-step flow matching for human trajectory forecasting via implicit maximum likelihood estimation based distillation. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 17282–17293, 2025.

[15] Tianpei Gu, Guangyi Chen, Junlong Li, Chunze Lin, Yongming Rao, Jie Zhou, and Jiwen Lu. Stochastic trajectory prediction via motion indeterminacy diffusion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 17113–17122, 2022.

[16] Xunjiang Gu, Guanyu Song, Igor Gilitschenski, Marco Pavone, and Boris Ivanovic. Producing and leveraging online map uncertainty in trajectory prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14521–14530, 2024.

[17] Agrim Gupta, Justin Johnson, Li Fei-Fei, Silvio Savarese, and Alexandre Alahi. Social GAN: Socially acceptable trajectories with generative adversarial networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2255–2264, 2018.

[18] Xin Hu, Ke Qin, Guiduo Duan, Ming Li, Yuan-Fang Li, and Tao He. SPADE: Spatial-aware denoising network for open-vocabulary panoptic scene graph generation with long-and local-range context reasoning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 15562–15572, 2025.

[19] Yue Hu, Siheng Chen, Ya Zhang, and Xiao Gu. Collaborative motion prediction via neural motion message passing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6319–6328, 2020.

[20] Zhiyu Huang, Haochen Liu, and Chen Lv. GameFormer: Game-theoretic modeling and learning of transformer-based interactive prediction and planning for autonomous driving. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3903–3913, 2023.

[21] Jaewoo Jeong, Daehee Park, and Kuk-Jin Yoon. Multi-agent long-term 3D human pose forecasting via interaction-aware trajectory conditioning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1617–1628, 2024.

[22] Jaewoo Jeong, Seohee Lee, Daehee Park, Giwon Lee, and Kuk-Jin Yoon. Multi-modal knowledge distillation-based human trajectory forecasting. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 24222–24233, 2025.

[23] Xiaosong Jia, Li Chen, Penghao Wu, Jia Zeng, Junchi Yan, Hongyang Li, and Yu Qiao. Towards capturing the temporal dynamics for trajectory prediction: a coarse-to-fine approach. In *Conference on Robot Learning*, pages 910–920. PMLR, 2023.

[24] Chiyu Jiang, Andre Cornman, Cheolho Park, Benjamin Sapp, Yin Zhou, Dragomir Anguelov, et al. MotionDiffuser: Controllable multi-agent motion prediction using diffusion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9644–9653, 2023.

[25] Sungjune Kim, Hyung-gun Chi, Hyerin Lim, Karthik Ramani, Jinkyu Kim, and Sangpil Kim. Higher-order relational reasoning for pedestrian trajectory prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15251–15260, 2024.

[26] Giwon Lee, Wooseong Jeong, Daehee Park, Jaewoo Jeong, and Kuk-Jin Yoon. Interaction-merged motion planning: Effectively leveraging diverse motion datasets for robust planning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 28610–28621, 2025.

[27] Giwon Lee, Daehee Park, Jaewoo Jeong, and Kuk-Jin Yoon. Non-differentiable reward optimization for diffusion-based autonomous motion planning. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 13215–13222, 2025.

[28] Namhoon Lee, Wongun Choi, Paul Vernaza, Christopher B Choy, Philip HS Torr, and Manmohan Chandraker. DESIRE: Distant future prediction in dynamic scenes with interacting agents. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 336–345, 2017.

[29] Junwei Liang, Lu Jiang, Kevin Murphy, Ting Yu, and Alexander Hauptmann. The garden of forking paths: Towards multi-future trajectory prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10508–10518, 2020.

[30] Haicheng Liao, Xuelin Li, Yongkang Li, Hanlin Kong, Chengyue Wang, Bonan Wang, Yanchen Guan, K Tam, and Zhenning Li. CDSTraj: Characterized diffusion and spatial-temporal interaction network for trajectory prediction in autonomous driving. In *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence, IJCAI-24*, pages 7331–7339, 2024.

[31] Haozhe Lin, Chunyu Wei, Li He, Yuchen Guo, Yunqi Zhao, Shanglong Li, and Lu Fang. GigaTraj: Predicting long-term trajectories of hundreds of pedestrians in gigapixel complex scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 19331–19340, 2024.

[32] Jinxiu Liu and Qi Liu. R3CD: Scene graph to image generation with relation-aware compositional contrastive control diffusion. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 3657–3665, 2024.

[33] 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 *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 2039–2049, 2024.

[34] Yu Liu, Zhijie Liu, Xiao Ren, You-Fu Li, and He Kong. Intention enhanced diffusion model for multi-modal pedestrian trajectory prediction. In *IEEE International Conference on Intelligent Transportation Systems (ITSC)*, pages 2063–2068, 2025.

[35] Katie Luo, Sergio Casas, Renjie Liao, Xinchen Yan, Yuwen Xiong, Wenyuan Zeng, and Raquel Urtasun. Safety-oriented pedestrian occupancy forecasting. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1015–1022. IEEE, 2021.

[36] Wenjie Luo, Cheol Park, Andre Cornman, Benjamin Sapp, and Dragomir Anguelov. JFP: Joint future prediction with interactive multi-agent modeling for autonomous driving. In *Conference on Robot Learning*, pages 1457–1467. PMLR, 2023.

[37] Weibo Mao, Chenxin Xu, Qi Zhu, Siheng Chen, and Yanfeng Wang. Leapfrog diffusion model for stochastic trajectory prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5517–5526, 2023.

[38] National Football League. Big Data Bowl. `https://github.com/nfl-football-ops/Big-Data-Bowl`, 2019. Accessed: 2025-11-06.

[39] Nigamaa Nayakanti, Rami Al-Rfou, Aurick Zhou, Kratarth Goel, Khaled S Refaat, and Benjamin Sapp. Wayformer: Motion forecasting via simple & efficient attention networks. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 2980–2987, 2023.

[40] Marion Neumeier, Sebastian Dorn, Michael Botsch, and Wolfgang Utschick. Reliable trajectory prediction and uncertainty quantification with conditioned diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 3461–3470, 2024.

[41] Jiquan Ngiam, Benjamin Caine, Vijay Vasudevan, Zhengdong Zhang, Hao-Tien Lewis Chiang, Jeffrey Ling, Rebecca Roelofs, Alex Bewley, Chenxi Liu, Ashish Venugopal, et al. Scene Transformer: A unified architecture for predicting multiple agent trajectories. In *International Conference on Learning Representations*, 2022.

[42] Bo Pang, Tianyang Zhao, Xu Xie, and Ying Nian Wu. Trajectory prediction with latent belief energy-based model. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11814–11824, 2021.

[43] Daehee Park, Hobin Ryu, Yunseo Yang, Jegyeong Cho, Jiwon Kim, and Kuk-Jin Yoon. Leveraging future relationship reasoning for vehicle trajectory prediction. In *International Conference on Learning Representations*, 2023.

[44] Daehee Park, Jaeseok Jeong, Sung-Hoon Yoon, Jaewoo Jeong, and Kuk-Jin Yoon. T4P: Test-time training of trajectory prediction via masked autoencoder and actor-specific token memory. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15065–15076, 2024.

[45] Daehee Park, Jaewoo Jeong, and Kuk-Jin Yoon. Improving transferability for cross-domain trajectory prediction via neural stochastic differential equation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 10145–10154, 2024.

[46] Muleilan Pei, Shaoshuai Shi, Xuesong Chen, Xu Liu, and Shaojie Shen. Foresight in motion: Reinforcing trajectory prediction with reward heuristics. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 28303–28312, 2025.

[47] Tran Phong, Haoran Wu, Cunjun Yu, Panpan Cai, Sifa Zheng, and David Hsu. What truly matters in trajectory prediction for autonomous driving? *Advances in Neural Information Processing Systems*, 36: 71327–71339, 2023.

[48] Alexandre Robicquet, Amir Sadeghian, Alexandre Alahi, and Silvio Savarese. Learning social etiquette: Human trajectory understanding in crowded scenes. In *European Conference on Computer Vision*, pages 549–565. Springer, 2016.

[49] Luke Rowe, Martin Ethier, Eli-Henry Dykhne, and Krzysztof Czarnecki. FJMP: Factorized joint multi-agent motion prediction over learned directed acyclic interaction graphs. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13745–13755, 2023.

[50] David Ruhe, Jonathan Heek, Tim Salimans, and Emiel Hoogeboom. Rolling diffusion models. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp, editors, *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 42818–42835. PMLR, 21–27 Jul 2024. URL `https://proceedings.mlr.press/v235/ruhe24a.html`.

[51] Tim Salzmann, Hao-Tien Lewis Chiang, Markus Ryll, Dorsa Sadigh, Carolina Parada, and Alex Bewley. Robots that can see: Leveraging human pose for trajectory prediction. *IEEE Robotics and Automation Letters*, 8(11):7090–7097, 2023.

[52] Atom Scott, Ikuma Uchida, Masaki Onishi, Yoshinari Kameda, Kazuhiro Fukui, and Keisuke Fujii. SoccerTrack: A dataset and tracking algorithm for soccer with fish-eye and drone videos. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 3569–3579, 2022.

[53] Shaoshuai Shi, Li Jiang, Dengxin Dai, and Bernt Schiele. Motion transformer with global intention localization and local movement refinement. *Advances in Neural Information Processing Systems*, 35: 6531–6543, 2022.

[54] Shaoshuai Shi, Li Jiang, Dengxin Dai, and Bernt Schiele. MTR++: Multi-agent motion prediction with symmetric scene modeling and guided intention querying. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 46(5):3955–3971, 2024.

[55] Hao Sun, Zhiqun Zhao, and Zhihai He. Reciprocal learning networks for human trajectory prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7416–7425, 2020.

[56] Jiankai Sun, Yiqi Jiang, Jianing Qiu, Parth Nobel, Mykel J Kochenderfer, and Mac Schwager. Conformal prediction for uncertainty-aware planning with diffusion dynamics model. *Advances in Neural Information Processing Systems*, 36:80324–80337, 2023.

[57] Bohan Tang, Yiqi Zhong, Ulrich Neumann, Gang Wang, Siheng Chen, and Ya Zhang. Collaborative uncertainty in multi-agent trajectory forecasting. *Advances in Neural Information Processing Systems*, 34:6328–6340, 2021.

[58] Xiaolong Tang, Meina Kan, Shiguang Shan, Zhilong Ji, Jinfeng Bai, and Xilin Chen. HPNet: Dynamic trajectory forecasting with historical prediction attention. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15261–15270, 2024.

[59] Neerja Thakkar, Tara Sadjadpour, Jathushan Rajasegeran, Shiry Ginosar, and Jitendra Malik. Poly-autoregressive prediction for modeling interactions. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 12402–12412, 2025.

[60] Ruisi Wang, Zhongang Cai, Fanyi Pu, Junxiang Xu, Wanqi Yin, Maijunxian Wang, Ran Ji, Chenyang Gu, Bo Li, Ziqi Huang, et al. Demystifying video reasoning. *arXiv preprint arXiv:2603.16870*, 2026.

[61] Di Wen, Haoran Xu, Zhaocheng He, Zhe Wu, Guang Tan, and Peixi Peng. Density-adaptive model based on motif matrix for multi-agent trajectory prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14822–14832, 2024.

[62] Erica Weng, Hana Hoshino, Deva Ramanan, and Kris Kitani. Joint metrics matter: A better standard for trajectory forecasting. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 20315–20326, 2023.

[63] Christopher Wewer, Bartlomiej Pogodzinski, Bernt Schiele, and Jan Eric Lenssen. Spatial reasoning with denoising models. In *International Conference on Machine Learning*, 2025.

[64] Conghao Wong, Beihao Xia, Ziqian Zou, Yulong Wang, and Xinge You. SocialCircle: Learning the angle-based social interaction representation for pedestrian trajectory prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 19005–19015, 2024.

[65] Tong Wu, Zhihao Fan, Xiao Liu, Hai-Tao Zheng, Yeyun Gong, Yelong Shen, Jian Jiao, Juntao Li, Zhongyu Wei, Jian Guo, Nan Duan, and Weizhu Chen. AR-Diffusion: Auto-regressive diffusion model for text generation. *Advances in Neural Information Processing Systems*, 36:39957–39974, 2023.

[66] Liwen Xiao, Zhiyu Pan, Zhicheng Wang, Zhiguo Cao, and Wei Li. SRefiner: Soft-braid attention for multi-agent trajectory refinement. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 960–969, 2025.

[67] Chenxin Xu, Maosen Li, Zhenyang Ni, Ya Zhang, and Siheng Chen. GroupNet: Multiscale hypergraph neural networks for trajectory prediction with relational reasoning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6498–6507, 2022.

[68] Chenxin Xu, Weibo Mao, Wenjun Zhang, and Siheng Chen. Remember intentions: Retrospective-memory-based trajectory prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6488–6497, 2022.

[69] Chenxin Xu, Robby T. Tan, Yuhong Tan, Siheng Chen, Yu Guang Wang, Xinchao Wang, and Yanfeng Wang. EqMotion: Equivariant multi-agent motion prediction with invariant interaction reasoning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1410–1420, 2023.

[70] Songru Yang, Zhenwei Shi, and Zhengxia Zou. Unified multi-agent trajectory modeling with masked trajectory diffusion. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 27563–27574, 2025.

[71] Guangyao Zhai, Evin Pınar Örnek, Shun-Cheng Wu, Yan Di, Federico Tombari, Nassir Navab, and Benjamin Busam. CommonScenes: Generating commonsense 3D indoor scenes with scene graph diffusion. *Advances in Neural Information Processing Systems*, 36:30026–30038, 2023.

[72] Eric Zhan, Stephan Zheng, Yisong Yue, Long Sha, and Patrick Lucey. Generating multi-agent trajectories using programmatic weak supervision. In *International Conference on Learning Representations*, 2019.

[73] Zihan Zhang, Richard Liu, Rana Hanocka, and Kfir Aberman. TEDi: Temporally-entangled diffusion for long-term motion synthesis. In *ACM SIGGRAPH 2024 Conference Papers*, pages 1–11, 2024.

[74] Tianyang Zhao, Yifei Xu, Mathew Monfort, Wongun Choi, Chris Baker, Yibiao Zhao, Yizhou Wang, and Ying Nian Wu. Multi-agent tensor fusion for contextual trajectory prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12126–12134, 2019.

[75] Guangcong Zheng, Xianpan Zhou, Xuewei Li, Zhongang Qi, Ying Shan, and Xi Li. LayoutDiffusion: Controllable diffusion model for layout-to-image generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 22490–22499, 2023.

[76] Zikang Zhou, Luyao Ye, Jianping Wang, Kui Wu, and Kejie Lu. HiVT: Hierarchical vector transformer for multi-agent motion prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8823–8833, 2022.

[77] Zikang Zhou, Jianping Wang, Yung-Hui Li, and Yu-Kai Huang. Query-centric trajectory prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 17863–17873, 2023.

SRA: Spatial Reasoning Adapter via Evolving Social Interaction Graphs for Trajectory Prediction

Supplementary Material

This supplementary material accompanies the main paper. It provides the notation used throughout, full implementation details of SRA in each host, additional ablations and analyses, and more qualitative results. Sections labeled with letters (Supp. A–I) are in this document; numbered sections refer to the main paper.

Contents

A Notation 17
B Implementation Details 17
C Additional Ablations 23
D Joint Metrics 24
E Comparison with Future-Aware Interaction Modules 24
F Robustness Across Scenes 25
G Inference Latency 25
H Limitations 26
I More Qualitative Results 28

A Notation

Table 5: **Notation** used in Sec. 3 of the main paper.

Symbol	Meaning
N, T_p, T_f	number of agents; observed and predicted horizon lengths
$X, x_{i,t}$	observed trajectories; position of agent i at past step t (t=0: last observation)
M (= K=20)	number of sampled future modes
τ, τ^k	noise level / reverse step (τ^K: pure noise, τ^0=0: clean)
y^τ	noisy future sample at level τ
$\hat{y}^0, \hat{y}^\tau$	host's current clean-future estimate at reverse step τ
$\tilde{y}_i^\tau$	estimate after SRA's gated residual correction (Eq. 1)
$\delta_i^\tau \in \mathbb{R}^{T_f \times 2}$	SRA residual for agent i
g_i	learned gate on the residual
$h_i^\tau \in \mathbb{R}^D$	host-provided agent feature (node feature of the graph)
$\sigma_{i,t}^\tau, \mathbf{s}^\tau$	predicted uncertainty of agent i at future step t; its log-variance
$\bar{\sigma}_i^\tau$	temporal mean of $\sigma_{i,t}^\tau$ (Sec. 3.5)
$\mathcal{G} = (\mathcal{V}, \mathcal{E}), \mathcal{N}(i)$	directed interaction graph; selected neighbors of target i (edges $j \to i$)
K_{nbr}	number of selected neighbors per agent (= 5)
u_i^τ	node feature used by the retrieval score (Eq. 4)
$r_{ij,t}^\tau, \tilde{r}_{ij,t}^0$	relative position of j w.r.t. i; the same in i's ego-centric frame
$\theta_{i,t}^0, \Delta\theta_{ij,t}^0$	heading of i; heading difference between j and i
R_i	rotation aligning i's heading at t=0 with the x-axis
$\hat{e}_{ij,t}^\tau$	per-timestep future relational feature of edge $j \to i$ (Eq. 7)
$p_{ij,t}^\tau, e_{ij}^\tau$	temporally mixed edge feature; uncertainty-pooled edge feature (Eq. 8)
$w_{ij,t}^\tau$	temporal pooling weight biased by $\Delta\sigma_{ij,t}^\tau = \sigma_{i,t}^\tau - \sigma_{j,t}^\tau$
w_{ij}^τ, ζ, b_w	per-edge uncertainty gate; its temperature (4) and offset (0.5) (Eq. 10)
$\alpha_{ij}^\tau, m_i^\tau$	attention weight; aggregated message (Eq. 11)
ψ^τ	noise-level embedding
L	number of message-passing layers
λ_σ	weight of the NLL loss $\mathcal{L}_\sigma$ (10^{-2})

B Implementation Details

B.1 Graph Integration: Output-Level vs. Embedding-Level

Although SRA's graph module exposes the same interface ($\hat{y}^\tau$, x, τ, σ) across all three host denoisers, the *point of injection* differs between MID/LED (output-level) and MoFlow (embedding-level). We explain the architectural motivation for each choice below.

Output-level injection in MID and LED. MID's vanilla transformer denoiser produces a noise estimate ε_θ directly from a sequence of past-context tokens; it has no intermediate layer that forms explicit agent representations of *future* trajectory state. LED's core denoiser is similarly structured, and its weights are *frozen* during SRA fine-tuning, which makes internal surgery impractical. Because neither architecture exposes a natural "future agent embedding" between encoder and decoder, we inject SRA's correction at the only available boundary: the final output.

Concretely, SRA maps the host's current $\hat{y}^\tau$ through the graph module to a residual $\delta^\tau \in \mathbb{R}^{A \times K \times T_f \times 2}$, then modifies the noise prediction directly:

$$\varepsilon'_\theta = \varepsilon_\theta + g_i \, \delta_i^\tau, \tag{13}$$

where $g_i \in [0, 1]$ is the learned scalar gate of Eq. 1. Because the correction operates in ε-space, the subsequent DDPM reverse step naturally translates it into a geometric shift on the predicted clean trajectory, with the correct noise-schedule scaling built in.

Embedding-level injection in MoFlow. MoFlow's Motion-Transformer-style decoder factorizes attention across trajectory modes (K-attention) and across agents (A-attention) in two sequential blocks. This provides a natural intermediate representation: after K-attention has already mixed information across modes, but before A-attention fuses inter-agent context, the node embedding matrix $\mathbf{Y}_{\text{emb}} \in \mathbb{R}^{B \times K \times A \times D}$ captures per-agent, per-mode state. Injecting here is strictly more expressive than output-level: the graph-refined embeddings then propagate through A-attention (whose inter-agent queries and keys depend on them), the context-fusion MLP, the motion decoder, and the regression head, giving the full decoder capacity to interpret the graph signal rather than bypassing the decoder with a simple additive correction.

The injection is therefore:

$$\mathbf{Y}_{\text{emb}} \leftarrow \text{FutureGraph}(\mathbf{Y}_{\text{emb}},\ \hat{y}^{\tau}_{\text{abs}},\ \mathbf{t}_{\text{emb}},\ \tau,\ \sigma)\,, \tag{14}$$

after which $\mathbf{Y}_{\text{emb}}$ continues through the remainder of the decoder unchanged. Since MoFlow already predicts per-agent log-variance through its native uncertainty head, σ is a first-class signal at this level, further motivating the tighter coupling.

Summary. The injection level is dictated by architecture, not by choice: MID and LED offer no intermediate future-agent embedding, so output-level is the natural and only option; MoFlow's MTR decoder structure creates an intermediate layer where the graph signal can synergize with the decoder's own agent-attention, making embedding-level injection both natural and strictly more expressive.

Zero initialization. In all three hosts, SRA's output projection layer is initialized to zero weights and zero bias. This guarantees that at the start of fine-tuning, the module is a strict no-op: the host model's learned dynamics are preserved and the graph contribution is phased in gradually.

First-step graph skip. At the very first denoising step there is no prior estimate $\hat{y}^{\tau}$ available from which to build interaction graph geometry. The graph is skipped (`skip_graph=True`) for this step, and the host denoiser runs without relational correction. From the second step onward, the previous step's $\hat{y}^{\tau}$ provides graph geometry.

Normalization for graph edges. Graph edge features are computed in absolute court/field coordinates (meters), not in the normalized training space. For MoFlow, this requires inverting the min-max normalization before passing $\hat{y}^{\tau}$ to the graph module. For MID and LED, the scale factor (TRAJ_SCALE $= 5$) and last observed position are added back before the graph receives the trajectory.

B.2 From refined embedding to per-timestep residual: the decoder

The aggregation step in Section 3.5 produces a refined per-agent embedding $h_i^{\tau,(L)} \in \mathbb{R}^D$ for each agent (and each predicted mode, in multi-modal hosts). This embedding has no temporal axis: the per-timestep dynamics of the relational interaction were already absorbed into the edge features through attention pooling (Eq. 8), and the message-passing aggregator operates strictly at the agent level. The decoder ϕ_θ^{dec} is the component that re-introduces time, mapping the per-agent vector back to a $T_f \times 2$ trajectory residual δ_i^{τ}.

Two design steps make this work cleanly across hosts.

Embedding-level gated residual. Before any host-specific decoding, the graph module emits a residual on the embedding itself, not on the trajectory. Let $h_i^{\tau,(0)}$ be the original (host-provided) per-agent embedding fed into the GNN, and $h_i^{\tau,(L)}$ the embedding after L message-passing layers. We compute the scalar sigmoid gate $g_i \in [0,1]$ of Eq. 1 from the host feature and the pooled graph message $\bar{m}_i$, and form the refined embedding

$$\tilde{h}_i^{\tau} = h_i^{\tau,(0)} + g_i\, W_{\text{out}}\, h_i^{\tau,(L)}, \quad g_i = \text{sigmoid}\Big(W_g\,[\,h_i^{\tau,(0)};\,\bar{m}_i\,]\Big)\,, \tag{15}$$

where $W_g \in \mathbb{R}^{1\times 2D}$ and $W_{\text{out}} \in \mathbb{R}^{D\times D}$. Because W_{out} is zero-initialized, $\tilde{h}_i^{\tau} \equiv h_i^{\tau,(0)}$ at the start of training, and the graph contribution is phased in gradually.

Host-specific time decoder. The refined embedding $\tilde{h}_i^{\tau}$ is then mapped to the per-timestep residual $\delta_i^{\tau} \in \mathbb{R}^{T_f\times 2}$. The decoder differs by host:

- **LED and MID** (output-level injection) use a small dedicated MLP that expands the agent-level embedding into a flattened trajectory residual, then reshapes:

$$\delta_i^{\tau} = \text{reshape}_{T_f\times 2}\Big(\phi_\theta^{\text{dec}}(\tilde{h}_i^{\tau})\Big), \quad \phi_\theta^{\text{dec}} : \mathbb{R}^D \to \mathbb{R}^{T_f\cdot 2}, \tag{16}$$

  where ϕ_θ^{dec} is a two-layer MLP: $\text{Linear}(D\to D)\to\text{ReLU}\to\text{Linear}(D\to T_f\cdot 2)$. The final linear layer is zero-initialized so the residual starts at zero.

- **MoFlow** (embedding-level injection) does *not* have a separate trajectory decoder inside the graph module: $\tilde{h}_i^\tau$ *replaces* the host's intermediate embedding and is fed back through the remaining MTR decoder layers (A-attention, context-fusion MLP, motion decoder, regression head). The host's own regression head, which already maps a D-dim node embedding to $T_f \cdot 2$ trajectory coordinates, plays the role of ϕ_θ^{dec}. No additional decoder is added.

Why no temporal transformer. A more elaborate alternative would have been a temporal transformer over node features after the GNN, mirroring the edge-side Ψ^{edge}. We tried this but found no measurable improvement: the relational temporal structure is already captured inside the per-edge feature e_{ij}^τ via attention pooling, and a second temporal transformer on per-agent features mostly added latency without changing accuracy. We accordingly use the simpler decoders above. This is also consistent with the host's existing architecture: MID's transformer denoiser, LED's frozen core, and MoFlow's MTR decoder all already produce per-timestep outputs from per-agent embeddings, so re-introducing a temporal mixer inside SRA would be redundant.

B.3 Training and Inference Algorithms

The following algorithms describe how SRA integrates into the training and inference loops of each host denoiser. We use $\hat{y}^\tau$ for the current clean-trajectory estimate, σ for per-agent uncertainty (log-variance), g for the learnable gate, and δ for the graph residual. Notation follows Section 3 of the main paper.

MID (Diffusion DDPM)

Algorithm 1 SRA + MID: Training (one mini-batch iteration)

Require: Past trajectories x, ground-truth future y_0; host transformer θ, graph module G.

Compute 6-channel past features $p = [x_{\text{abs}},\ x_{\text{rel}},\ x_{\text{vel}}]$
Encode past context: $c = \text{Encoder}_\theta(p,\ \text{mask})$
Sample diffusion level $\tau \sim \mathcal{U}\{1, \ldots, T\}$, noise $\varepsilon \sim \mathcal{N}(0, I)$
Forward diffuse: $y_\tau = \sqrt{\bar{\alpha}_\tau} \cdot y_0 + \sqrt{1 - \bar{\alpha}_\tau} \cdot \varepsilon$
Pass 1 {no gradient; build graph geometry from GT future (teacher forcing)}
  $\hat{y}_{\text{abs}}^{(1)} \leftarrow y_0 \cdot \text{TRAJ_SCALE} + \text{last_obs}$ {absolute court coords}
  $\varepsilon^{(1)},\ s \leftarrow \text{GraphDenoiserNet}_\theta(y_\tau,\ \beta_\tau,\ c;\ \hat{y}_{\text{abs}}^{(1)},\ \sigma{=}\text{None})$
  $\sigma \leftarrow \exp(s/2)$ {per-agent uncertainty from log-variance head}
Pass 2 {with gradient; use Pass 1 estimate for graph geometry}
  $\hat{y}^{(2)} \leftarrow \big(y_\tau - \sqrt{1 - \bar{\alpha}_\tau} \cdot \varepsilon^{(1)}\big) / \sqrt{\bar{\alpha}_\tau}$ {recover $\hat{y}_0$ from Pass 1}
  $\hat{y}_{\text{abs}}^{(2)} \leftarrow \hat{y}^{(2)} \cdot \text{TRAJ_SCALE} + \text{last_obs}$
  $\varepsilon^{(2)},\ s \leftarrow \text{GraphDenoiserNet}_\theta(y_\tau,\ \beta_\tau,\ c;\ \hat{y}_{\text{abs}}^{(2)},\ \sigma)$
  (Inside net: transformer encodes y_τ; graph G maps $\hat{y}_{\text{abs}}^{(2)}$ to δ;
    output-level: $\varepsilon^{(2)} = \varepsilon_{\text{base}}^{(2)} + g_i\,\delta$)
$\mathcal{L}_{\text{diff}} = \|\varepsilon^{(2)} - \varepsilon\|^2$ {host MSE loss}
$\mathcal{L}_\sigma = \frac{1}{2}\big(s + \|\varepsilon - \varepsilon_{\text{base}}^{(2)}\|^2 e^{-s}\big)$ {heteroscedastic NLL}
$\mathcal{L} = \mathcal{L}_{\text{diff}} + \lambda_\sigma\,\mathcal{L}_\sigma$; backpropagate, update θ and G

Algorithm 2 SRA + MID: Inference (one scene, K samples)

Require: Past trajectories x; K samples, 10 DDIM steps (stride 10 over T=100).

Compute p, encode $c = \mathrm{Encoder}(p, \mathrm{mask})$
Initialize K noisy futures: $y_T \sim \mathcal{N}(0, I)^{K \times A \times T_f \times 2}$; $\hat{y}_{\mathrm{abs}}^{\mathrm{prev}} \leftarrow$ `None`
for $\tau = T,\ T{-}10,\ \ldots,\ 10$ **do**
  Pass 1 (no grad):
    $\varepsilon^{(1)},\ s \leftarrow \mathrm{GraphDenoiserNet}(y_\tau, \beta_\tau, c;\ \hat{y}_{\mathrm{abs}}^{\mathrm{prev}},\ \sigma{=}\mathrm{None},\ \texttt{skip} = (\hat{y}_{\mathrm{abs}}^{\mathrm{prev}}{=}\texttt{None}))$
    $\sigma \leftarrow \exp(s/2)$
  Pass 2 (no grad):
    $\hat{y}^{(2)} \leftarrow (y_\tau - \sqrt{1-\bar{\alpha}_\tau} \cdot \varepsilon^{(1)})/\sqrt{\bar{\alpha}_\tau}$
    $\hat{y}_{\mathrm{abs}}^{(2)} \leftarrow \hat{y}^{(2)} \cdot \mathrm{TRAJ_SCALE} + \mathrm{last_obs}$
    $\varepsilon^{(2)},\ _ \leftarrow \mathrm{GraphDenoiserNet}(y_\tau, \beta_\tau, c;\ \hat{y}_{\mathrm{abs}}^{(2)},\ \sigma)$
    (output-level injection: $\varepsilon^{(2)} = \varepsilon_{\mathrm{base}}^{(2)} + g_i\,\delta$*)*
  $\hat{y}_0^{(2)} \leftarrow (y_\tau - \sqrt{1-\bar{\alpha}_\tau} \cdot \varepsilon^{(2)})/\sqrt{\bar{\alpha}_\tau}$
  $\hat{y}_{\mathrm{abs}}^{\mathrm{prev}} \leftarrow \hat{y}_0^{(2)} \cdot \mathrm{TRAJ_SCALE} + \mathrm{last_obs}$ {update for next step}
  DDIM step: $y_{\tau-10} = \sqrt{\bar{\alpha}_{\tau-10}} \cdot \hat{y}_0^{(2)} + \sqrt{1-\bar{\alpha}_{\tau-10}} \cdot \varepsilon^{(2)}$
end for
return Unnormalize $\hat{y}_0^{(2)}$ to absolute court coordinates; return K predicted futures

LED (Leapfrog DDPM)

Algorithm 3 SRA + LED: Training (one mini-batch iteration)

Require: Past trajectories x, GT future y_0; **frozen** core denoiser Φ, leapfrog initializer ψ, graph module G.

Compute 6-channel past p, social mask
Run initializer: $(\hat{s}_{\mathrm{pred}},\ \mu,\ \log\sigma^2) \leftarrow \mathrm{LEDInit}_\psi(p, \mathrm{mask})$
Scale samples: $\hat{s}_{\mathrm{pred}} \leftarrow e^{\log\sigma^2/2} \cdot \hat{s}_{\mathrm{pred}}\ /\ \mathrm{std}(\hat{s}_{\mathrm{pred}})$
Initialize K=20 modes: $\mathrm{loc} = \hat{s}_{\mathrm{pred}} + \mu\,[\mathrm{None}]$; $\sigma = \log\sigma^2$
for each half $h \in \{\mathrm{loc}_{:,:10},\ \mathrm{loc}_{:,10:}\}$ **do**
  $y_{\mathrm{cur}} \leftarrow h$
  for $\tau = 4, 3, 2, 1, 0$ **do**
    $\varepsilon_\theta \leftarrow \Phi(y_{\mathrm{cur}},\ \beta_\tau,\ p,\ \mathrm{mask})$ {FROZEN core denoiser}
    $\hat{y}^\tau \leftarrow (y_{\mathrm{cur}} - \sqrt{1-\bar{\alpha}_\tau} \cdot \varepsilon_\theta)/\sqrt{\bar{\alpha}_\tau}$
    $\delta \leftarrow G(\hat{y}^\tau,\ x,\ \tau,\ \sigma)$
    $\varepsilon'_\theta \leftarrow \varepsilon_\theta - \frac{\sqrt{\bar{\alpha}_\tau}}{\sqrt{1-\bar{\alpha}_\tau}} \cdot \delta$ {output-level; delta in $\hat{y}_0$ space}
    $y_{\mathrm{cur}} \leftarrow \frac{1}{\sqrt{\alpha_\tau}}\big(y_{\mathrm{cur}} - \frac{1-\alpha_\tau}{\sqrt{1-\bar{\alpha}_\tau}}\varepsilon'_\theta\big) + \sqrt{\beta_\tau} \cdot \xi \cdot \epsilon_0$ $\{\xi \ll 1\}$
  end for
end for
$\hat{y}_{\mathrm{pred}} = \mathrm{concat}(y_{\mathrm{cur},0},\ y_{\mathrm{cur},1})$
$\mathcal{L}_{\mathrm{init}} =$ trajectory regression loss$(\hat{y}_{\mathrm{pred}},\ y_0)$
$\mathcal{L}_\sigma = \frac{1}{2}(\log\sigma^2 + \|y_0 - \mu\|^2 e^{-\log\sigma^2})$
$\mathcal{L} = \mathcal{L}_{\mathrm{init}} + \lambda_\sigma\,\mathcal{L}_\sigma$; backpropagate through ψ and G only (core denoiser Φ frozen)

Algorithm 4 SRA + LED: Inference (one scene)

Require: Past trajectories x; K=20 modes, 5 leapfrog steps.

Compute p, mask
$(\hat{s}_{\text{pred}},\ \mu,\ \log\sigma^2) \leftarrow \text{LEDInit}(p, \text{mask})$; scale, form loc, σ
for each half $h \in \{\text{loc}_{:,:10},\ \text{loc}_{:,10:}\}$ **do**
  $y_{\text{cur}} \leftarrow h$
  for $\tau = 4, 3, 2, 1, 0$ **do**
    $\varepsilon_\theta \leftarrow \Phi(y_{\text{cur}},\ \beta_\tau,\ p,\ \text{mask})$
    $\hat{y}^\tau \leftarrow (y_{\text{cur}} - \sqrt{1-\bar{\alpha}_\tau}\cdot\varepsilon_\theta)/\sqrt{\bar{\alpha}_\tau}$
    $\delta \leftarrow G(\hat{y}^\tau,\ x,\ \tau,\ \sigma)$
    $\varepsilon'_\theta \leftarrow \varepsilon_\theta - \frac{\sqrt{\bar{\alpha}_\tau}}{\sqrt{1-\bar{\alpha}_\tau}}\cdot\delta$ {output-level injection}
    $y_{\text{cur}} \leftarrow \frac{1}{\sqrt{\alpha_\tau}}\left(y_{\text{cur}} - \frac{1-\alpha_\tau}{\sqrt{1-\bar{\alpha}_\tau}}\varepsilon'_\theta\right) + \sqrt{\beta_\tau}\cdot\xi\cdot\epsilon_0$
  end for
end for
return $\text{concat}(y_{\text{cur},0},\ y_{\text{cur},1})\cdot\text{TRAJ_SCALE} + \text{last_obs}$ (K=20 absolute futures)

MoFlow (Flow Matching)

Algorithm 5 SRA + MoFlow: Training (one mini-batch iteration)

Require: Past trajectories x, GT future y_0 (min-max normalized to $[-1, 1]$); MTR encoder E, motion transformer M (includes graph G and logvar head).

Encode past context: $\mathbf{x}_{\text{data}} = E(x)$
Sample time: $t \sim \text{LogitNormal}(-0.5,\ 1.5)$
Sample noise: $\varepsilon \sim \mathcal{N}(0, I)$
Linear interpolation: $y_t = (1-t)\cdot\varepsilon + t\cdot y_0$ {flow-matching trajectory}
Pass 1 {no gradient; teacher-forcing with y_0}
  $y^{\text{GT}}_{0,\text{abs}} \leftarrow \text{unnorm}(y_0) + \text{last_obs}$ {absolute coords for graph edges}
  $(\hat{y}^{(1)},\ _,\ \ell^{(1)}) \leftarrow M.\text{forward}(y_t,\ t,\ \mathbf{x}_{\text{data}};\ y^{\text{GT}}_{0,\text{abs}},\ \sigma{=}\text{None})$
  (Inside M: K-attn $\rightarrow$ [***embedding-level:*** $\mathbf{Y}_{\text{emb}} \leftarrow G(\mathbf{Y}_{\text{emb}},\ y^{\text{GT}}_{0,\text{abs}}, \ldots)$ *] $\rightarrow$ A-attn $\rightarrow$ decoder $\rightarrow \hat{y}^{(1)}$)*
  $\sigma \leftarrow \left(\ell^{(1)}\right)^{1/2}$ {per-agent uncertainty from Pass 1 logvar}
Pass 2 {with gradient; use $\hat{y}^{(1)}$ for graph geometry}
  $\hat{y}^{(1)}_{\text{abs}} \leftarrow \text{unnorm}(\hat{y}^{(1)}) + \text{last_obs}$
  $(v_{\text{pred}},\ \hat{c},\ \ell^{(2)}) \leftarrow M.\text{forward}(y_t,\ t,\ \mathbf{x}_{\text{data}};\ \hat{y}^{(1)}_{\text{abs}},\ \sigma)$
  (Inside M: K-attn $\rightarrow$ [***embedding-level:*** $\mathbf{Y}_{\text{emb}} \leftarrow G(\mathbf{Y}_{\text{emb}},\ \hat{y}^{(1)}_{\text{abs}},\ \sigma, \ldots)$ *] $\rightarrow$ A-attn $\rightarrow$ decoder $\rightarrow v_{\text{pred}}$)*
Target velocity: $v^* = (y_0 - y_t)/(1-t)$
$\mathcal{L}_{\text{reg}} = \|v_{\text{pred}} - v^*\|^2$ {flow-matching regression}
$\mathcal{L}_{\text{cls}} = $ mode classification loss$(\hat{c},\ y_0)$
$\mathcal{L}_\sigma = \text{NLL}(\ell^{(2)},\ y_0)$ {MoFlow native uncertainty loss}
$\mathcal{L} = \mathcal{L}_{\text{reg}} + \mathcal{L}_{\text{cls}} + 0.01\cdot\mathcal{L}_\sigma$; backpropagate, update E and M

Algorithm 6 SRA + MoFlow: Inference (one scene, K samples)

Require: Past trajectories x; K=20 samples, 10 Euler steps (dt=0.1).

Encode past: $\mathbf{x}_{\text{data}} = E(x)$
Initialize: $y_T \sim \mathcal{N}(0, I)^{B\times K\times A\times T_f\times 2}$ {tied noise across K modes}; $\hat{y}_{\text{abs}}^{\text{prev}} \leftarrow$ `None`
for $s = 0, 1, \ldots, 9$ (current time $t_s = s/10$) **do**
  Apply input scaling: $y_{t_s}^{\text{in}} \leftarrow y_{t_s} \cdot \text{scale}(t_s)$
  Pass 1 (no grad):
    $(\hat{y}^{(1)},\ _,\ \ell^{(1)}) \leftarrow M.\text{forward}(y_{t_s}^{\text{in}},\ t_s,\ \mathbf{x}_{\text{data}};\ \hat{y}_{\text{abs}}^{\text{prev}},\ \sigma{=}\text{None},\ \texttt{skip}{=}(\hat{y}_{\text{abs}}^{\text{prev}}{=}\texttt{None}))$
    (embedding-level: G applied between K-attn and A-attn inside M)
    $\sigma \leftarrow \left(\ell^{(1)}\right)^{1/2}$; $\hat{y}_{\text{abs}}^{(1)} \leftarrow \text{unnorm}(\hat{y}^{(1)}) + \text{last_obs}$
  Pass 2 (no grad):
    $(v_{\text{pred}},\ _,\ _) \leftarrow M.\text{forward}(y_{t_s}^{\text{in}},\ t_s,\ \mathbf{x}_{\text{data}};\ \hat{y}_{\text{abs}}^{(1)},\ \sigma)$
    (embedding-level with σ: G refines $\mathbf{Y}_{\text{emb}}$, then A-attn, decoder)
  $\hat{y}_0 \leftarrow v_{\text{pred}}$ {model directly predicts clean y_0}
  $\hat{y}_{\text{abs}}^{\text{prev}} \leftarrow \text{unnorm}(\hat{y}_0) + \text{last_obs}$ {update for next step}
  Euler update: $y_{t_{s+1}} \leftarrow y_{t_s} + v_{\text{pred}} \cdot \text{d}t$
end for
return $\text{unnorm}(\hat{y}_0) + \text{last_obs}$ (K=20 absolute future trajectories)

C Additional Ablations

C.1 Number of neighbors K_{nbr}

Table 6 sweeps the top-K_{nbr} neighborhood size used in the sparse selection step (Sec. 3.3) over $\{3, 5, 10\}$ on the NBA dataset, for all three host denoisers. The pattern is consistent: $K_{\mathrm{nbr}}{=}5$ is the best or near-best setting in every case, with $K_{\mathrm{nbr}}{=}3$ trailing slightly and $K_{\mathrm{nbr}}{=}10$ either flat (MID, MoFlow) or actively destabilizing (LED, where FDE jumps from 1.161 at $K_{\mathrm{nbr}}{=}5$ to 2.303 at $K_{\mathrm{nbr}}{=}10$).

Model	K_{nbr}	ADE	FDE
MID + SRA	3	0.960	1.346
	5	0.957	1.346
	10	0.968	1.356
LED + SRA	3	0.775	1.163
	5	0.778	1.161
	10	0.783	2.303
MoFlow + SRA	3	0.703	0.893
	5	0.695	0.870
	10	0.704	0.878

Table 6: Ablation on the number of selected neighbors K_{nbr} on NBA (ADE / FDE, best of 20). $K_{\mathrm{nbr}}{=}5$ is the default.

This validates a key design intuition: the optimal neighborhood size is governed by *interaction density*, not by the number of agents in the scene. NBA has 11 agents, but each player effectively interacts with only a handful of teammates and nearby defenders, so $K_{\mathrm{nbr}}{=}5$ already covers the relevant peer set. Larger K_{nbr} admits weakly-related agents whose noisy edge features dilute the aggregated message; only considering the core interactions results in more accurate predictions. We therefore fix $K_{\mathrm{nbr}}{=}5$ as the default in all main experiments.

D Joint Metrics

ADE and FDE take the minimum over modes independently for each agent, so they do not reflect whether the predicted agents are consistent with one another. Joint ADE and Joint FDE (JADE/JFDE) [62] instead select the best mode jointly over all agents in the scene, and therefore improve only when the predicted agents are mutually consistent. We do not use collision rate: physical contact between players is a normal part of basketball, soccer, and football, so a collision counter would penalize predictions that are in fact correct.

Table 7: **Joint metrics** JADE / JFDE [62] on the sports benchmarks, which select one mode jointly for all agents in the scene. Better of each host pair in **bold**.

	NBA		Soccer		Football	
Method	JADE	JFDE	JADE	JFDE	JADE	JFDE
MID	1.977	3.706	0.789	1.577	**0.581**	**1.184**
MID + SRA	**1.945**	**3.597**	**0.755**	**1.518**	0.603	1.238
LED	1.609	2.851	0.601	1.121	0.550	1.071
LED + SRA	**1.580**	**2.842**	**0.575**	**1.055**	**0.501**	**0.969**
MoFlow	1.476	2.708	0.654	1.067	0.509	0.991
MoFlow + SRA	**1.454**	**2.675**	**0.636**	**1.037**	**0.503**	**0.976**

SRA improves JADE and JFDE (Table 7) for every host on NBA and Soccer, by up to 5.9%, and for LED and MoFlow on Football, by up to 9.5%. Because these metrics are computed on a single jointly selected mode, the improvement indicates that the corrections SRA applies are mutually consistent across agents rather than accurate only in isolation. MID on Football is the exception: its marginal ADE/FDE improve, but its joint metrics degrade (consistent over three evaluation runs), so on this pair SRA sharpens per-agent accuracy without improving single-sample joint coherence.

E Comparison with Future-Aware Interaction Modules

Of the prior methods that use predicted futures [20, 14, 23], MoFlow is already one of our hosts, and SRA improves it on all four benchmarks (Table 1).

Adapter-level comparison. GameFormer [20] is the closest at the mechanism level, since its level-k reasoning also conditions each agent on the predicted futures of the others. Because SRA is an adapter, we compare at the level of adapters: we implement level-k reasoning as an adapter that receives the same tensor at the same insertion point, returns a gated residual on the same embedding, and follows the same training and evaluation protocol (NBA, minADE_{20} / minFDE_{20} at 4.0 s), so that the comparison isolates *how* the predicted future is used.

Table 8: **Adapter-level comparison** on NBA (ADE / FDE). GameFormer's level-k reasoning [20] is placed in the same slot as SRA, with the same input, injection point, and training protocol.

Host	Baseline	+ level-k [20]	+ SRA (ours)
MID (diffusion)	1.037 / 1.472	0.984 / 1.394	**0.957 / 1.346**
LED (leapfrog)	0.839 / 1.204	0.836 / 1.266	**0.778 / 1.161**
MoFlow (flow matching)	0.703 / 0.884	0.949 / 1.098	**0.695 / 0.870**
Module parameters	—	1.30 M	**0.63 M**

As Table 8 shows, level-k reasoning is inconsistent as an adapter: it improves MID, is mixed on LED, and substantially degrades MoFlow. It reads the host's current estimate with no measure of how reliable that estimate is, the same condition that our cumulative ablation isolates (Table 2), where ungated future-conditioned interaction helps some hosts and hurts others. SRA weights each message by the relative uncertainty of its two endpoints and improves all three hosts with about half the parameters.

Standalone comparison. The coarse-to-fine approach (C2F) [23] refines its prediction along the temporal axis and has no module that consumes other agents' predicted futures, so there is no equivalent slot to swap. We therefore compare against C2F, and against standalone GameFormer, as complete predictors (Table 9). All entries use the same NBA split and the minADE_{20} / minFDE_{20} protocol.

Table 9: **Standalone future-aware predictors vs. hosts with SRA** on NBA (ADE / FDE). Percentages are the relative improvement of SRA over its host.

Category	Method	ADE ↓	FDE ↓
Standalone future-aware	GameFormer [20]	0.848	0.951
	C2F [23]	0.904	1.063
Host + adapter	MID [15]	1.037	1.472
	MID + SRA	0.957 (+7.7%)	1.346 (+8.6%)
	LED [37]	0.839	1.204
	LED + SRA	0.778 (+7.3%)	1.161 (+3.6%)
	MoFlow [14]	0.703	0.884
	MoFlow + SRA	**0.695** (+1.1%)	**0.870** (+1.6%)

A standalone predictor is a fixed artifact: its accuracy is a property of that one architecture. An adapter has no such ceiling, since its gain is realized on whatever host it is attached to and rises with that host. Applied to our strongest host, SRA surpasses both standalone future-aware predictors in ADE and FDE.

F Robustness Across Scenes

A natural concern is that strengthening contextual information could overfit specific scenes. SRA's added context is largely scene-independent: it takes no map, scene, or agent-identity input, and it is derived from the host's own estimate rather than retrieved from anything fitted to the training scenes. The module operates on the relative geometry between agent pairs, which makes it invariant to global translation and equivariant to agent permutation by construction, and it does not observe absolute positions in scene coordinates. What SRA learns is how relative future geometry and confidence between two agents determine the correction one applies to the other, a regularity that holds across scenes rather than within any one of them.

The empirical evidence is consistent with this. (1) SRA improves ADE on all 12 host–dataset combinations and FDE on 11 of 12 (Table 1), including SDD, whose top-down pedestrian motion differs from the sports benchmarks in domain, agent count, and capture modality; the same module transfers unchanged. (2) The gains are distributed rather than concentrated: SRA lowers per-scene ADE on 60–90% of the test scenes for every host (Table 10), so the aggregate is not carried by a favorable subset. (3) Figure 5 shows gains in every uncertainty bin, with the largest gains on the highest-σ scenes.

Table 10: **Percentage of test scenes in which SRA lowers the host's ADE.**

Host	NBA	Soccer	Football	Average
MID	74.7%	81.5%	71.0%	75.7%
LED	87.3%	82.6%	89.7%	86.5%
MoFlow	66.6%	74.0%	59.7%	66.8%

G Inference Latency

We measure the inference cost added by SRA on each host denoiser with an optimized implementation, on a single NVIDIA RTX 3090 GPU with batch size 16 and $K{=}20$ predicted modes on NBA.

The computation added by SRA is structurally small. Rebuilding the graph is a two-term retrieval score over agent pairs (a dot product plus a geometric bias over four summary statistics, with no

Table 11: **Inference latency on NBA** (K=20, batch 16, single RTX 3090). Per-sample time is the per-batch time divided by the batch size; Δ is the overhead added by SRA.

Method	Per-batch (ms)	Per-sample (ms)	Δ per-sample (ms)	Δ %
LED	233.2	14.58	–	–
LED + SRA	256.8	16.05	+1.47	+10.1%
MoFlow	86.0	5.37	–	–
MoFlow + SRA	222.8	13.93	+8.56	+159.4%
MID	803.7	50.23	–	–
MID + SRA	847.4	52.96	+2.73	+5.4%

per-timestep encoding), and the expensive operations, relational encoding and message passing, run only on the $N \cdot K_{\mathrm{nbr}}$ retained edges rather than all $N(N-1)$ pairs, which is what sparse selection exists for. The module has 0.63 M parameters, shared across all agents, edges, and reverse steps.

In absolute terms, SRA adds 1.5 to 8.6 ms per sample. The largest relative figure, $+159\%$ on MoFlow, reflects how little the unmodified host costs rather than how much SRA costs: per-sample latency moves from 5.4 ms to 13.9 ms, which is still faster than either unmodified LED or MID. NBA is sampled at 5 Hz, so a scene must be processed within 200 ms for real-time operation, and the slowest configuration (MID + SRA) uses 53 ms of that budget.

H Limitations

Training requirement. SRA is modular, not training-free. It can be integrated into an existing predictor without replacing or redesigning its generative core, but it requires a host-specific injection point (output-level for MID and LED, embedding-level for MoFlow; Sec. B.1) and joint training or fine-tuning with the host. It cannot be attached to a frozen, pretrained host without any training.

Scope of compatible hosts. SRA assumes that the host (i) refines a continuous future estimate through an iterative process, (ii) exposes a prediction or intermediate representation into which a correction can be injected, and (iii) provides agent features from which an uncertainty head can be trained. We validate it on three hosts that span DDPM, leapfrog-DDPM, and flow matching; regression- and classification-based predictors without an iterative refinement chain are not covered, and extending SRA to them remains future work.

Inference latency overhead. SRA rebuilds a future-conditioned interaction graph at every reverse step, on top of the host's own computation. With the optimized implementation the overhead is 1.5–8.6 ms per scene (Table 11), well inside the 200 ms budget of 5 Hz sports data, but the relative overhead is large for very fast hosts such as MoFlow ($+159\%$), and the all-pairs retrieval score grows as $\mathcal{O}(N^2)$ in the number of agents, which may matter for large crowds.

Dependence on the host denoiser. SRA constructs its graph from the host denoiser's intermediate future estimate, so the quality of the graph depends on the quality of that estimate. If the host moves toward an incorrect mode at an early generation step, or mispredicts an interaction-critical agent, the adapter can build edges from an inaccurate future geometry. Zero-initialized residuals, gated corrections, sparse neighborhoods, uncertainty weighting, and iterative graph recomputation reduce the impact of such errors but do not eliminate them. The method does not provide a formal stability guarantee for the coupled host and adapter dynamics.

Metric coverage. We report marginal (ADE/FDE) and joint (JADE/JFDE) displacement metrics. Joint metrics capture whether the predicted agents are mutually consistent, but they still do not measure map violation, social-rule violation, physical feasibility, or downstream planning cost, and we do not report collision rate because physical contact is a normal part of the sports we evaluate. SRA improves the joint metrics in 8 of 9 sports host–dataset pairs; for MID on Football, joint metrics degrade although marginal metrics improve.

Dataset and scene scope. The experimental evaluation is concentrated on SDD and sports benchmarks. These datasets primarily contain scenes with a single dominant agent class, such as pedestrians or sports players, and the relational encoder uses a shared agent-centric representation. We do not evaluate mixed-class scenes with heterogeneous dynamics, physical footprints, right-of-way rules, or map-dependent interactions. We also do not test robustness to perception noise, occlusion, identity switches, asynchronous sensing, or large-scale crowds.

Broader impact. More coherent multi-agent forecasts can make planners for autonomous vehicles and robots that share space with people safer and less conservative, and can support sports analytics. Conversely, forecasts that look coherent may be over-trusted: a downstream system should not treat SRA's predictions or its uncertainty estimates as guarantees, particularly outside the domains evaluated here.

I More Qualitative Results

Figures 6 and 7 compare predictions with and without SRA on additional NBA scenes. With SRA, the predicted agents are more coherent with one another, because each prediction is refined against the predicted futures of its neighbors. Figure 8 shows LED with SRA on complex NBA scenes: agents in the middle of a crowded scene are assigned high uncertainty, and SRA predicts them more accurately.

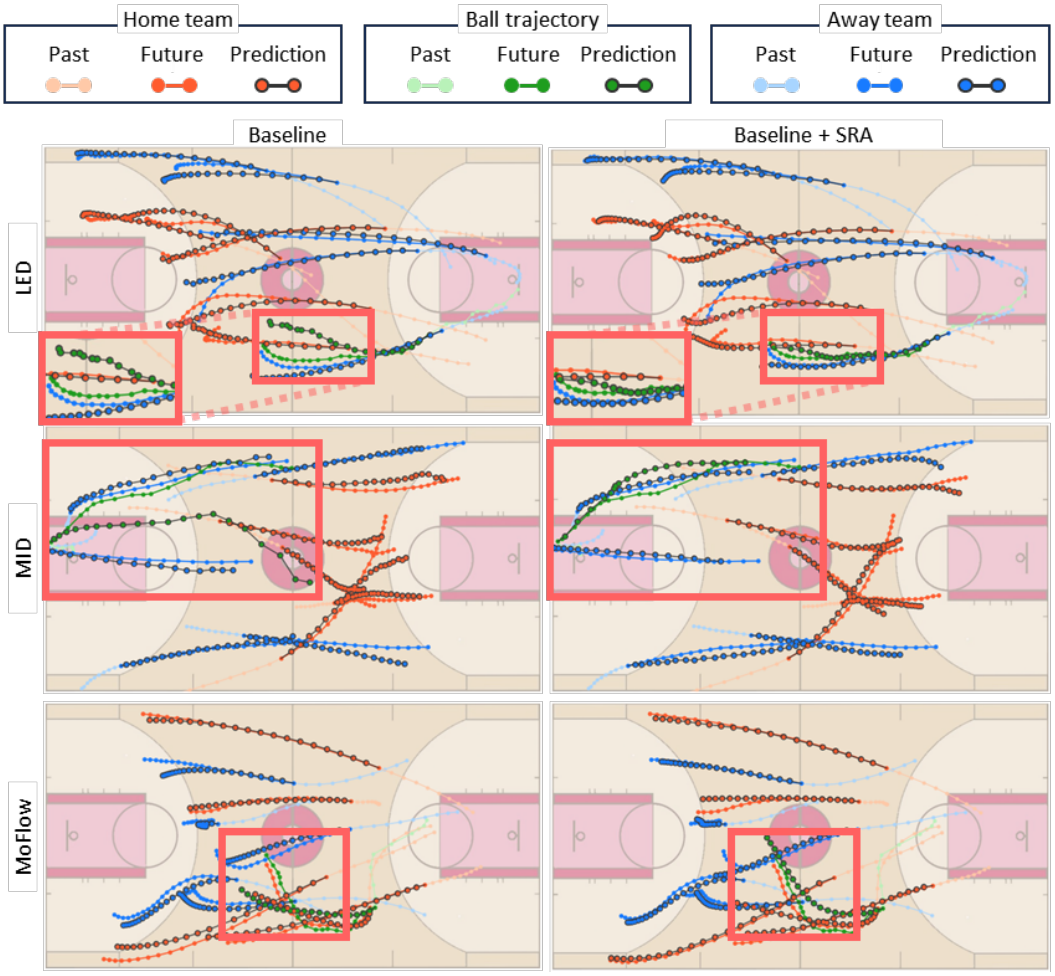

Figure 6: Additional qualitative comparisons on NBA, in the same format as Fig. 3. Rows: hosts (LED, MID, MoFlow); left: host alone, right: host + SRA. Light, medium, and dark dots are the observed past, the ground-truth future, and the prediction (best of $K=20$ modes by ADE); red boxes mark where the predictions differ.

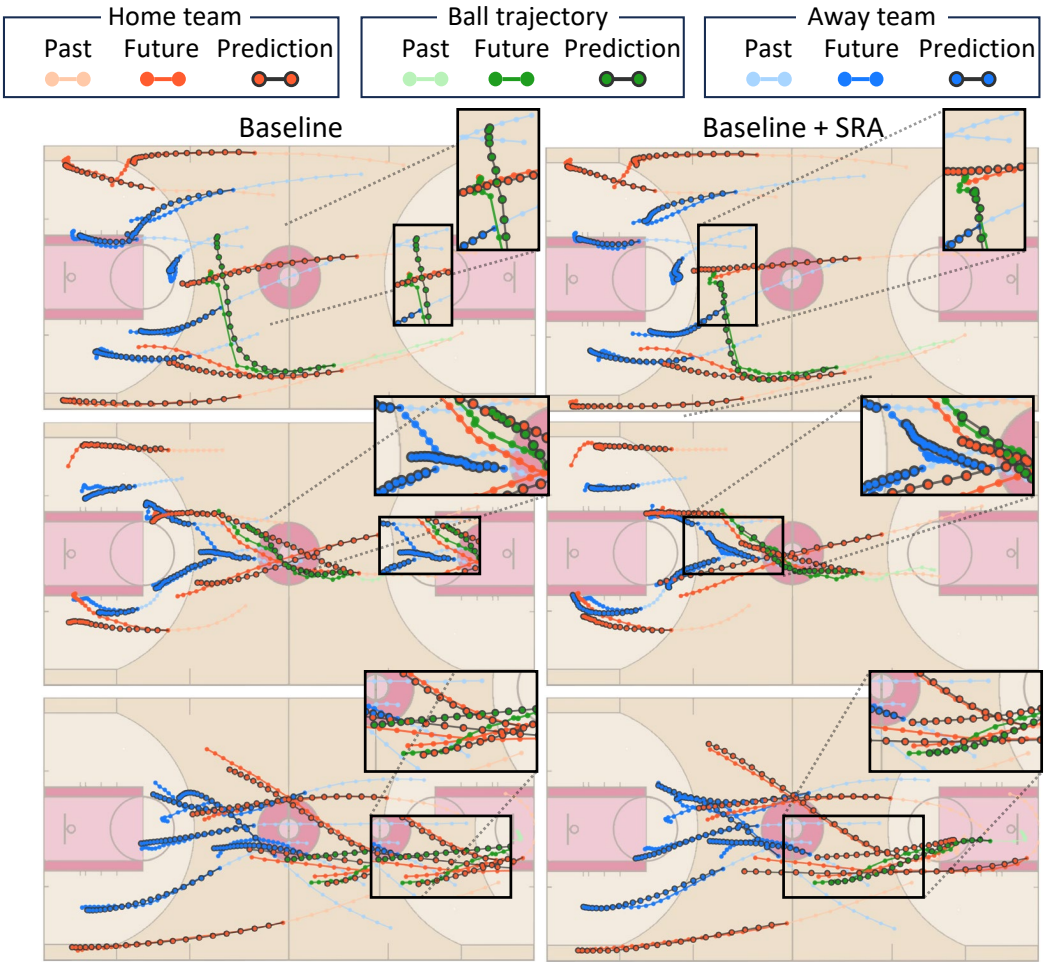

Figure 7: Additional qualitative comparisons on NBA. Rows (top to bottom): LED, MID, and MoFlow as hosts; left: host alone, right: host + SRA. Light, medium, and dark dots are the observed past, the ground-truth future, and the prediction (best of $K=20$ modes by ADE). Black boxes mark where the predictions differ and are enlarged in the insets.

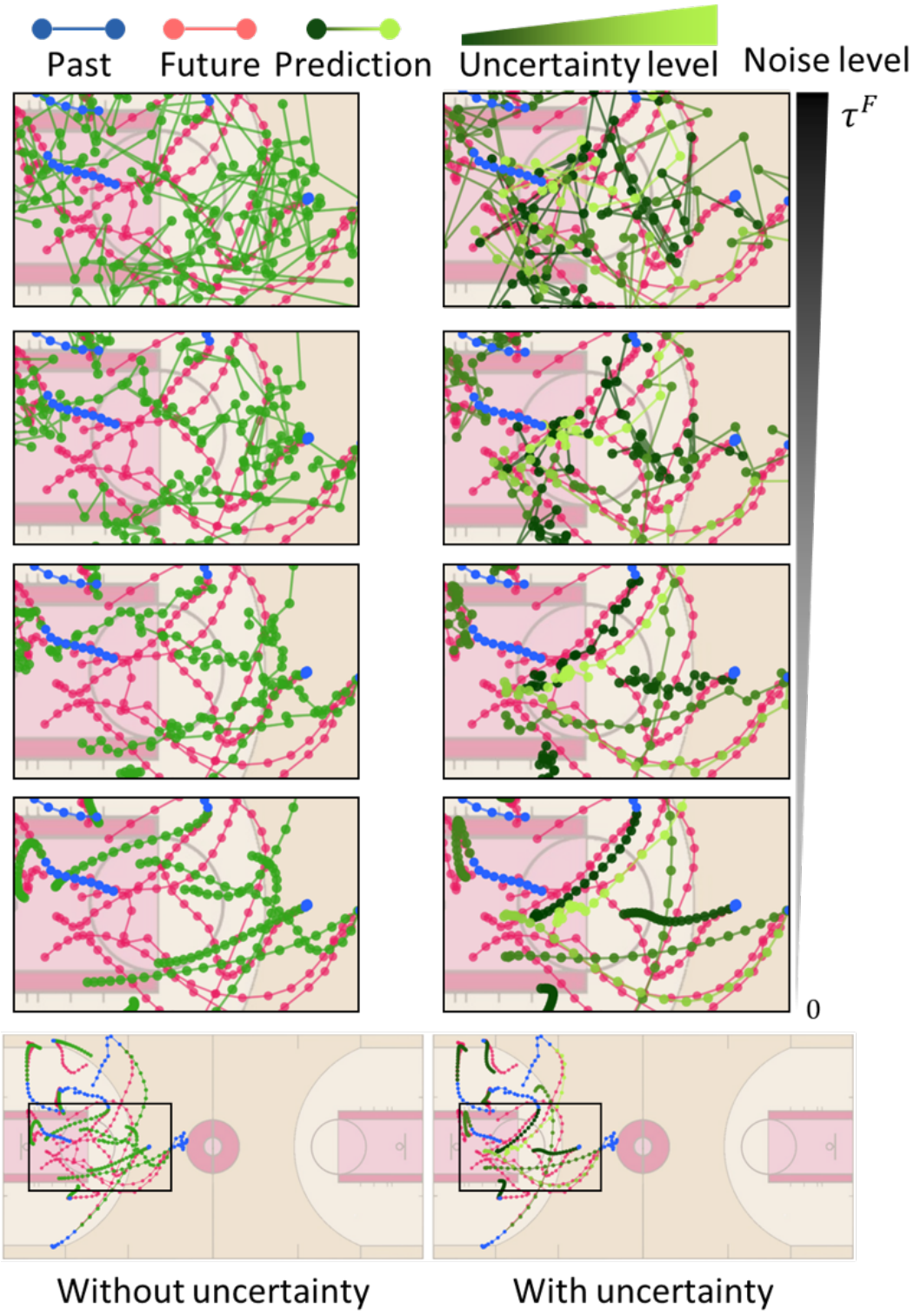

Figure 8: Additional denoising examples without (left) and with (right) uncertainty, LED on NBA, in the same format as Fig. 4. Rows above the full scene (bottom) show the boxed region from high noise (top) to the final step; blue: past, pink: ground-truth future, green: prediction shaded by predicted uncertainty. Agents in the crowded center receive high uncertainty and are predicted more accurately with uncertainty-weighted message passing.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The paper states its main claims in the abstract and introduction, and supports them through the proposed method and empirical results.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Limitations are summarized in Sec. 6 and discussed in detail in Supp. H (training requirement, scope of compatible hosts, latency, dependence on the host estimate, metric coverage, and dataset scope).

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [N/A]

Justification: The paper does not include theoretical results.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: The paper provides the method description, details, datasets, metrics, and evaluation protocol needed to reproduce the main results.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: Code is released at `https://github.com/Jaewoo97/SRA`; all datasets are public.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: The paper specifies the experimental settings, including datasets, training details, evaluation metrics, and implementation choices.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No]

   Justification: The paper does not report error bars.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).

- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: The paper reports the compute resources used for the experiments, including hardware and runtime information.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The research follows the NeurIPS Code of Ethics and does not involve deceptive, harmful, or privacy-invasive data collection.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: The broader-impact paragraph in Supp. H discusses potential positive impacts for safer trajectory forecasting as well as the risk of over-trusting coherent-looking forecasts in safety-critical systems.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.

- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The paper does not release high-risk models or datasets that require special safeguards.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [No]

    Justification: The paper cites the existing datasets and codebases used in the experiments, but does not explicitly list all licenses and terms of use.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.

- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: The only new asset is our code, released at `https://github.com/Jaewoo97/SRA` with documentation for setup, training, and evaluation. The paper introduces no new datasets.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing or research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing or research with human subjects, so IRB approval is not applicable.

    Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: The core method does not use LLMs as an important, original, or non-standard component.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.