

MoFlow: One-Step Flow Matching for Human Trajectory Forecasting via Implicit Maximum Likelihood Estimation based Distillation

Supplementary Material

The supplementary material is organized as follows. First, we introduce additional implementation details in Appendix A, including our MoFlow flow-matching objective in Appendix A.1, training and sampling steps in Appendix A.2 and Appendix A.3, as well as the architectural details in Appendix A.4. In Appendix B, we include additional qualitative results on the NBA sports dataset, demonstrating the effectiveness of our proposed methods. Moreover, we present additional ablation studies in Appendix C to showcase the success of inference-time speed-up, thanks to our IMLE distillation scheme in Appendix C.1, and the impact of training hyperparameters of IMLE in Appendix C.3.

A. Further Implementation Details

A.1. Flow Matching Objective Formulation

In Section 4.2, we define the multi-modal motion prediction objective for the flow matching model and explain the key implementation details. To maintain self-consistency, we reintroduce the fundamental concepts from scratch.

Recall that our goal is to generate K trajectories $\{Y_1, Y_2, \dots, Y_K\}$ to capture the diverse motion patterns of the agents under consideration, given their overall context information denoted as C . In the flow matching framework, we extend the notation by introducing a superscript to indicate flow time, with the denoising ODE intermediate states represented as $\{Y_1^t, Y_2^t, \dots, Y_K^t\}, t \in [0, 1]$.

However, in the dataset, we observe only a single future trajectory Y^1 conditioned on the context C . This limitation prevents us from directly adopting the vanilla flow matching objective, which involves linearly mixing clean data and noisy vectors to construct training objectives. Instead, we utilize data-space prediction and develop the multi-modal learning loss accordingly.

To achieve this, we first transform the original loss, which involves learning a vector field v_θ , into predicting the data at time $t = 1$:

$$\begin{aligned} \mathcal{L}_{\text{FM}} &= \mathbb{E}_{Y^t, Y^1, t} \left[\left\| v_\theta(Y^t, C, t) - (Y^1 - Y^0) \right\|_2^2 \right], \\ &= \mathbb{E}_{Y^t, Y^1, t} \left[\left\| v_\theta(Y^t, C, t) - \frac{Y^1 - Y^t}{1 - t} \right\|_2^2 \right], \\ &= \mathbb{E}_{Y^t, Y^1, t} \left[\left\| \frac{Y^t + (1 - t)v_\theta(Y^t, C, t) - Y^1}{1 - t} \right\|_2^2 \right]. \end{aligned}$$

We then define the network as a reparameterized model D_θ , which implicitly learns the vector field v_θ through a linear

transformation:

$$\begin{aligned} D_\theta(Y^t, C, t) &:= Y^t + (1 - t)v_\theta(Y^t, C, t), \\ \mathcal{L}_{\text{FM}} &= \mathbb{E}_{Y^t, Y^1, t} \left[\frac{\|D_\theta(Y^t, C, t) - Y^1\|_2^2}{(1 - t)^2} \right]. \end{aligned}$$

Note that we are merely rearranging the network modules, while the loss functions remain exactly equivalent to those in the vanilla framework.

Next, we design a training loss to encourage the data prediction model D_θ to learn multi-modal trajectories. To achieve this, we employ a standard Transformer structure (as illustrated in the first figure in the main text) to generate K correlated predictions. Specifically, the model D_θ produces K scene-level waypoint predictions, denoted as $\{S_i\}_{i=1}^K, S_i \in \mathbb{R}^{A \times 2T_f}$, along with the corresponding classification logits $\{\zeta_i\}_{i=1}^K, \zeta_i \in \mathbb{R}$. For simplicity, we omit the time-dependent coefficients and apply a combined regression and classification loss as follows:

$$\bar{\mathcal{L}}_{\text{FM}} = \mathbb{E}_{Y^t, Y^1, t} \left[\|S_{j^*} - Y^1\|_2^2 + \text{CE}(\zeta_{1:K}, j^*) \right], \quad (\text{A.1})$$

$$j^* = \arg \min_j \|S_j - Y^1\|_2^2, \quad (\text{A.2})$$

where $\text{CE}(\cdot, \cdot)$ means the cross-entropy loss.

A.2. MoFlow Teacher Model Training

We present the training algorithm for the teacher model with a modified objective in Algorithm 1. Notably, we observe that using tied noise across all K components stabilizes the training process. In contrast, untied noise introduces excessive variability, making convergence significantly more challenging. For the flow time scheduler $p_t(\cdot)$, we employ a logit-normal distribution: $\text{logit}(t) \sim \mathcal{N}(\mu_t, \sigma_t^2)$. In practice, one can sample a random variable $\kappa \sim \mathcal{N}(\mu_t, \sigma_t^2)$ and apply the standard logistic function, $\frac{1}{1 + e^{-\kappa}}$, to obtain the desired samples for t . The parameters $\mu_t = -0.5$ and $\sigma_t = 1.5$ are selected based on a hyperparameter sweep.

Empirically, we observe that the data space prediction loss function defined in Eq. (A.1) suffers from overfitting when t is close to 1. This is because, in this regime, the noisy input $Y_{1:K}^t$ becomes highly similar to the clean trajectory $Y_{1:K}^1$. When conditioned on the future trajectory $Y_{1:K}^1$, the noisy vector $Y_{1:K}^t$ at time t follows a Gaussian distribution $\mathcal{N}(tY_{1:K}^1, (1 - t)^2 I)$. Its variance, $(1 - t)^2$, decreases quadratically as t approaches 1. These overfitting effects cause the flow matching model to adopt a cheating solution during training, relying excessively on the input $Y_{1:K}^t$.

Algorithm 1 MoFlow Teacher Model Training

```

1: Require: Training dataset  $p_{\mathcal{D}}$ , learning rate  $\eta$ , teacher model  $D_{\theta}$  for data prediction
2: Initialize the parameters  $\theta$  of the network  $D_{\theta}$ .
3: repeat
4:    $(Y^1, C) \sim p_{\mathcal{D}}$   $\triangleright$  Sample observed data
5:    $Y_s^0 \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ 
6:    $Y_{1:K}^0 \leftarrow \text{repeat}(Y_s^0, K)$   $\triangleright$  Tied noise
7:    $Y_{1:K}^1 \leftarrow \text{repeat}(Y^1, K)$ 
8:    $t \sim p_t(t)$   $\triangleright$  Time scheduler
9:    $Y_{1:K}^t \leftarrow tY_{1:K}^1 + (1-t)Y_{1:K}^0$ 
10:   $S_{1:K}, \zeta_{1:K} \leftarrow D_{\theta}(Y_{1:K}^t, C, t)$ 
11:   $\theta \leftarrow \theta - \eta \nabla_{\theta} \tilde{\mathcal{L}}_{\text{FM}}(S_{1:K}, \zeta_{1:K}, Y^1)$   $\triangleright$  Eq. (A.1)
12: until converged

```

Algorithm 2 MoFlow Teacher Model Sampling

```

1: Require: Evaluation dataset  $p'_{\mathcal{D}}$ , teacher model  $D_{\theta}$  for data prediction, sampling steps  $T$ 
2: repeat
3:    $(\cdot, C) \sim p'_{\mathcal{D}}$   $\triangleright$  Sample context data
4:    $Y_s^0 \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ 
5:    $Y_{1:K}^0 \leftarrow \text{repeat}(Y_s^0, K)$   $\triangleright$  Tied noise
6:   for  $n \in [0, 1, \dots, T-1]$  do
7:      $S_{1:K}, \zeta_{1:K} \leftarrow D_{\theta}(Y_{1:K}^{\tau_n}, C, \tau_n)$ 
8:      $v_{\theta}^{(i)} = \frac{S_i - Y_i^{\tau_n}}{1 - \tau_n}, \quad \forall i \in [K]$ 
9:      $v_{\theta} = \text{concat}(v_{\theta}^{(1)}, v_{\theta}^{(2)}, \dots, v_{\theta}^{(K)})$ 
10:     $Y_{1:K}^{\tau_{n+1}} \leftarrow Y_{1:K}^{\tau_n} + (\tau_{n+1} - \tau_n)v_{\theta}$ 
11:   end for
12: until finished

```

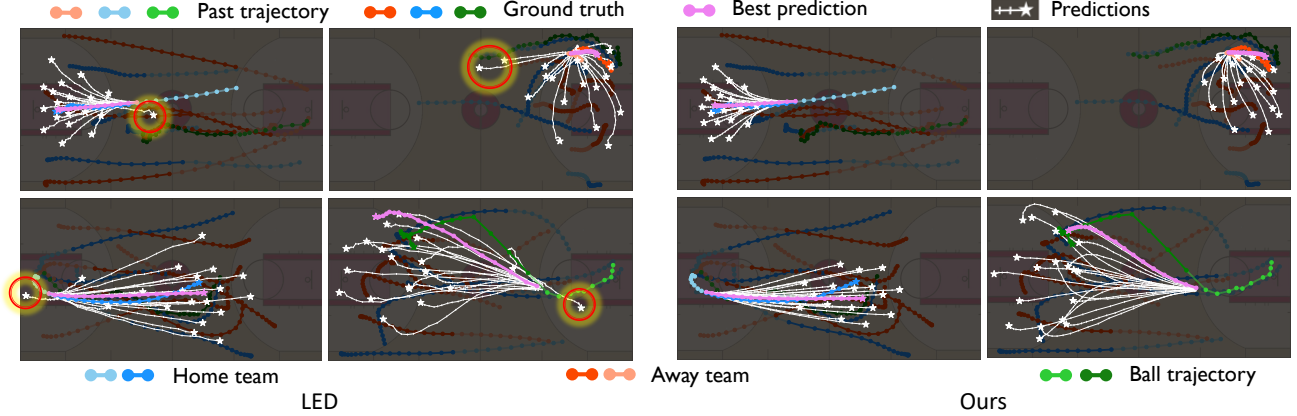


Figure 1. Qualitative results on NBA dataset in terms of diversity. Our method generates diverse samples that are more socially plausible. Some of the trajectories generated by LED model, which are highlighted by red circles, do not adhere to the basketball game patterns or rules. (Light color indicates past trajectory while dark color means future trajectory; blue/orange/green color: two teams and the basketball; pink color: the sample that is the closest to the Ground truth in L_2 sense among $K = 20$ predictions)

and generalizing poorly on the test dataset. To address this issue, we propose a flow time-dependent masking mechanism to encourage the model to extract useful signals from the context C . Specifically, we introduce a masking mechanism applied to the noise embedding, as illustrated in Fig. 1, which uses an S-shaped logistic function as the threshold:

$$f_m(t) = \frac{1}{1 + e^{-k(t-m)}}.$$

During training, for a randomly sampled time step t from the time scheduler, we mask the noisy embedding with zeros with a probability of $f_m(t)$. This serves as a simple embedding-level dropout mechanism, applied only during training. In our experiments, we set $k = 20$ and $m = 0.5$.

A.3. MoFlow Teacher Model Sampling

In Algorithm 2, we solve the denoising ODEs during sampling by leveraging the K -correlated trajectory predictions $\{S_i\}_{i=1}^K$ produced by D_{θ} to compute the K -shot vector

field:

$$v_{\theta}^{(i)} = \frac{S_i - Y_i^t}{1 - t}, \quad \forall i \in [K].$$

The initial values Y_i^0 for all components are sampled from a standard normal distribution. The process iteratively updates $Y_{1:K}^t$ based on the predicted vector field as the flow time t progresses towards $t = 1$.

We use a mapping function τ_n to represent the actual flow time t at sampling iteration $n \in [0, T]$, given a total computation budget of T steps. The mapping function must satisfy two boundary conditions: $\tau_0 = 0$ and $\tau_T = 1$. In our experiments, we set $T = 100$. Following [3, 10], we design a non-linear time mapping function as follows:

$$\tau_n = \begin{cases} \frac{n}{1000}, & \text{if } n \leq \frac{T}{2}, \\ \frac{T}{500} + \frac{(1.0 - \frac{T}{500})(n - \frac{T}{2})^p}{(\frac{T}{2})^p}, & \text{if } n > \frac{T}{2}. \end{cases}$$

Empirically, we set $p = 5$ based on a hyperparameter grid search.

Table 1. Ablation study on various components of our MoFlow on the **NBA** dataset. (PE-K) Positional Encoding (PE) on number of predictions (K); (PE-A) PE on agent-level only; (w/o PE) No PE; (Uniform) uniform noise/time schedule; (IID) i.i.d. noise instead of shared noise; (w/o Mask) Turn off the flow-time dependent masking mechanism; (Ours) Our MoFlow with all modules on. We report $\min_{20}\text{ADE}/\min_{20}\text{FDE}$ (meters) for empirical performance. We **bold** the top results.

<i>Time</i>	PE-K	PE-A	w/o PE	Uniform	IID	w/o Mask	Ours
1.0s	0.19 /0.26	0.41/0.74	0.45/0.84	0.19 /0.26	0.26/0.44	0.19 /0.27	0.19/0.25
2.0s	0.35/ 0.47	0.88/1.77	1.02/2.10	0.35/0.48	0.59/1.12	0.38/0.56	0.34/0.47
3.0s	0.52 /0.68	1.37/2.72	1.61/3.22	0.53/0.68	0.97/1.87	0.59/0.80	0.52/0.67
4.0s	0.71 /0.88	1.83/3.51	2.14/4.05	0.71 /0.88	1.36/2.46	0.82/1.07	0.71/0.87

A.4. Network Architecture

Building on prior studies [4, 7, 18], we adopt the spatio-temporal transformer encoder to jointly model temporal and social dimensions, capturing complex agent interactions over time and space. To enhance performance across datasets, the encoder architecture is dataset-specific: the spatio-temporal transformer encoder is used for ETH-UCY and SDD, while a PointNet-like encoder [11] is employed for the NBA dataset. Our spatio-temporal encoder processes both the context embedding and noise embedding through MLP layers, which are further combined with the flow-matching time signal during the teacher model’s training. The encoder for the student IMLE model employs a similar structure, differing only in its handling of the flow-matching time. In both transformer-based encoders leverage skip connections and share a configuration of 128 features, a feed-forward dimension of 512, eight attention heads, and consist of four layers.

In our the motion decoder, we employ additional self-attention to capture interactions among K scene predictions, as depicted in Fig 1., complementing the temporal and social interactions modeled by the encoder. The embedding dimension, feed-forward dimension, and number of attention heads in the decoder match those of the spatial temporal encoder. The decoder comprises four blocks with each block performing factorized self-attention both over the sample dimension and the agent dimension [8]. To circumvent the potential overfitting issue, the attention dropout rate is set to 0.1.

Both our MoFlow model and the IMLE model are smaller in size compared to the LED initializer [7], with $\sim 1\text{M}$ fewer parameters.

A.5. Model Training Details

We preprocess the trajectories using a simple min-max normalization technique, scaling future relative motion to the range $[-1, 1]$ linearly to facilitate the training of the flow-matching model. A basic transformer architecture, depicted in Fig 1., serves as the backbone of our approach. We find the standard transformer implementation to be sufficient for our task, requiring no special training tricks to

achieve strong performance. To facilitate the learning of inter-agent spatial relationships, we apply sinusoidal positional encoding across agents, assigning unique relative positions to each agent’s representation. In the motion decoder, we reinforce the agent-level positional encoding and introduce an additional positional encoding at the prediction level. Self-attention is then applied alternately at the agent and prediction levels, strengthening inter-agent interactions and improving overall scene coherence. The student model trained with IMLE objective shares the same architecture as the teacher model, except that it does not need the flow time positional encoding layers. For teacher model sampling, we adopt 100 steps to solve the denoising ODEs and generate samples. These samples are used both to evaluate teacher model performance and to train the IMLE distillation models. To enhance IMLE training efficiency, we compute and save the teacher model samples in advance. All the training is conducted on NVIDIA RTX6000 and A40 GPUs using the AdamW [6] optimizer in PyTorch [9], with weight decay set to 0.01.

B. Further Qualitative Results

We would love to show more qualitative results on NBA dataset. In Fig. 1, we compare our 20 predictions with the samples¹ generated by LED [7] model. Upon closer examination of the predictions generated by the LED model, we observe that certain trajectories, which highlighted by the red circles, move in implausible, opposite directions. In contrast, our model effectively captures the general movement, steering the trajectories towards more plausible directions. Notably, the two figures in the first row of the LED model predictions are classified as backcourt violations defined in the basketball rules. Such violations are rare in the NBA training dataset. Notice that violations are not present in our predictions, indicating that our model generates trajectories that are more realistic and contextually appropriate for basketball games. According to Fig. 5, we have success-

¹Note that these are the exactly two same scenes with identical ego agents, as demonstrated by their paper and [codebase](#). The coloring has been adjusted to align with our visualizations, and we have replaced their mean prediction with the best-of-20 prediction for consistency.

	$\omega = 0$				$\omega = 1$	$\omega = 5$
	DDPM	LED	MoFlow	IMLE	MoFlow*	MoFlow*
(a) @2s	0.44/0.64	0.37/0.57	0.34/0.47	0.34/0.47	0.37/0.52	0.42/0.61
(a) @4s	0.94/1.21	0.81/1.10	0.71/0.87	0.71/0.86	0.75/0.96	0.81/1.12
(b) @2s	1.15/2.30	0.89/1.84	0.83/1.64	0.83/1.65	0.84/1.68	0.86/1.73
(b) @4s	2.40/4.65	1.95/3.84	1.71/3.34	1.73/3.35	1.74/3.37	1.81/3.49
MASD [17]	6.41	15.74	5.85	5.78	5.54	5.00

Table 2. Joint metrics performance on NBA dataset. * means we train our MoFlow using the new objective. (a) $\min_{20}\text{ADE}/\min_{20}\text{FDE}$ metric; (b) a new metric $\min_{20}\text{JADE}/\min_{20}\text{JFDE}$; (c) MASD diversity metric [13, 17]. DDPM refers to the stage-one diffusion model used in LED [7]. We use the checkpoint from the codebase provided by LED.

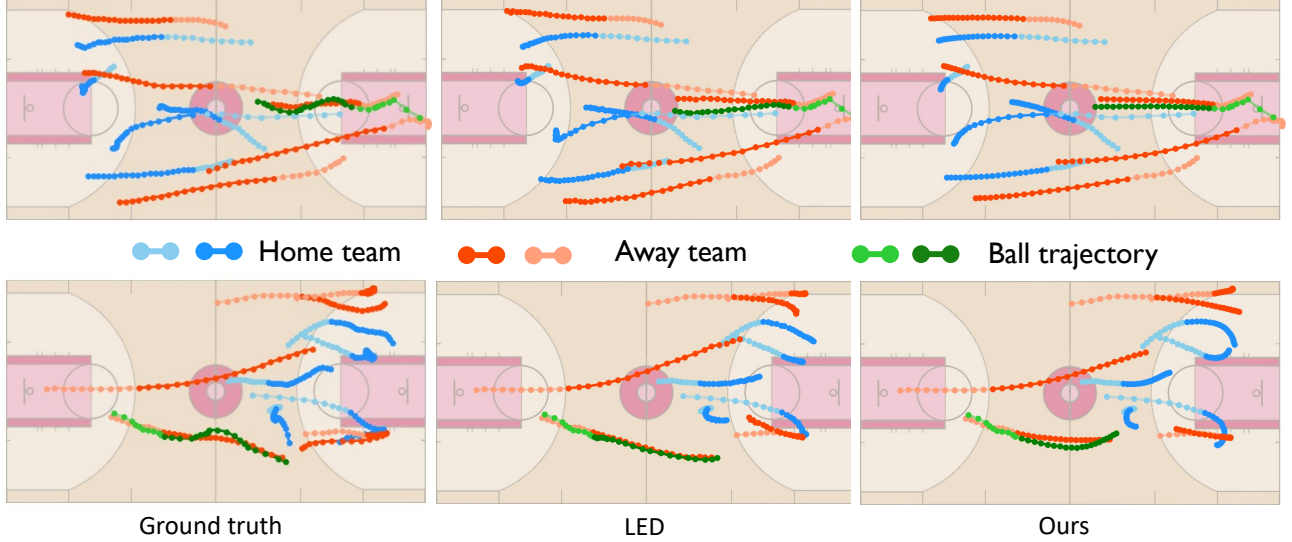


Figure 2. More qualitative results on the NBA dataset show a comparison between the best-of-20 predictions from our MoFlow IMLE distillation method, the best-of-20 predictions the LED method, and the ground truth future trajectories. The visualization demonstrates that our approach produces predictions that more closely align with the ground truth trajectories compared to the LED model. (Light color indicates past trajectory while dark color means future trajectory; blue/orange/green color: two teams and the basketball)

Table 3. Ablations on IMLE sample size m on the NBA dataset. We report $\min_{20}\text{ADE}/\min_{20}\text{FDE}$ (meters) for performance.

Trajectory Time	$m = 5$	$m = 10$	$m = 20$	$m = 40$
1.0s	0.22/0.27	0.19/0.26	0.18/0.25	OOM
2.0s	0.38/0.42	0.35/0.48	0.35/0.47	OOM
3.0s	0.58/0.70	0.55/0.69	0.52/0.67	OOM
Total (4.0s)	0.78/0.95	0.73/0.89	0.71/0.87	OOM

fully predicted the direction of the future trajectories given the context on a regular basis while other state-of-the-art method failed to achieve. In particular, we highlight the subtle yet essential differences among the best-of-20 predictions from our MoFlow IMLE distillation method, the best-of-20 predictions from the LED method, and the ground truth future trajectories by zooming in on the details.

It is important to highlight that our MoFlow model can generate K scene-level trajectory predictions, with corresponding classification logits $\{\zeta_i\}_{i=1}^K, \zeta_i \in \mathbb{R}$, depicted by Fig. 4 and Fig. 3. This means we have an empirical dis-

tribution for all the trajectories that we generate for each agent. While the predicted trajectories may sometimes be widely distributed, our MoFlow model effectively differentiates their plausibility. Additionally, we compute the Pearson correlation between ADE and predicted K probabilities for each agent and average the results across all trajectories, yielding a negative correlation of -0.40 . Next, we report ADE@4s of the trajectory with largest probability and ADE@4s with the least one on NBA dataset [1.94/3.41] to validate the output probabilities. The performance gap is significant.

C. Additional Experimental Results

In this section, we conduct ablation experiments on the NBA dataset to validate our model design choices. Due to a recent paper [14], we will report another set of results under the joint metric to show the capacity of our models.

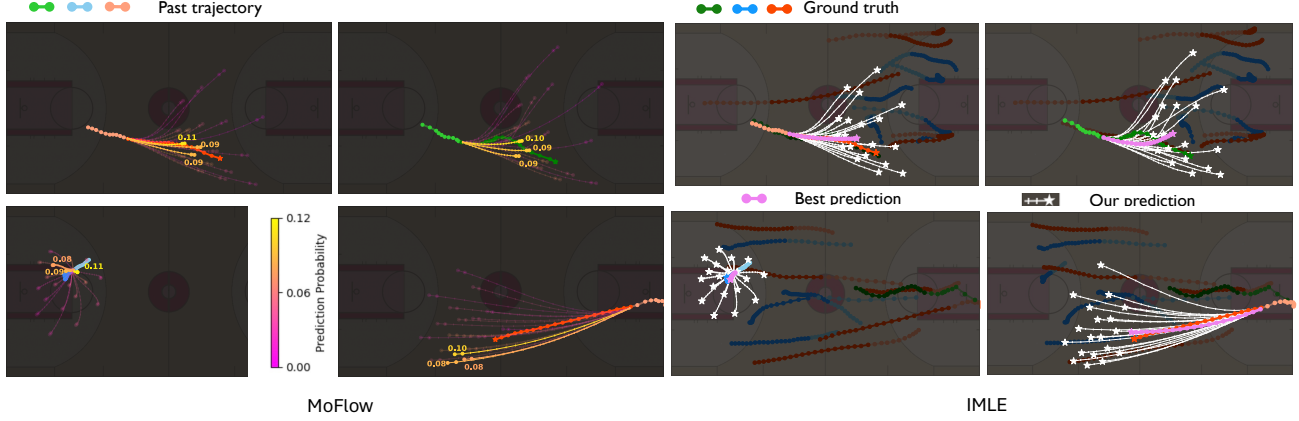


Figure 3. Some scenes from NBA dataset with MoFlow predictions for one agent and corresponding IMLE predictions.

Table 4. Comparison with baseline distillation models on **ETH-UCY** dataset. $\min_{20}\text{ADE}/\min_{20}\text{FDE}$ (meters) are reported. Bold/underlined fonts represent the best/second-best result. Note that we evaluate our approach on this dataset with the same split as SGAN [5] while Tab. 2. in the main text is based on the dataset and train-test split released by the LED [7] group.

Subsets	MID [4]	GroupNet [15]	TUTR [12]	EqMotion [16]	EigenTraj [1]	LED [7]	SingularTraj [2]	MoFlow	IMLE
ETH	0.39/0.66	0.46/0.73	0.40/0.61	0.40/0.61	<u>0.36/0.53</u>	0.39/0.58	0.35/0.42	0.39/0.55	0.40/0.61
HOTEL	0.13/0.22	0.15/0.25	0.11/0.18	<u>0.12/0.18</u>	<u>0.12/0.19</u>	0.11/0.17	0.13/0.19	0.11/0.17	0.12/0.18
UNIV	0.22/0.45	0.26/0.49	<u>0.23/0.42</u>	<u>0.23/0.43</u>	0.24/0.34	0.26/0.44	0.25/0.44	<u>0.22/0.39</u>	0.24/0.44
ZARA1	0.17/0.30	0.21/0.39	0.18/0.34	0.18/0.32	0.19/0.33	0.18/0.26	0.19/0.32	0.17/0.29	0.17/0.31
ZARA2	<u>0.13/0.27</u>	0.17/0.33	<u>0.13/0.25</u>	<u>0.13/0.23</u>	0.14/0.24	<u>0.13/0.22</u>	0.15/0.25	0.12/0.22	0.13/0.24
AVG	<u>0.21/0.38</u>	0.25/0.44	<u>0.21/0.36</u>	0.32/0.35	<u>0.21/0.34</u>	<u>0.21/0.33</u>	<u>0.21/0.32</u>	0.21/0.32	0.21/0.36

C.1. Sampling Speed-up Effect

We demonstrate the sampling speed-up effect in Tab. 5, reporting the average sampling time per scene for runtime. Thanks to the one-step inference enabled by IMLE distillation, our student model achieves significantly faster sampling with 100x fewer NFEs, reducing runtime by 98% on the same hardware. Notably, we found that IMLE distillation does not compromise empirical performance in our setup. Moreover, both our teacher and student model

Table 5. Sampling speed-up comparison on the **NBA** dataset. We report $\min_{20}\text{ADE}/\min_{20}\text{FDE}$ (meters) for empirical performance. DDPM refers to the teacher diffusion model used in LED [7].

Efficiency	MoFlow	IMLE	DDPM	LED
NFE	100	1	100	6
Runtime (ms)	33.20	0.70	796	22.38
Performance	MoFlow	IMLE	DDPM	LED
1.0s	0.18/0.25	0.18/0.25	0.20/0.28	0.18/0.27
2.0s	0.34/0.47	0.35/0.47	0.43/0.64	0.37/0.56
3.0s	0.52/0.67	0.52/0.67	0.68/0.95	0.58/0.84
Total (4.0s)	0.71/0.87	0.71/0.87	0.93/1.20	0.81/1.10

achieve faster sampling speeds than state-of-the-art methods, while delivering superior performance.

C.2. Ablations on Flow Matching Configurations

To assess the significance of each module, we conduct an ablation study on different components of our MoFlow. In Section 4.2, we discussed the design of input-output dimension adaptation and the time schedule. Here, we demonstrate the superiority of our current configuration through the results presented in Tab. 1.

From Tab. 1, we observe that positional encoding applied at the predictions (K) level provides a greater advantage compared to positional encoding applied at the agent level. Additionally, incorporating shared noise yields strong final results without leading to the variance explosion observed in the IID column.

C.3. Ablations on IMLE Configurations

According to the IMLE training principle outlined in Algorithm 1, the student model needs to sample $m > 1$ instances to select the one closest to the teacher model’s sample. We present the comparison results for different values



Figure 4. Scenes from ETH-UCY datasets with MoFlow generation. The probability besides the trajectories are normalized from classification logit $\{\zeta_i\}_{i=1}^K$ via Softmax.

of m in Tab. 3. We choose $m = 20$ for its superior empirical performance. Note that m effectively enlarges the training-time mini-batch size by a factor of m . Therefore, setting m too large can lead to out-of-memory issues. Moreover, we present the JADE/JFDE results and conduct ablation on ω in by re-training our MoFlow with the new objective, which is identical to equations (10,11) [14]. Based on Tab. 2, we observe a trade-off between these two metrics: a marginal improvement in JADE/JFDE leads to a significant drop in ADE/FDE performance. Next, we assess our model’s ability to preserve the sample quality of the teacher model. Specifically, when analyzing the metric map-aware self-distance (MASD), we observe that the LED model fails to maintain this quantity, exhibiting a substantial deviation of $\sim 9\text{m}$ from the teacher model (DDPM). In contrast, our IMLE model preserves this quantity with remarkable accu-

racy, achieving a deviation of only $\sim 0.1\text{m}$. These results further underscore the superiority of our approach in the distillation task.

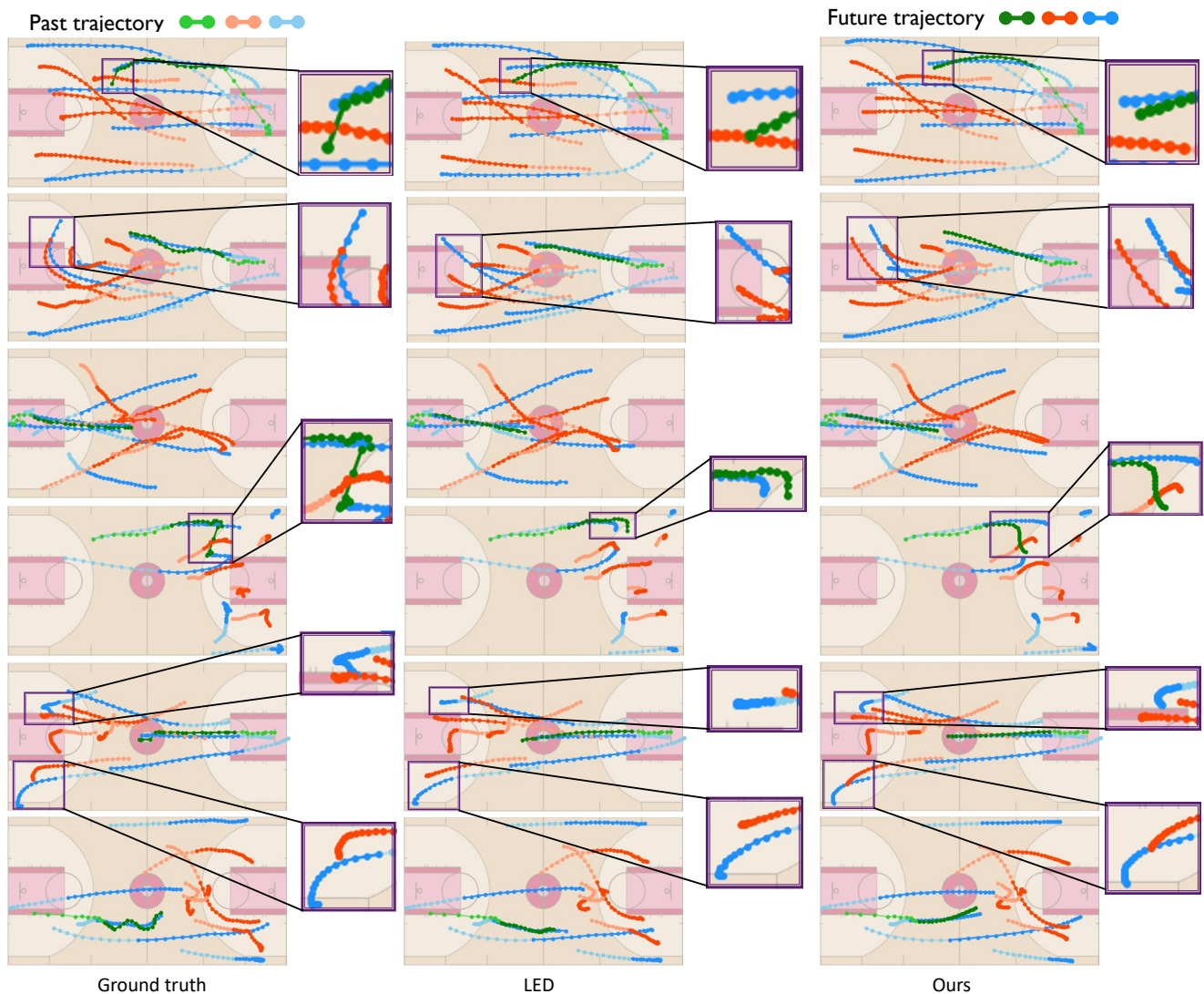


Figure 5. More qualitative results on the NBA dataset show a comparison among the best-of-20 predictions from our MoFlow IMLE distillation method, the best-of-20 predictions the LED method, and the ground truth future trajectories. The visualization demonstrates that our approach produces predictions that more closely align with the ground truth trajectories compared to the LED model. The key differences are zoomed in from the purple rectangles and displayed side-by-side, contrasting the ground truth, the best samples generated by the LED model with those produced by our approach. (blue/orange/green color: home team, away team and the basketball)

References

- ## References
- [1] Inhwon Bae, Jean Oh, and Hae-Gon Jeon. Eigentrajectory: Low-rank descriptors for multi-modal trajectory forecasting. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023. 5
 - [2] Inhwon 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*, 2024. 5
 - [3] Patrick Esser, Sumith Kulal, Andreas Blattmann, Rahim Entezari, Jonas Müller, Harry Saini, Yam Levi, Dominik Lorenz, Axel Sauer, Frederic Boesel, et al. Scaling rectified flow transformers for high-resolution image synthesis. In *Forty-first International Conference on Machine Learning*, 2024. 2
 - [4] 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 (CVPR)*, pages 17113–17122, 2022. 3, 5
 - [5] 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. 5

- [6] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019. 3
- [7] 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 (CVPR)*, pages 5517–5526, 2023. 3, 4, 5
- [8] Jiquan Ngiam, Vijay Vasudevan, Benjamin Caine, 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 future trajectories of multiple agents. In *International Conference on Learning Representations*, 2021. 3
- [9] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019. 3
- [10] Adam Polyak, Amit Zohar, Andrew Brown, Andros Tjandra, Animesh Sinha, Ann Lee, Apoorv Vyas, Bowen Shi, Chih-Yao Ma, Ching-Yao Chuang, et al. Movie gen: A cast of media foundation models. *arXiv preprint arXiv:2410.13720*, 2024. 2
- [11] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 652–660, 2017. 3
- [12] Liushuai Shi, Le Wang, Sanping Zhou, and Gang Hua. Trajectory unified transformer for pedestrian trajectory prediction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 9675–9684, 2023. 5
- [13] Simon Suo, Sebastian Regalado, Sergio Casas, and Raquel Urtasun. Trafficsim: Learning to simulate realistic multi-agent behaviors. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10400–10409, 2021. 4
- [14] 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. 4, 6
- [15] 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 (CVPR)*, pages 6498–6507, 2022. 5
- [16] 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. 5
- [17] Ye Yuan and Kris M. Kitani. Diverse trajectory forecasting with determinantal point processes. In *International Conference on Learning Representations*, 2020. 4
- [18] Ye Yuan, Xinshuo Weng, Yanglan Ou, and Kris Kitani. Agentformer: Agent-aware transformers for socio-temporal multi-agent forecasting. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021. 3