

Unified Uncertainty-Aware Diffusion for Multi-Agent Trajectory Modeling –Supplemental–

Guillem Capellera^{1,2} Antonio Rubio² Luis Ferraz² Antonio Agudo¹

¹Institut de Robòtica i Informàtica Industrial, CSIC-UPC ²Kognia Sports Intelligence

{guillem.capellera, antonio.agudo}@upc.edu {antonio.rubio, luis.ferraz}@kogniasports.com

1. Unified Uncertainty-Aware Diffusion

In this section we depict the details of our Unified Uncertainty-aware Diffusion (U2Diff) approach.

1.1. Implementation

U2Diff is trained for 100 epochs on an RTX A6000, with a batch size of 64 for the Basketball-U dataset and 16 for the others. Training time varies between 30 to 120 minutes, depending on the dataset size. For the forecasting task in NBA dataset (see Table 2 of the main paper), individual-based metrics minADE₂₀/minFDE₂₀ are computed using directly DDPM sampling. The remaining other cases are computed using the DDIM, as explained in the main paper.

1.2. Architecture details

The proposed architecture builds upon CSDI framework [55], introducing key modifications tailored to the task of multi-agent trajectory completion. The first modification replaces the original Transformer-based temporal processing module with the Temporal Mamba, specifically designed to model temporal dynamics without the need for explicit positional encodings. This change addresses the sub-optimal performance of temporal positional encodings observed during the denoising process. Additionally, the final linear head of the architecture is redesigned to output both the mean and standard deviation of the predicted noise distribution, enabling uncertainty estimation. The full architecture is depicted in Fig. 1.

Input Embedding As outlined in the main paper, the input embedding tensor J of dimension $T \times N \times 256$ is constructed from the concatenation of tensors \mathbf{X}^{co} and \mathbf{X}_s . The denoising step index s is also embedded into a 128-dimensional space using a linear layer followed by a SiLU activation function before being incorporated into each residual denoising block. Additionally, a learnable embedding of dimension 64 is assigned for each agent, referred to as emb , and concatenated with the binary mask \mathbf{M} , forming the mask/embedding tensor, denoted as \mathbf{M}_{emb} .

Residual Denoising Block Each residual denoising block

processes three inputs: the embedding tensor J , the embedded denoising step s , and the mask/embedding tensor \mathbf{M}_{emb} . First, the embedding of s is projected to a 256-dimensional space using a linear transformation to align with the dimensionality of J . This transformed embedding is then summed to each state in J . Next, the resulting tensor passes through the **Social-temporal Block** (see Fig. 1), which sequentially applies the Temporal Mamba and the Social Transformer. The output tensor from this block is projected to a 512-dimensional space. In parallel, the tensor \mathbf{M}_{emb} is also embedded into the same $T \times N \times 512$ dimensional space. These two tensors are summed to form the input to the next stage.

This combined tensor is processed by a **Gate-filter Block**. Here the tensor is split into two components: Gate and Filter, both of size $T \times N \times 256$. The Gate is passed through a sigmoid activation, while the Filter is passed through a hyperbolic tangent (Tanh). The two resulting tensors are state-wise multiplied (Hadamard product) and then projected back into a 512-dimensional space via a linear transformation. The output of the Gate-filter Block is split into two tensors of size $T \times N \times 256$. The first tensor is added to the original J , producing a refined version of J that is passed to the next residual denoising block. The second tensor serves as the skip connection output, J_{skip} , of the current residual denoising block.

Output tensor The J_{skip} outputs from all residual denoising blocks are summed together and passed through a linear layer followed by a ReLU activation function. This operation produces a tensor of dimension $T \times N \times 4$. This tensor is further split into two tensor of size $T \times N \times 2$ each: predicted mean noise, representing the central tendency of the noise distribution $\epsilon_{\theta}^{\mu}(\mathbf{X}_s, s, \mathbf{X}^{\text{co}})$, and the predicted standard deviation $\epsilon_{\theta}^{\sigma}(\mathbf{X}_s, s, \mathbf{X}^{\text{co}})$, obtained after applying a Sigmoid function to bound the values within the interval (0,1). Both tensors are symbolized as μ and σ in Fig. 1.

This design ensures that the model not only reconstructs trajectories but also quantifies uncertainty at each predicted state by modeling the noise distribution with a diagonal covariance structure, as explained in the main paper.

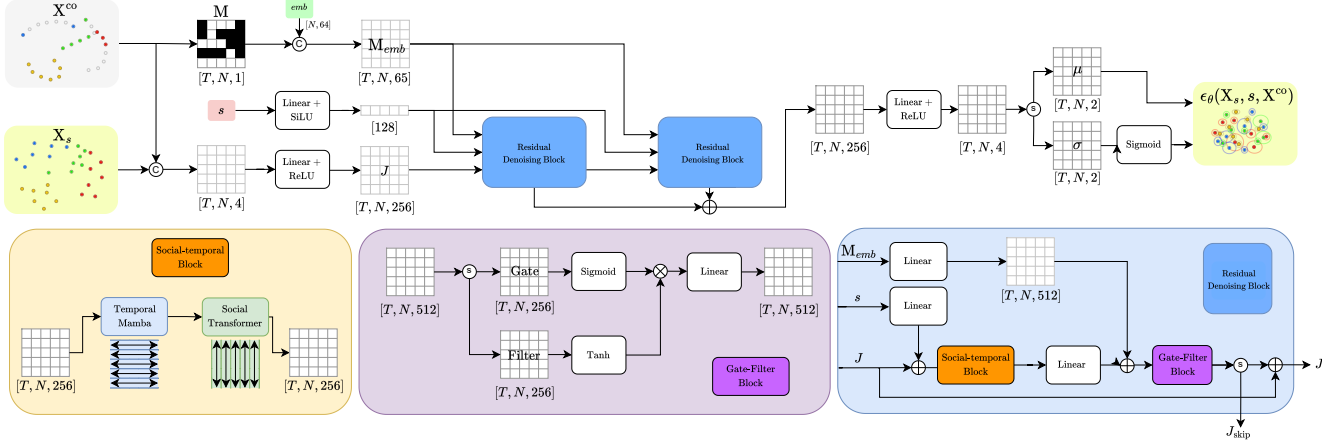


Figure 1. Unified Uncertainty-aware Diffusion Model (U2Diff)

1.3. Architecture ablation

The main paper provides an ablation analysis on setting $\lambda = 0$ or training from scratch, demonstrating the influence of the proposed loss function. Here, we present a detailed ablation study focusing on the key architectural components of U2Diff to further substantiate their contributions.

Table 1 reports results using the minSADE₂₀ metric across all datasets for the following configurations: **w/o TM** (without the Temporal Mamba) which is replaced with a Transformer Encoder, reverting to as the original design in CSDI [55], and **w/o ST** (without Social Transformer) responsible for encoding agent interactions.

We showcase the importance of replacing Temporal Mamba by transformers to enhance significantly the performance with respect the original CSDI [55]. Also we show the extreme importance in encoding the agents interactions with the Social Transformer in our multi-agent domain.

U2Diff	Basketball-U (Feet)	Football-U (Yards)	Soccer-U (Pixels)	NBA (Meters)
w/o TM (CSDI [55])	3.74	2.70	58.78	1.73
w/o ST	4.06	4.75	96.66	1.92
Ours ($\lambda = 0$)	3.10	2.37	51.27	1.50
Ours	3.13	2.35	51.14	1.48

Table 1. Ablation study on U2Diff using the minSADE₂₀ ↓ metric. The results are presented across all evaluated datasets.

2. Rank Neural Network

This section provides an in-depth explanation of the Rank Neural Network (RankNN) approach.

2.1. Implementation

During training, we generate $K = 20$ modes online using the U2Diff model with trained and frozen weights. The RankNN architecture is trained using a batch size of 32

scenes for 20 epochs. Depending on the dataset, each epoch on a RTX A6000 takes between 30 and 180 minutes.

To ensure convergence in Basketball-U and Soccer-U datasets, where stochasticity posed challenges, we first pre-trained RankNN using online generations from a suboptimal U2Diff model trained with a batch size four times larger. Since the suboptimal model produces more distinguishable variations in quality, this pretraining phase makes it easier for RankNN to learn meaningful ranking patterns before fine-tuning on higher-quality generations.

2.2. Architecture details

To complement the explanation provided in the main paper, Fig. 2 presents a visual diagram of the RankNN architecture. Similar to U2Diff, the Social-Temporal operation involves sequential processing through the Temporal Mamba and Social Transformer modules.

2.3. Architecture ablation

We perform an ablation study on the RankNN architecture, presented in Table 2, using the Spearman correlation coefficient (ρ) as the evaluation metric to measure the alignment between e and SADE. The study evaluated the following configurations: **w/o TM w/o ST** skips the Social-temporal Block entirely, bypassing both Temporal Mamba and Social Transformer processing; **w/o TM** excludes the Temporal Mamba processing, while retaining the Social Transformer; **w/o ST** excludes the Social Transformer processing, while retaining the Temporal Mamba; **w/o MST** skips the Multi-scene Transformer processing; and **w/o VAR** removes the predicted variance $\text{Var}(\mathbf{X}_0)$ from the input to RankNN, using only the mean predicted locations \mathbf{X}_0 and the binary mask M . The last configuration reduces the input tensor from $K \times T \times N \times 5$ to $K \times T \times N \times 3$.

The results emphasize the importance of processing states through the Social-Temporal Block, demonstrating

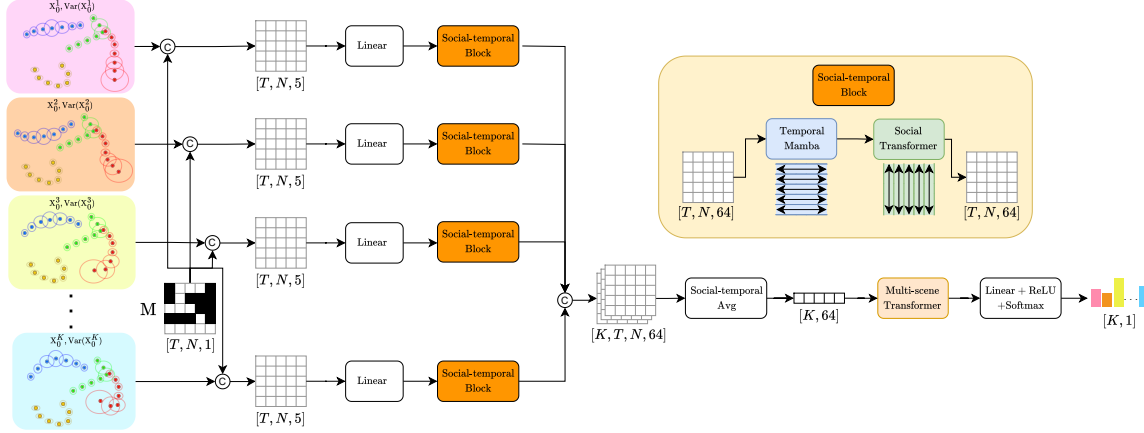


Figure 2. **Rank Neural Network architecture (RankNN).**

the critical contributions of both the Temporal Mamba and the Social Transformer. Furthermore, the w/o MST configuration results in sub-optimal performance, particularly on the Football-U dataset, underscoring the value of Multi-scene Transformer processing. Finally, the w/o VAR configuration highlights the essential role of the predicted variance $\text{Var}(\mathbf{X}_0)$ from U2Diff.

approach effectively captures uncertainty in trajectory predictions.

RankNN	Basketball-U	Football-U	Soccer-U	NBA
w/o TM w/o ST	0.28	0.37	0.31	0.40
w/o TM	0.35	0.50	0.54	0.45
w/o ST	0.39	0.51	0.34	0.45
w/o MST	0.52	0.47	0.65	0.50
w/o VAR	0.54	0.56	0.55	0.50
Ours	0.56	0.59	0.72	0.51

Table 2. **Ablation study on RankNN using the Mean of Spearman correlation coefficient ($\rho \uparrow$).** The results are presented across all evaluated datasets.

3. Qualitative results

We present additional qualitative results in Fig. 3, which extends Figure 4 from the main paper. In this figure, we compare the performance of our method with LED[38] and UniTraj[60] across all four datasets. Notably, our method outperforms LED in capturing the ball-possessor relationship, a key aspect of multi-agent sports scenarios. We explicitly highlight the ball-possessor with a pink rectangle, making it easier to visualize how different methods handle this critical interaction. Regarding the completion tasks, our method shows better alignment of the predictions with the ground truth, especially in the already observed-reconstructed states, compared UniTraj.

In addition, we include the variant Ours-20, which represents the distribution of predictions over 20 sampled trajectories. This variant serves to highlight the alignment between the predicted variance from our model (Ours) and the sampling approximation (Ours-20), demonstrating how our

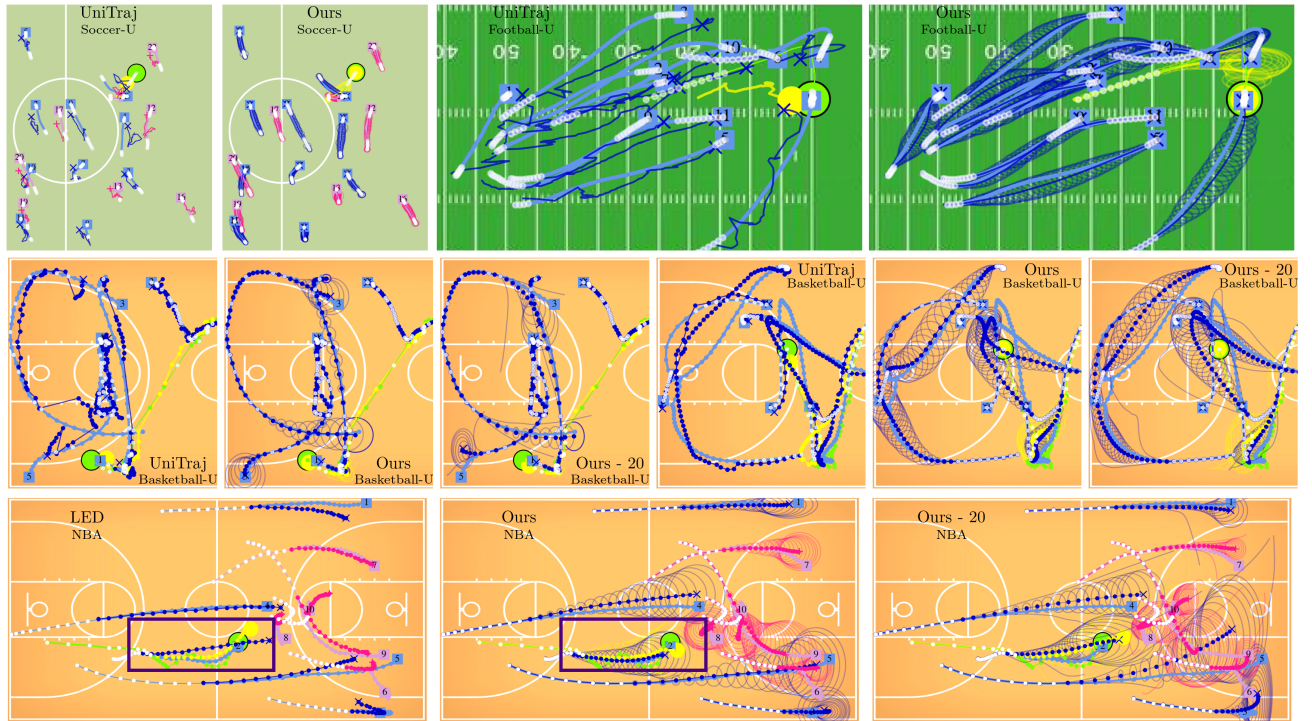


Figure 3. **Qualitative comparisons in trajectory completion and forecasting.** Ground truth player locations are shown in bright blue and pink, and the ball in green. Model input observations are represented in white. In LED, Unitraj and Ours, the predicted mode with the best minSADE_{20} is shown, with players in dark blue and pink, and the ball in yellow. Our model’s predictions also include estimated state-wise variances. In Ours-20, the prediction distribution over 20 samples is illustrated, including upper and lower bound modes.