

# Leapfrog Diffusion Model for Stochastic Trajectory Prediction

Weibo Mao<sup>1</sup>, Chenxin Xu<sup>1</sup>, Qi Zhu<sup>1</sup>, Siheng Chen<sup>1,2\*</sup>, Yanfeng Wang<sup>2,1</sup>,

<sup>1</sup>Shanghai Jiao Tong University, <sup>2</sup>Shanghai AI Laboratory

{kirino.mao,xcxwakaka,georgezhu,sihengc,wangyanfeng}@sjtu.edu.cn

## Abstract

To model the indeterminacy of human behaviors, stochastic trajectory prediction requires a sophisticated multi-modal distribution of future trajectories. Emerging diffusion models have revealed their tremendous representation capacities in numerous generation tasks, showing potential for stochastic trajectory prediction. However, expensive time consumption prevents diffusion models from real-time prediction, since a large number of denoising steps are required to assure sufficient representation ability. To resolve the dilemma, we present LEapfrog Diffusion model (LED), a novel diffusion-based trajectory prediction model, which provides real-time, precise, and diverse predictions. The core of the proposed LED is to leverage a trainable leapfrog initializer to directly learn an expressive multi-modal distribution of future trajectories, which skips a large number of denoising steps, significantly accelerating inference speed. Moreover, the leapfrog initializer is trained to appropriately allocate correlated samples to provide a diversity of predicted future trajectories, significantly improving prediction performances. Extensive experiments on four real-world datasets, including NBA/NFL/SDD/ETH-UCY, show that LED consistently improves performance and achieves 23.7%/21.9% ADE/FDE improvement on NFL. The proposed LED also speeds up the inference 19.3/30.8/24.3/25.1 times compared to the standard diffusion model on NBA/NFL/SDD/ETH-UCY, satisfying real-time inference needs. Code is available at <https://github.com/MediaBrain-SJTU/LED>.

## 1. Introduction

Trajectory prediction aims to predict the future trajectories for one or multiple interacting agents conditioned on their past movements. This task plays a significant role in numerous applications, such as autonomous driving [5, 24], drones [11], surveillance systems [46], human-robot interaction systems [6], and interactive robotics [21, 26]. Recently, lots of fascinating research progresses have been made from

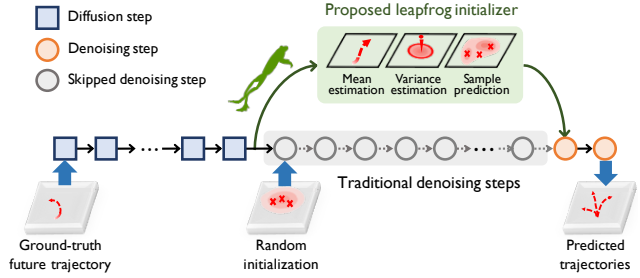


Figure 1. Leapfrog diffusion model uses the leapfrog initializer to estimate the denoised distribution and substitute a long sequence of traditional denoising steps, accelerating inference and maintaining representation capacity.

many aspects, including temporal encoding [7, 14, 47, 54], interaction modeling [1, 16, 19, 44, 50], and rasterized prediction [12, 13, 27, 49, 55]. In practice, to capture multiple possibilities of future trajectories, a real-world prediction system needs to produce multiple future trajectories. This leads to the emergence of stochastic trajectory prediction, aiming to precisely model the distribution of future trajectories.

Previous works have proposed a series of deep generative models for stochastic trajectory prediction. For example, [16, 19] exploit the generator adversarial networks (GANs) to model the future trajectory distribution; [28, 39, 50] consider the conditional variational auto-encoders (CVAEs) structure; and [3] uses the conditional normalizing flow to relax the Gaussian prior in CVAEs and learn more representative priors. Recently, with the great success in image generation [18, 34] and audio synthesis [4, 22], denoising diffusion probabilistic models have been applied to time-series analysis and trajectory prediction, and show promising prediction performances [15, 45]. Compared to many other generative models, diffusion models have advantages in stable training and modeling sophisticated distributions through sufficient denoising steps [9].

However, there are two critical problems in diffusion models for stochastic trajectory prediction. First, the real-time inference is time-consuming [15]. To ensure the representation ability and generate high-quality samples, an adequate number of denoising steps are required in standard diffusion models, which costs more computational time. For

\*Corresponding author.

example, experiments show that on the NBA dataset, diffusion models need about 100 denoising steps to achieve decent prediction performances, which would take  $\sim 886$ ms to predict; while the next frame comes every 200ms. Second, as mentioned in [2], a limited number of independent and identically distributed samples might not be able to capture sufficient modalities in the underlying distribution of a generative model. Empirically, a few independent sampled trajectories could miss some important future possibilities due to the lack of appropriate sample allocation, significantly deteriorating prediction performances.

In this work, we propose leapfrog diffusion model (LED), a novel diffusion-based trajectory prediction model, which significantly accelerates the inference speed and enables adaptive and appropriate allocations of multiple correlated predictions, providing sufficient diversity in predictions. The core idea of the proposed LED is to learn a rough, yet sufficiently expressive distribution to initialize denoised future trajectories; instead of using a plain Gaussian distribution as in standard diffusion models. Specifically, our forward diffusion process is the same as standard diffusion models, which assures that the ultimate representation ability is pristine; while in the reverse denoising process, we leverage a powerful initializer to produce correlated diverse samples and leapfrog or skip a large number of denoising steps; and then, use only a few denoising steps to refine the distribution.

To implement such a leapfrog initializer, we consider a reparameterization to alleviate the learning burden. We disassemble a denoised distribution into three parts: mean trajectory, variance, and sample positions under the normalized distribution. To estimate these three, we design three corresponding trainable modules, each of which leverages both a social encoder and a temporal encoder to learn the social-temporal features and produce accurate estimation. Furthermore, all the sample positions are simultaneously generated based on the same social-temporal features, enabling appropriate sample allocations to provide diversity.

To evaluate the effectiveness of the proposed method, we conduct experiments on four trajectory prediction datasets: NBA, NFL Football Dataset, Stanford Drones Dataset, and ETH-UCY. The quantitative results show we outperform the previous methods and achieve state-of-the-art performance. Specifically, compared to MID [15], the proposed leapfrog diffusion model reduces the average prediction time from  $\sim 886$ ms to  $\sim 46$ ms on the NBA dataset, while achieving a 15.6%/13.4% ADE/FDE improvement.

The main contributions are concluded as follows,

- We propose a novel LEapfrog Diffusion model (LED), which is a denoising-diffusion-based stochastic trajectory prediction model. It achieves precise and diverse predictions with fast inference speed.
- We propose a novel trainable leapfrog initializer to directly model sophisticated denoised distributions, acceler-

ating inference speed, and adaptively allocating the sample diversity, improving prediction performance.

- We conduct extensive experiments on four datasets including NBA, NFL, SDD, and ETH-UCY. Results show that i) our approach consistently achieves state-of-the-art performance on all datasets; and ii) our method speeds up the inference by around 20 times compared to the standard diffusion model, satisfying real-time prediction needs.

## 2. Related Work

**Trajectory prediction.** Early works on trajectory prediction focus on a deterministic approach by exploring force models [17, 31], RNNs [1, 33, 48], and frequency analysis [29, 30]. For example, [17] models an agent’s behavior with attractive and repulsive forces and builds the force equations for prediction. To capture the multi-modalities and model future distribution, recent works start to focus on stochastic trajectory prediction and have proposed a series of deep generative models. Generative Adversarial Network (GAN) structures [8, 10, 16, 19, 37, 43] are proposed to generate multiple future trajectory distribution. [23, 28, 39, 50, 52] use the Variational Auto-Encoder (VAE) structure and learn the distribution through variational inference. [3] relaxes the Gaussian prior and proposes to use the normalizing flow. Heatmap [12, 13, 27] is used for modeling future trajectories’ distribution on rasterized images. In this work, we propose a new diffusion-based model for trajectory prediction. Compared to previous generative models, our method has a large representation capacity and can model sophisticated trajectory distributions by using a number of diffusion steps. We also enable the correlation between samples to adaptively adjust sample diversity, improving prediction performance.

**Denoising diffusion probabilistic models.** Denoising diffusion probabilistic models (diffusion models) [18, 40, 42] have recently achieved significant results in image generation [9, 34] and audio synthesis [4, 22]. The idea of diffusion models is first proposed by DPM [40], which imitates the diffusion process in non-equilibrium statistical physics and reconstructs the data distribution using the denoising model. Later, [36, 45] propose diffusion models, combining with the seq-to-seq models, for probabilistic time series forecasting. MID [15] is the first to build diffusion models for trajectory prediction in modeling the indeterminacy variation process.

The standard diffusion models use hundreds of denoising steps, preventing these models from real-time applications. To accelerate the sampling process, DDIM [41] first predicts the original data and then estimates the direction to the next expected timestamp based on the non-Markov process. PD [38] applies the knowledge distillation on the denoising steps with a deterministic diffusion sampler, which will be repeated for times to accelerate the sampling. All these fast sampling methods start denoising from noise inputs, which are randomly and independently initialized. In this work, we

use a trainable leapfrog initializer to initialize a sufficiently expressive distribution, which replaces a large number of former denoising steps for much faster inference speed.

### 3. Background

#### 3.1. Problem Formulation

Trajectory prediction aims to predict an agent's future trajectory based on the past trajectories of itself and surrounding agents. For a to-be-predicted agent, let  $\mathbf{X} = [\mathbf{x}^{-T_p+1}, \mathbf{x}^{-T_p+2}, \dots, \mathbf{x}^0] \in \mathbb{R}^{T_p \times 2}$  be the observed past trajectory over  $T_p$  timestamps where  $\mathbf{x}^t \in \mathbb{R}^2$  records the 2D spatial coordinate at timestamp  $t$ . Let  $\mathcal{N}$  be the neighbouring agent set and  $\mathbb{X}_{\mathcal{N}} = [\mathbf{X}_{\mathcal{N}_1}, \mathbf{X}_{\mathcal{N}_2}, \dots, \mathbf{X}_{\mathcal{N}_L}] \in \mathbb{R}^{L \times T_p \times 2}$  be the past trajectories of neighbours, where  $\mathbf{X}_{\mathcal{N}_\ell} \in \mathbb{R}^{T_p \times 2}$  is the trajectory of the  $\ell$ th neighbour. The corresponding ground-truth future trajectory for the to-be-predicted agent is  $\mathbf{Y} = [\mathbf{y}^1, \mathbf{y}^2, \dots, \mathbf{y}^{T_f}] \in \mathbb{R}^{T_f \times 2}$  over  $T_f$  timestamps, where  $\mathbf{y}^t \in \mathbb{R}^2$  is the 2D coordinate at future timestamp  $t$ .

Because of the indeterminacy of future trajectories, it is usually more reliable to predict more than one trajectory to capture multiple possibilities. Here we consider stochastic trajectory prediction, which predicts the distribution of a future trajectory, instead of a single future trajectory. The goal of stochastic trajectory prediction is to train a prediction model  $g_\theta(\cdot)$  with parameters  $\theta$  to generate a distribution  $\mathcal{P}_\theta = g_\theta(\mathbf{X}, \mathbb{X}_{\mathcal{N}})$ . Based on this distribution  $\mathcal{P}_\theta$ , we can draw  $K$  samples,  $\hat{\mathcal{Y}} = \{\hat{\mathbf{Y}}_1, \hat{\mathbf{Y}}_2, \dots, \hat{\mathbf{Y}}_K\}$ , so that at least one sample is close to the ground-truth future trajectory. The overall learning problem is

$$\theta^* = \min_{\theta} \min_{\hat{\mathcal{Y}} \in \hat{\mathcal{Y}}} D(\hat{\mathbf{Y}}_i, \mathbf{Y}), \quad \text{s.t. } \hat{\mathcal{Y}} \sim \mathcal{P}_\theta. \quad (1)$$

#### 3.2. Diffusion Model for Trajectory Prediction

Here we present a standard diffusion model for trajectory prediction, which lays a foundation for the proposed method. The core idea is to learn and refine a sophisticated underlying distribution of trajectories through cascading a series of simple denoising steps. To implement this, a diffusion model performs a forward diffusion process to intentionally add a series of noises to a ground-truth future trajectory; and then, it uses a conditional denoising process to recover the future trajectory from noise inputs conditioned on past trajectories.

Mathematically, let  $\mathbf{X}$  and  $\mathbb{X}_{\mathcal{N}}$  be the past trajectories of the ego agent and the neighboring agents, respectively, and  $\mathbf{Y}$  be the future trajectory of the ego agent. The diffusion model for trajectory prediction works as follows,

$$\mathbf{Y}^0 = \mathbf{Y}, \quad (2a)$$

$$\mathbf{Y}^\gamma = f_{\text{diffuse}}(\mathbf{Y}^{\gamma-1}), \quad \gamma = 1, \dots, \Gamma, \quad (2b)$$

$$\hat{\mathbf{Y}}_k^\Gamma \stackrel{i.i.d.}{\sim} \mathcal{P}(\hat{\mathbf{Y}}^\Gamma) = \mathcal{N}(\hat{\mathbf{Y}}^\Gamma; \mathbf{0}, \mathbf{I}), \text{ sample } K \text{ times}, \quad (2c)$$

$$\hat{\mathbf{Y}}_k^\gamma = f_{\text{denoise}}(\hat{\mathbf{Y}}_k^{\gamma+1}, \mathbf{X}, \mathbb{X}_{\mathcal{N}}), \quad \gamma = \Gamma-1, \dots, 0, \quad (2d)$$

where  $\mathbf{Y}^\gamma$  is the noisy trajectory at the  $\gamma$ th diffusion step and  $\hat{\mathbf{Y}}_k^\gamma$  is the  $k$ th sample of denoised trajectory at the  $\gamma$ th denoising step. The final  $K$  predicted trajectories are  $\hat{\mathcal{Y}} = \{\hat{\mathbf{Y}}_1^0, \hat{\mathbf{Y}}_2^0, \dots, \hat{\mathbf{Y}}_K^0\}$ .

Step (2a) initializes the diffused trajectory; Step (2b) uses a forward diffusion operation  $f_{\text{diffuse}}(\cdot)$  to successively add noises to  $\mathbf{Y}^{\gamma-1}$  and obtain the diffused trajectory  $\mathbf{Y}^\gamma$ ; Step (2c) draws  $K$  independent and identically distributed samples to initialize denoised trajectories  $\hat{\mathbf{Y}}_k^\Gamma$  from a normal distribution; and Step (2d) iteratively applies a denoising operation  $f_{\text{denoise}}(\cdot)$  to obtain the denoised trajectory  $\hat{\mathbf{Y}}_k^\gamma$  conditioned on past trajectories  $\mathbf{X}, \mathbb{X}_{\mathcal{N}}$ . Note that i) Steps (2a) and (2b) correspond to the forward diffusion process and are not used in inference; ii) During training,  $\mathbf{Y}^\gamma$  is naturally the supervision for  $\hat{\mathbf{Y}}_k^\gamma$  at the  $\gamma$ th step. Conceptually, each denoising step is the reverse of the diffusion step, and each pair of  $\mathbf{Y}^\gamma$  and  $\hat{\mathbf{Y}}_k^\gamma$  shares the same underlying distribution.

The standard diffusion model is expressively powerful in learning sophisticated distributions and has achieved great success in many generation tasks. However, the task of motion prediction requires real-time inference but the running time of a diffusion model is constrained by the large number of denoising steps. Meanwhile, less denoising steps usually cause a weaker representation ability of future distributions. To achieve higher efficiency while preserving a promising representation ability, we propose leapfrog diffusion model, which uses a trainable initializer to capture sophisticated distributions and substitute a large number of denoising steps.

### 4. Leapfrog Diffusion Model

#### 4.1. System Architecture

In this section, we propose the leapfrog diffusion model. Here leapfrog means that a large number of small denoising steps can be replaced by a single, yet powerful leapfrog initializer, which can significantly accelerate the inference speed without losing representation ability. Let  $\mathbf{X}$  and  $\mathbb{X}_{\mathcal{N}}$  be the past trajectories of the ego agent and its neighboring agents, and  $\mathbf{Y}$  be the future trajectory of the ego agent. Denote  $\tau$  as the leapfrog step. The overall procedure of the proposed leapfrog diffusion model is formulated as follows,

$$\mathbf{Y}^0 = \mathbf{Y}, \quad (3a)$$

$$\mathbf{Y}^\gamma = f_{\text{diffuse}}(\mathbf{Y}^{\gamma-1}), \quad \gamma = 1, \dots, \Gamma, \quad (3b)$$

$$\hat{\mathcal{Y}}^\tau \stackrel{K}{\sim} \mathcal{P}(\hat{\mathcal{Y}}^\tau) = f_{\text{LSG}}(\mathbf{X}, \mathbb{X}_{\mathcal{N}}), \quad (3c)$$

$$\hat{\mathbf{Y}}_k^\gamma = f_{\text{denoise}}(\hat{\mathbf{Y}}_k^{\gamma+1}, \mathbf{X}, \mathbb{X}_{\mathcal{N}}), \quad \gamma = \tau-1, \dots, 0. \quad (3d)$$

Compared to the standard diffusion model (2), the main difference lies in Step (3c). The standard diffusion initializes the  $\Gamma$ th denoised distribution  $\mathcal{P}(\hat{\mathbf{Y}}^\Gamma)$  by a plain normal distribution (2c) and requires a lot of denoising steps to enrich the expressiveness of the denoised distribution; while in Step (3c), we propose a novel leapfrog initial-

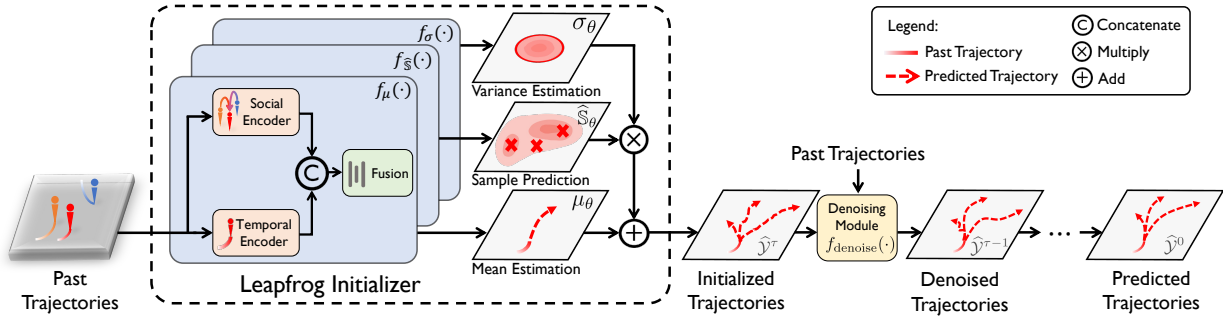


Figure 2. Proposed leapfrog diffusion model (LED) in inference phase. The red agent is the to-be-predicted agent. LED first predicts  $K$  initialized trajectories at  $\tau$ th denoised step through a trainable leapfrog initializer. Then, followed by a few denoising steps, LED obtains the final predictions. In leapfrog initializer, LED learns statistics and generates correlated samples with the reparameterization.

izer  $f_{\text{LSG}}(\cdot)$  to directly model the  $\tau$ th denoised distribution  $\mathcal{P}(\hat{\mathbf{Y}}^\tau)$ , which is hypothetically equivalent to the output of executing  $(\Gamma - \tau)$  denoising steps (2d). We then draw samples from the distribution  $\mathcal{P}(\hat{\mathbf{Y}}^\tau)$  and obtain  $K$  future trajectories  $\hat{\mathbf{Y}}^\tau = \{\hat{\mathbf{Y}}_1^\tau, \hat{\mathbf{Y}}_2^\tau, \dots, \hat{\mathbf{Y}}_K^\tau\}$ , where  $\hat{\mathbf{Y}}^\tau$  in (3d) means  $K$  samples are dependent to intentionally allocate appropriate sample diversity. Then, in Step (3d), we only need to apply the remaining  $\tau$  denoising steps for each trajectory  $\hat{\mathbf{Y}}_k^\tau$  to obtain the final prediction  $\hat{\mathbf{Y}} = \{\hat{\mathbf{Y}}_1^0, \hat{\mathbf{Y}}_2^0, \dots, \hat{\mathbf{Y}}_K^0\}$ .

Note that i) the proposed leapfrog diffusion model reduces the denoising steps from  $\Gamma$  to  $\tau (\ll \Gamma)$  in Step (3d) as the leapfrog initializer directly provides the trajectories at denoising step  $\tau$ , accelerating the inference; ii) instead of taking independent and identically distributed samples in Step (2c), the proposed leapfrog initializer generates  $K$  trajectories  $\hat{\mathbf{Y}}^\tau$  simultaneously in Step (3c), allowing  $K$  samples to be aware of each other; and iii) the standard diffusion model and the proposed leapfrog diffusion model share the same forward diffusion process, assuring that the representation capacity is not reduced.

## 4.2. Leapfrog Initializer

We now dive into the design details of the proposed leapfrog initializer, which leapfrogs  $(\Gamma - \tau)$  denoising steps. In leapfrog initializer, we model the  $\tau$ th denoised distribution  $\mathcal{P}(\hat{\mathbf{Y}}^\tau)$  through learning models. However, it is nontrivial for a learning model to directly capture the sophisticated distribution, which usually causes unstable training. To ease the learning burden of the model, we disassemble the distribution  $\mathcal{P}(\hat{\mathbf{Y}}^\tau)$  into three representative parts: the mean, global variance and sample prediction. For each part, we design trainable modules correspondingly. Mathematically, let  $\mathbf{X}$  and  $\mathbb{X}_\mathcal{N}$  be the past trajectories of the ego agent and the neighboring agents, respectively. The proposed leapfrog initializer generates  $K$  samples as follows,

$$\begin{aligned} \mu_\theta &= f_\mu(\mathbf{X}, \mathbb{X}_\mathcal{N}) \in \mathbb{R}^{T_f \times 2}, \\ \sigma_\theta &= f_\sigma(\mathbf{X}, \mathbb{X}_\mathcal{N}) \in \mathbb{R}, \\ \hat{\mathbf{S}}_\theta &= [\hat{\mathbf{S}}_{\theta,1}, \dots, \hat{\mathbf{S}}_{\theta,K}] = f_{\hat{\mathbf{S}}}(\mathbf{X}, \mathbb{X}_\mathcal{N}, \sigma_\theta) \in \mathbb{R}^{T_f \times 2 \times K}, \\ \hat{\mathbf{Y}}_k^\tau &= \mu_\theta + \sigma_\theta \cdot \hat{\mathbf{S}}_{\theta,k} \in \mathbb{R}^{T_f \times 2}, \end{aligned} \quad (4)$$

where  $f_\mu(\cdot)$ ,  $f_\sigma(\cdot)$ ,  $f_{\hat{\mathbf{S}}}(\cdot)$  are three trainable modules,  $\mu_\theta$ ,  $\sigma_\theta$  are the mean and standard deviation of  $\mathcal{P}(\hat{\mathbf{Y}}^\tau)$ , respectively, and  $\hat{\mathbf{S}}_{\theta,k}$  is the normalized positions for the  $k$ th sample.

To be specific, the mean estimate module  $f_\mu(\cdot)$  infers the mean trajectory of the  $\tau$ th denoised distribution  $\hat{\mathcal{P}}(\hat{\mathbf{Y}}^\tau)$  with past trajectories  $(\mathbf{X}, \mathbb{X}_\mathcal{N})$ . The mean trajectory  $\mu_\theta$  is shared across all the  $K$  samples. The variance estimate module  $f_\sigma(\cdot)$  infers the standard deviation of the  $\tau$ th denoised distribution  $\hat{\mathcal{P}}(\hat{\mathbf{Y}}^\tau)$ , reflecting the overall uncertainty of the trajectory, which is also shared across all the  $K$  samples. The sample prediction module  $f_{\hat{\mathbf{S}}}(\cdot)$  takes the past trajectories  $(\mathbf{X}, \mathbb{X}_\mathcal{N})$  and the predicted uncertainty  $\sigma_\theta$  as the input and predicts  $K$  normalized positions where each  $\hat{\mathbf{S}}_{\theta,k} \in \mathbb{R}^{T_f \times 2}$ .

Note that i) the reparameterization in Eq. (4) allows us to avoid learning a raw sophisticated distribution, making the training much easier; and ii)  $K$  normalized predictions are generated simultaneously from the same underlying feature, assuring appropriately allocated trajectories with variance estimation and better capturing the multi-modalities.

To implement the three trainable modules:  $f_\mu(\cdot)$ ,  $f_\sigma(\cdot)$ ,  $f_{\hat{\mathbf{S}}}(\cdot)$ , we consider a similar network design: a social encoder to capture social influence, a temporal encoder to learn temporal embedding, and an aggregation layer to fuse both social and temporal information; see Figure 2. Here we take the mean estimation module  $f_\mu(\cdot)$  as an example. The mean trajectory is obtained as follows,

$$\mathbf{e}_{\mu_\theta}^{\text{social}} = \text{softmax} \left( \frac{f_q(\mathbf{X}) f_k(\mathbb{X}_\mathcal{N})^\top}{\sqrt{d}} \right) f_v(\mathbb{X}_\mathcal{N}), \quad (5a)$$

$$\mathbf{e}_{\mu_\theta}^{\text{temp}} = f_{\text{GRU}}(f_{\text{conv1D}}(\mathbf{X})), \quad (5b)$$

$$\mu_\theta = f_{\text{fusion}}([\mathbf{e}_{\mu_\theta}^{\text{social}}; \mathbf{e}_{\mu_\theta}^{\text{temp}}]). \quad (5c)$$

Step (5a) obtains the social embedding  $\mathbf{e}_{\mu_\theta}^{\text{social}}$  based on the multi-head attention with  $d$  the embedding dimension and  $f_q(\cdot)$ ,  $f_k(\cdot)$ ,  $f_v(\cdot)$  the query/key/value embedding functions. Step (5b) obtains the temporal embedding through the feature encoder  $f_{\text{conv1D}}(\cdot)$ , mapping the raw coordinates into the high-dimensional feature, followed by the gated recurrent units  $f_{\text{GRU}}(\cdot)$ , capturing the temporal dependence in the high dimensional sequence. Step (5c) concatenates both social and temporal embeddings and uses a multi-layer perceptron  $f_{\text{fusion}}(\cdot)$  to obtain the final mean estimation. Note



that the sample prediction module  $f_{\hat{S}}(\cdot)$  also takes the estimated standard deviation as the input, working as

$$\begin{aligned} \mathbf{e}_{\hat{S}_\theta}^\sigma &= f_{\text{encode}}(\sigma_\theta), \\ \hat{S}_\theta &= f_{\text{fusion}}([\mathbf{e}_{\hat{S}_\theta}^{\text{social}} : \mathbf{e}_{\hat{S}_\theta}^{\text{temp}} : \mathbf{e}_{\hat{S}_\theta}^\sigma]), \end{aligned}$$

where an encoder  $f_{\text{encode}}(\cdot)$  operates on the estimated variance  $\sigma_\theta$  and generates high dimensional embedding  $\mathbf{e}_{\hat{S}_\theta}^\sigma$ . By this, the variance estimation also involves in the sample prediction process, instead of just scaling these prediction.

After obtaining  $K$  samples  $\hat{\mathbf{Y}}^\tau = \{\hat{\mathbf{Y}}_1^\tau, \hat{\mathbf{Y}}_2^\tau, \dots, \hat{\mathbf{Y}}_K^\tau\}$  from leapfrog initializer, we execute the remaining  $\tau$  denoising steps to iteratively refine those predicted trajectories (3d).

### 4.3. Denoising Module

Here we elaborate the design of a denoising module  $f_{\text{denoise}}(\cdot)$ , which denoises the trajectory  $\hat{\mathbf{Y}}_k^{\gamma+1}$  conditioned on past trajectories  $(\mathbf{X}, \mathbb{X}_N)$ . In a denoising module, two parts are trainable: a transformer-based context encoder  $f_{\text{context}}(\cdot)$  to learn a social-temporal embedding and a noise estimation module  $f_\epsilon(\cdot)$  to estimate the noise to reduce. Mathematically, the  $\gamma$ th denoising step works as follows,

$$\mathbf{C} = f_{\text{context}}(\mathbf{X}, \mathbb{X}_N), \quad (6a)$$

$$\epsilon_\theta^\gamma = f_\epsilon(\hat{\mathbf{Y}}_k^{\gamma+1}, \mathbf{C}, \gamma + 1), \quad (6b)$$

$$\hat{\mathbf{Y}}_k^\gamma = \frac{1}{\sqrt{\alpha_\gamma}}(\hat{\mathbf{Y}}_k^{\gamma+1} - \frac{1 - \alpha_\gamma}{\sqrt{1 - \bar{\alpha}_\gamma}}\epsilon_\theta^\gamma) + \sqrt{1 - \alpha_\gamma}\mathbf{z}, \quad (6c)$$

where  $\alpha_\gamma$  and  $\bar{\alpha}_\gamma = \prod_{i=1}^\gamma \alpha_i$  are parameters in the diffusion process and  $\mathbf{z} \sim \mathcal{N}(\mathbf{z}; \mathbf{0}, \mathbf{I})$  is a noise. Step (6a) uses a context encoder  $f_{\text{context}}(\cdot)$  on past trajectories  $(\mathbf{X}, \mathbb{X}_N)$  to obtain the context condition  $\mathbf{C}$ , which shares a similar structure to mean estimation module  $f_\mu(\cdot)$ ; Step (6b) estimates the noise  $\epsilon_\theta^\gamma$  in the noisy trajectory  $\hat{\mathbf{Y}}_k^{\gamma+1}$  through noise estimation  $f_\epsilon(\cdot)$  implemented by multi-layer perceptions with the context  $\mathbf{C}$ ; Step (6c) provides a standard denoising step [18]; see more details in the supplementary material.

### 4.4. Training Objective

To train a leapfrog diffusion model, we consider a two-stage training strategy, where the first stage trains a denoising module and the second stage focuses on a leapfrog initializer. The reason to use two stages is because the training of leapfrog initializer is more stable given fixed distribution  $\mathcal{P}(\hat{\mathbf{Y}}^\tau)$ , avoiding non-convergent training.

Concretely, the first stage trains a denoising module  $f_{\text{denoise}}(\cdot)$  in Step (3d) based on a standard training schedule of a diffusion model [15, 18] through noise estimation loss:

$$\mathcal{L}_{\text{NE}} = \|\epsilon - f_\epsilon(\mathbf{Y}^{\gamma+1}, f_{\text{context}}(\mathbf{X}, \mathbb{X}_N), \gamma + 1)\|_2,$$

where  $\gamma \sim \mathcal{U}\{1, 2, \dots, \Gamma\}$ ,  $\epsilon \sim \mathcal{N}(\epsilon; \mathbf{0}, \mathbf{I})$  and the diffused trajectory  $\mathbf{Y}^{\gamma+1} = \sqrt{\bar{\alpha}_\gamma} \mathbf{Y}^0 + \sqrt{1 - \bar{\alpha}_\gamma} \epsilon$ . We then back-

---

### Algorithm 1 Leapfrog Diffusion Model in Inference

---

**Input:** Observed trajectories  $\mathbf{X}, \mathbb{X}_N$ , Leapfrog step  $\tau$

**Output:** Predicted trajectories  $\hat{\mathbf{Y}}$

- 1:  $\mu_\theta = f_\mu(\mathbf{X}, \mathbb{X}_N)$  ▷ Mean estimation
  - 2:  $\sigma_\theta = f_\sigma(\mathbf{X}, \mathbb{X}_N)$  ▷ Variance estimation
  - 3:  $\hat{S}_\theta = f_{\hat{S}}(\mathbf{X}, \mathbb{X}_N, \sigma_\theta)$  ▷ Sample prediction
  - 4:  $\hat{\mathbf{Y}}_k^\tau = \mu_\theta + \sigma_\theta \cdot \hat{S}_{\theta,k}, k = 1, \dots, K$  ▷ Reparameterization
  - 5: **for**  $\gamma = \tau - 1, \dots, 0$  **do**
  - 6:    $\hat{\mathbf{Y}}_k^\gamma = f_{\text{denoise}}(\hat{\mathbf{Y}}_k^{\gamma+1}, \mathbf{X}, \mathbb{X}_N)$  ▷ Denoising step
  - 7: **end for**
  - 8:  $\hat{\mathbf{Y}} = \hat{\mathbf{Y}}^0 = \{\hat{\mathbf{Y}}_1^0, \dots, \hat{\mathbf{Y}}_K^0\}$
  - 9: **return**  $\hat{\mathbf{Y}}$
- 

propagate this loss and train the parameters in the context encoder  $f_{\text{context}}(\cdot)$  and the noise estimation module  $f_\epsilon(\cdot)$ .

In the second stage, we optimize a leapfrog diffusion model with a trainable leapfrog initializer and frozen denoising modules. For each sample, the loss function is

$$\begin{aligned} \mathcal{L} &= \mathcal{L}_{\text{distance}} + \mathcal{L}_{\text{uncertainty}} \\ &= w \cdot \min_k \|\mathbf{Y} - \hat{\mathbf{Y}}_k\|_2 + \left( \frac{\sum_k \|\mathbf{Y} - \hat{\mathbf{Y}}_k\|_2}{\sigma_\theta^2 K} + \log \sigma_\theta^2 \right), \end{aligned}$$

where  $w \in \mathbb{R}$  is a hyperparameter weight. The first term constrains the minimum distance in  $K$  predictions. Intuitively, if a leapfrog initializer generates high-quality estimations for distribution  $\mathcal{P}(\hat{\mathbf{Y}}^\tau)$ , then one of the  $K$  predictions in  $\hat{\mathbf{Y}}$  should be close to the ground-truth trajectory  $\mathbf{Y}$ . The second term normalizes the variance estimation  $\sigma_\theta$  in reparameterization (4) through an uncertainty loss, balancing the prediction diversity and mean accuracy. Note that the variance estimation controls the dispersion of the predictions, bridging scenery complexity and prediction diversity. The first part  $\frac{\sum_k \|\mathbf{Y} - \hat{\mathbf{Y}}_k\|_2}{\sigma_\theta^2 K}$  makes the value of  $\sigma_\theta$  proportional to the complexity of the scenario. The second part  $\log \sigma_\theta^2$  is a regulariser used to avoid a trivial solution for  $\sigma_\theta$ , i.e., generating high variance for all predictions.

Technically, we can also explicitly supervise the estimation of leapfrog initializer in stage two, since the distribution  $\mathcal{P}(\hat{\mathbf{Y}}^\tau)$  can be denoised from a normal distribution. For the explicit supervision, we draw  $M \gg K$  samples from  $\mathcal{P}(\hat{\mathbf{Y}}^\Gamma)$  under the normal distribution and iteratively denoise these samples through Step (2d) until we get expected denoised trajectories  $\hat{\mathbf{Y}}^\tau$ . And then, we calculate the statistics of the denoised distribution  $\mathcal{P}(\hat{\mathbf{Y}}^\tau)$  using these  $M$  samples, serving as explicit supervisions for mean estimation  $f_\mu(\cdot)$  and variance estimation  $f_\sigma(\cdot)$ . However, since  $\tau \ll \Gamma$ , we need to run  $(\Gamma - \tau) \approx \Gamma$ -steps denoising for  $M \gg K$  samples to get statistics, resulting in unacceptable time and storage consumption for training (e.g.  $\sim 6$  days per epoch on NBA

Table 1. Comparison with baseline models on NBA dataset. minADE<sub>20</sub>/minFDE<sub>20</sub> (meters) are reported. **Bold/underlined** fonts represent the best/second-best result. Compared to the previous SOTA method, MID, our method achieves a 15.6%/13.4% ADE/FDE improvement.

Time	Social-GAN [16] CVPR'18	STGAT [20] ICCV'19	Social-STGCNN [32] CVPR'20	PECNet [28] ECCV'20	STAR [53] ECCV'20	Trajectron++ [39] ECCV'20	MemoNet [51] CVPR'22	NPSN [2] CVPR'22	GroupNet [50] CVPR'22	MID [15] CVPR'22	Ours
1.0s	0.41/0.62	0.35/0.51	0.34/0.48	0.40/0.71	0.43/0.66	0.30/0.38	0.38/0.56	0.35/0.58	<u>0.26/0.34</u>	0.28/0.37	<b>0.18/0.27</b>
2.0s	0.81/1.32	0.73/1.10	0.71/0.94	0.83/1.61	0.75/1.24	0.59/0.82	0.71/1.14	0.68/1.23	<u>0.49/0.70</u>	0.51/0.72	<b>0.37/0.56</b>
3.0s	1.19/1.94	1.04/1.75	1.09/1.77	1.27/2.44	1.03/1.51	0.85/1.24	1.00/1.57	1.01/1.76	0.73/1.02	<u>0.71/0.98</u>	<b>0.58/0.84</b>
Total(4.0s)	1.59/2.41	1.40/2.18	1.53/2.26	1.69/2.95	1.13/2.01	1.15/1.57	1.25/1.47	1.31/1.79	<u>0.96/1.30</u>	<u>0.96/1.27</u>	<b>0.81/1.10</b>

Table 2. Comparison with baseline models on NFL dataset. minADE<sub>20</sub>/minFDE<sub>20</sub> (meters) are reported. **Bold/underlined** fonts represent the best/second-best result. Compared to the previous SOTA method, MID, our method achieves a 23.7%/21.9% improvement.

Time	Social-GAN [16] CVPR'18	STGAT [20] ICCV'19	Social-STGCNN [32] CVPR'20	PECNet [28] ECCV'20	STAR [53] ECCV'20	Trajectron++ [39] ECCV'20	LB-EBM [35] CVPR'21	NPSN [2] CVPR'22	GroupNet [50] CVPR'22	MID [15] CVPR'22	Ours
1.0s	0.37/0.68	0.35/0.64	0.45/0.64	0.52/0.97	0.49/0.84	0.41/0.65	0.75/1.05	0.43/0.64	0.32/0.57	<u>0.30/0.58</u>	<b>0.21/0.34</b>
2.0s	0.83/1.53	0.82/1.60	1.06/1.87	1.19/2.47	1.02/1.84	0.93/1.65	1.26/2.28	0.83/1.52	0.73/1.39	<u>0.71/1.31</u>	<b>0.49/0.91</b>
Total(3.2s)	1.44/2.51	1.39/2.48	1.82/3.18	1.99/3.84	1.51/2.97	1.54/2.58	1.90/3.25	1.32/2.27	1.21/2.15	<u>1.14/1.92</u>	<b>0.87/1.50</b>

dataset). We thus do not use explicit supervision.

#### 4.5. Inference Phase

During the inference, instead of the  $\Gamma$ -steps' denoising, leapfrog diffusion model only takes  $\tau$ -steps, accelerating the inference. To be specific, we first generate  $K$  correlated samples to model the distribution  $\mathcal{P}(\hat{\mathbf{Y}}^\tau)$  using the trained leapfrog initializer. Then, these samples will be fed into the denoising process and iteratively fine-tuned to produce the final predictions; see Algorithm 1.

### 5. Experiments

#### 5.1. Datasets

We evaluate our method on four trajectory prediction datasets, including two sports datasets (NBA SportVU Dataset, NFL Football Dataset) and two pedestrian datasets (Stanford Drone Dataset, ETH-UCY).

**NBA SportVU Dataset (NBA):** NBA trajectory dataset is collected by NBA using the SportVU tracking system, which records the trajectories of the 10 players and the ball in real basketball games. In this task, we predict the future 4.0s (20 frames) using the 2.0s (10 frames) past trajectory.

**NFL Football Dataset (NFL):** NFL Football Dataset records the position of every player on the field during each play in the 2017 year. We predict the 22 players' (11 players per team) and the ball's future 3.2s (16 frames) trajectory using the historical 1.6s (8 frames) trajectory.

**Stanford Drone Dataset (SDD):** SDD is a large-scale pedestrian dataset collected from a university campus in bird's eye view. Following previous works [28, 51], we use the standard train-test split and predict the future 4.8s (12 frames) using 3.2s (8 frames) past.

**ETH-UCY:** ETH-UCY dataset contains 5 subsets: ETH, HOTEL, UNIV, ZARA1, and ZARA2, containing various motion scenes. We use same segment length of 8s as SDD following previous works [19, 28] and use the leave-one-out approach with four sets for training and a left set for testing.

#### 5.2. Implementation Details

In the leapfrog diffusion model, we set the diffusion step  $\Gamma = 100$  for all four datasets and the leapfrog step  $\tau = 5$  on the NBA dataset. In the leapfrog initializer, we build a transformer-based social encoder where the feed-forward dimension is set to 256, the number of heads is 2, and 2 encoder layers are applied; we apply the temporal encoder with 1D convolution kernel being 3, and output channel setting to 32, and we also build a GRU with the hidden size of 256. In the denoising module, we apply the same parameters transformer to extract the context information, and we build the core denoising module with a hidden size of 256. To train the leapfrog diffusion model, we train the denoising module for 100 epochs with an initial learning rate of  $10^{-2}$  and decay to half every 16 epochs. With a frozen denoising module, we then train the leapfrog initializer for 200 epochs with an initial learning rate of  $10^{-4}$ , decaying by 0.9 every 32 epochs. We set weight parameter  $w_1 = 50$  to emphasize the distance loss. The entire framework is trained with the Adam optimizer on one GTX-3090 GPU. All models are implemented with PyTorch 1.7.1. See more details in the supplementary material.

#### 5.3. Comparison with SOTA Methods

We measure the performance of different trajectory prediction methods using two metrics: minADE<sub>K</sub> and minFDE<sub>K</sub>, following previous work [28, 50]. 1) minADE<sub>K</sub> calculates the minimum time-averaged distance among  $K$  predictions and the ground-truth future trajectory; 2) minFDE<sub>K</sub> measures the minimum distance among the  $K$  predicted endpoints and the ground-truth endpoints. We calculate these two metrics at different timestamps on sports datasets to better evaluate the performance.

**NBA dataset.** We compare our method with the current 10 state-of-the-art prediction methods at different timestamps; see Table 1. We see that i) our method significantly outperforms all baselines in ADE and FDE at all timestamps. Our method reduces the ADE/FDE at 4.0s from 0.96/1.27

Table 3. Comparison with baseline models on SDD dataset. minADE<sub>20</sub>/minFDE<sub>20</sub> (meters) are reported. **Bold/underlined** fonts represent the best/second-best result. Our method achieves the best performance in ADE/FDE. \* represents the reproduced results from open source.

Time	Social-GAN [16] CVPR'18	SOPHIE [37] CVPR'19	Trajectron++ [39] ECCV'20	NMMP [19] CVPR'20	Evolve-Graph [25] NIPS'20	PECNet [28] ECCV'20	MemoNet [51] CVPR'22	NPSN [2] CVPR'22	GroupNet [50] CVPR'22	MID* [15] CVPR'22	Ours
4.8s	27.23/41.44	16.27/29.38	19.30/32.70	14.67/26.72	13.90/22.90	9.96/15.88	<u>8.56/12.66</u>	<u>8.56/11.85</u>	9.31/16.11	9.73/15.32	<b>8.48/11.66</b>

Table 4. Comparison with baseline models on ETH-UCY dataset. minADE<sub>20</sub>/minFDE<sub>20</sub> (meters) are reported. **Bold/underlined** fonts represent the best/second-best result. In most subsets, our method achieves the best or second-best performance in ADE/FDE.

Subset	Social-GAN [16] CVPR'18	NMMP [19] CVPR'20	STAR [53] ECCV'20	PECNet [28] ECCV'20	Trajectron++ [39] ECCV'20	Agentformer [54] ICCV'21	MemoNet [51] CVPR'22	NPSN [2] CVPR'22	GroupNet [50] CVPR'22	MID [15] CVPR'22	Ours
ETH	0.87/1.62	0.61/1.08	0.36/0.65	0.54/0.87	0.61/1.02	0.45/0.75	0.40/0.61	0.40/0.76	0.46/0.73	<b>0.39/0.66</b>	<b>0.39/0.58</b>
Hotel	0.67/1.37	0.33/0.63	0.17/0.36	0.18/0.24	0.19/0.28	0.14/0.22	<b>0.11/0.17</b>	0.12/0.18	0.15/0.25	0.13/0.22	<b>0.11/0.17</b>
Univ	0.76/1.52	0.52/1.11	0.31/0.62	0.35/0.60	0.30/0.54	0.25/0.45	0.24/0.43	<b>0.22/0.41</b>	0.26/0.49	<b>0.22/0.45</b>	0.26/0.43
Zara1	0.35/0.68	0.32/0.66	0.29/0.52	0.22/0.39	0.24/0.42	0.18/0.30	0.18/0.32	<b>0.17/0.31</b>	0.21/0.39	<b>0.17/0.30</b>	0.18/0.26
Zara2	0.42/0.84	0.43/0.85	0.22/0.46	0.17/0.30	0.18/0.32	0.14/0.24	0.14/0.24	<b>0.12/0.24</b>	0.17/0.33	0.13/0.27	<b>0.13/0.22</b>
AVG	0.61/1.21	0.41/0.82	0.26/0.53	0.29/0.48	0.30/0.51	0.23/0.39	<b>0.21/0.35</b>	<b>0.21/0.38</b>	0.25/0.44	<b>0.21/0.38</b>	<b>0.21/0.33</b>

Table 5. Ablation of leapfrog initializer in the leapfrog diffusion model on NFL with various prediction numbers  $K$ . Each module in the leapfrog initializer is beneficial.

Mean $\mu_\theta$	Variance $\sigma_\theta$	Sample $\hat{\mathbb{S}}_\theta$	$K=2$	$K=4$
✓		correlated	2.04±0.18/4.08±0.48	1.63±0.13/3.05±0.16
	✓	correlated	1.95±0.08/3.90±0.22	1.49±0.01/2.86±0.02
✓	✓	i.i.d	2.36±0.13/4.31±0.22	1.90±0.07/3.31±0.05
✓	✓	correlated	<b>1.84±0.05/3.61±0.11</b>	<b>1.47±0.01/2.83±0.02</b>
Mean $\mu_\theta$	Variance $\sigma_\theta$	Sample $\hat{\mathbb{S}}_\theta$	$K=8$	$K=20$
✓		correlated	1.25±0.02/2.31±0.04	0.99±0.03/1.68±0.04
	✓	correlated	1.23±0.01/2.20±0.01	0.95±0.01/1.54±0.02
✓	✓	i.i.d	1.51±0.04/2.67±0.07	1.18±0.02/1.90±0.03
✓	✓	correlated	<b>1.18±0.01/2.19±0.01</b>	<b>0.89±0.01/1.51±0.02</b>

to 0.81/1.10 compared to the current state-of-the-art methods, MID, achieving 15.6%/13.4% improvement; and ii) performance improvement over previous methods increases with timestamps, reflecting the proposed method can capture more sophisticated distributions at further timestamps.

**NFL dataset.** We compare our method with the current 10 state-of-the-art prediction methods at different timestamps; see Table 2. We see that our model significantly outperforms all baselines in ADE and FDE at all timestamps. Our method reduces the ADE/FDE at 3.2s from 1.14/1.92 to 0.87/1.50 compared to the current state-of-the-art methods, MID, achieving 23.7%/21.9% improvement.

**SDD dataset.** We compare our method with the current 10 state-of-the-art prediction methods; see Table 3. We see that our method reduces FDE from 11.85 to 11.66 compared to the current state-of-the-art method, NPSN. Notably, the original MID [15] uses a different protocol from all the other methods, we update its code for a fair comparison.

**ETH-UCY dataset.** We compare our method with 10 state-of-the-art prediction methods; see Table 4. We see that i) our method reduces FDE from 0.35 to 0.33 compared to the current state-of-the-art method, MemoNet, achieving a 5.7% improvement; and ii) our method achieves the best or second best to the best performance on most of the subsets.

Table 6. Different steps  $\Gamma/\tau$  in the standard/leapfrog diffusion model on NBA.  $\tau = 5$  provides the best performance.

Method	Steps	1.0s	2.0s	3.0s	Total(4.0s)	Inference (ms)
Standard Diffusion ( $\Gamma$ )	10	0.45/0.51	0.98/1.55	1.62/2.56	2.21/2.77	~87
	50	0.26/0.36	0.56/0.91	0.89/1.42	1.21/1.73	~446
	100	0.21/0.28	0.44/0.64	0.69/0.95	0.94/1.21	~886
	200	0.21/0.29	0.44/0.65	0.69/0.97	0.94/1.21	>1s
	500	0.21/0.30	0.45/0.68	0.70/0.99	0.95/1.23	>1s
Leapfrog Diffusion ( $\tau$ )	3	0.20/0.31	0.40/0.62	0.62/0.88	0.84/1.10	~30
	5	<b>0.18/0.27</b>	<b>0.37/0.56</b>	<b>0.58/0.84</b>	<b>0.81/1.10</b>	~46
	10	<b>0.17/0.27</b>	0.37/0.58	0.59/0.85	0.82/1.08	~89

## 5.4. Ablation Studies

**Effect of components in leapfrog initializer.** We explore the effect of three key components in leapfrog initializer, including mean estimation, variance estimation, and sample prediction. Table 5 presents the results with mean and variance based on 5 experimental trials. We see that i) the leapfrog initializer achieves stable results with better performance even when prediction number  $K$  is small; and ii) the proposed mean estimation, variance estimation, and sample prediction all contribute to promoting prediction accuracy.

**Effect of leapfrog step  $\tau$ .** Table 6 reports the influence of different leapfrog steps in LED. We see that i) under similar inference time, our method significantly outperforms the standard diffusion model with better representation ability; ii) when  $\tau$  is too small, leapfrog initializer targets to learn more sophisticated distribution, causing worse prediction performance; and iii) when  $\tau$  is too large, leapfrog initializer has already captured the denoised distribution, encountering performance bottleneck and wasting inference time.

**Comparison to other fast sampling methods.** Table 7 compares the performance of our method and the other two fast sampling methods: PD [38] and DDIM [41]. We see that our method significantly outperforms two fast sampling methods under similar inference time since the proposed LED promotes the correlation between predictions.

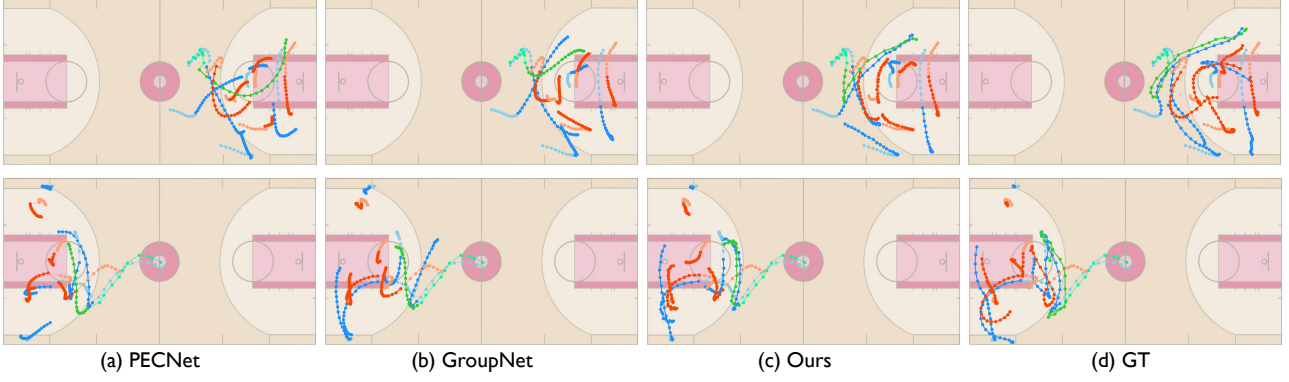


Figure 3. Visualization comparison on NBA. We compare the best-of-20 predictions by our method and two previous methods. Our method generates a more precise trajectory prediction. (Light color: past trajectory; blue/red/green color: two teams and the basketball.)

Table 7. Comparison to other fast sampling methods on NBA.  $\eta = 1$  in DDIM. Our method achieves the best performance.

Method	1.0s	2.0s	3.0s	Total(4.0s)	Inference (ms)
PD (K=1)	0.20/0.33	0.45/0.75	0.72/1.13	0.98/1.39	~452
PD (K=2)	0.21/0.34	0.46/0.78	0.73/1.15	0.98/1.41	~230
PD (K=3)	0.23/0.37	0.48/0.79	0.73/1.15	0.98/1.43	~121
PD (K=4)	0.25/0.38	0.50/0.80	0.75/1.16	0.99/1.44	~64
DDIM (S=2)	0.20/0.29	0.42/0.65	0.66/0.96	0.91/1.21	~530
DDIM (S=10)	0.22/0.32	0.44/0.71	0.69/1.04	0.93/1.31	~107
DDIM (S=20)	0.24/0.35	0.49/0.81	0.76/1.21	1.02/1.51	~54
<b>Ours</b>	<b>0.18/0.27</b>	<b>0.37/0.56</b>	<b>0.58/0.84</b>	<b>0.81/1.10</b>	<b>~46</b>

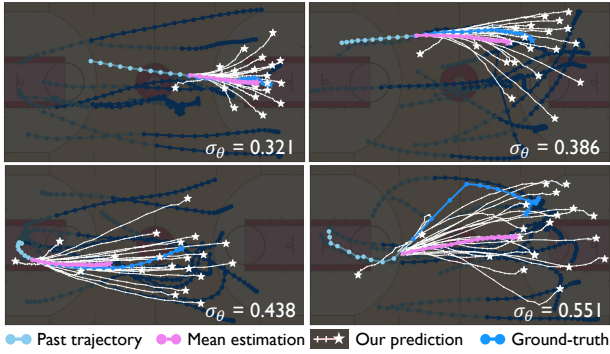


Figure 4. Mean and variance estimation in leapfrog initializer on NBA with  $K=20$ . The estimated variance can reflect the scene complexity of the current agent and produce diverse predictions.

### 5.5. Qualitative Results

**Visualization of predicted trajectory.** Figure 3 compares the predicted trajectories of two baselines PECNet and GroupNet, our LED (Ours), and the ground-truth (GT) trajectories on the NBA dataset. We see that our method produces more accurate predictions than the previous methods.

**Visualization of estimated mean and variance.** Figure 4 illustrates the mean and variance estimation in the leapfrog initializer under four scenes on the NBA dataset. We see that the variance estimation can well describe the scene complexity for the current agent by the learned variance, showing the rationality of our variance estimation.

**Visualization of different sampling mechanisms.** Figure 5 compares two sampling mechanisms: I.I.D sampling and correlated sampling in the leapfrog initializer. We see

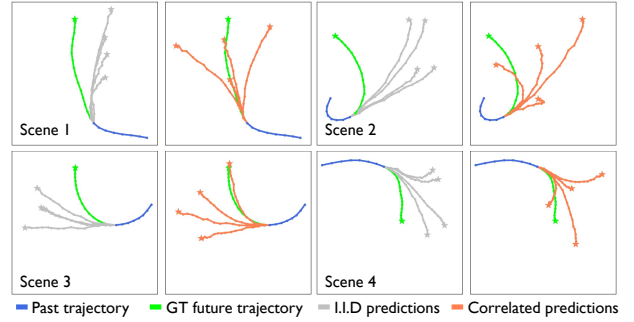


Figure 5. Comparison between I.I.D and correlated sampling mechanisms in NFL with  $K=4$ . Correlated samples appropriately capture multi-modalities, significantly improving prediction performances. that the proposed correlated sampling can appropriately allocate sample diversity and capture more modalities when the number of trials  $K$  is small.

## 6. Conclusion

This paper proposes the leapfrog diffusion model (LED), a diffusion-based trajectory prediction model, which significantly accelerates the overall inference speed and enables appropriate allocations of multiple correlated predictions. During the inference, LED directly models and samples from the denoised distribution through a novel leapfrog initializer with reparameterization. Extensive experiments show that our method achieves state-of-the-art performance on four real-world datasets and satisfies real-time inference needs.

**Limitation and future work.** This work achieves inference acceleration for trajectory prediction tasks partially because the dimension of trajectory data is relatively small and the corresponding distribution is much easier to learn compared with those of image/video data. A possible future work is to explore diffusion models and fast sampling methods for higher-dimensional tasks.

## Acknowledgements

This research is partially supported by National Natural Science Foundation of China under Grant 62171276 and the Science and Technology Commission of Shanghai Municipal under Grant 21511100900 and 22DZ2229005.



## 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. [1](#), [2](#)
- [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. [2](#), [6](#), [7](#)
- [3] Apratim Bhattacharyya, Michael Hanselmann, Mario Fritz, Bernt Schiele, and Christoph-Nikolas Straehle. Conditional flow variational autoencoders for structured sequence prediction. *arXiv preprint arXiv:1908.09008*, 2019. [1](#), [2](#)
- [4] Nanxin Chen, Yu Zhang, Heiga Zen, Ron J Weiss, Mohammad Norouzi, and William Chan. Wavegrad: Estimating gradients for waveform generation. In *International Conference on Learning Representations*, 2020. [1](#), [2](#)
- [5] Siheng Chen, Baoan Liu, Chen Feng, Carlos Vallespi-Gonzalez, and Carl Wellington. 3d point cloud processing and learning for autonomous driving: Impacting map creation, localization, and perception. *IEEE Signal Processing Magazine*, 38(1):68–86, 2020. [1](#)
- [6] Yujiao Cheng, Liting Sun, Changliu Liu, and Masayoshi Tomizuka. Towards efficient human-robot collaboration with robust plan recognition and trajectory prediction. *IEEE Robotics and Automation Letters*, 5(2):2602–2609, 2020. [1](#)
- [7] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014. [1](#)
- [8] Patrick Dendorfer, Sven Elflein, and Laura Leal-Taixé. Mgan: 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. [2](#)
- [9] Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. *Advances in Neural Information Processing Systems*, 34:8780–8794, 2021. [1](#), [2](#)
- [10] Liangji Fang, Qinhong Jiang, Jianping Shi, and Bolei Zhou. Tpnnet: Trajectory proposal network for motion prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6797–6806, 2020. [2](#)
- [11] Dario Floreano and Robert J Wood. Science, technology and the future of small autonomous drones. *Nature*, 521(7553):460–466, 2015. [1](#)
- [12] Thomas Gilles, Stefano Sabatini, Dzmitry Tsishkou, Bogdan Stanculescu, and Fabien Moutarde. Home: Heatmap output for future motion estimation. In *2021 IEEE International Intelligent Transportation Systems Conference*, pages 500–507, 2021. [1](#), [2](#)
- [13] Thomas Gilles, Stefano Sabatini, Dzmitry Tsishkou, Bogdan Stanculescu, and Fabien Moutarde. Gohome: Graph-oriented heatmap output for future motion estimation. In *International Conference on Robotics and Automation*, pages 9107–9114, 2022. [1](#), [2](#)
- [14] Francesco Giuliani, Irtiza Hasan, Marco Cristani, and Fabio Galasso. Transformer networks for trajectory forecasting. In *International Conference on Pattern Recognition*, pages 10335–10342. IEEE, 2021. [1](#)
- [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. [1](#), [2](#), [5](#), [6](#), [7](#)
- [16] 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. [1](#), [2](#), [6](#), [7](#)
- [17] Dirk Helbing and Peter Molnar. Social force model for pedestrian dynamics. *Physical review E*, 51(5):4282, 1995. [2](#)
- [18] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *Advances in Neural Information Processing Systems*, volume 33, pages 6840–6851, 2020. [1](#), [2](#), [5](#)
- [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. [1](#), [2](#), [6](#), [7](#)
- [20] Yingfan Huang, Huikun Bi, Zhaoxin Li, Tianlu Mao, and Zhaoqi Wang. Stgat: Modeling spatial-temporal interactions for human trajectory prediction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6272–6281, 2019. [6](#)
- [21] Takayuki Kanda, Hiroshi Ishiguro, Tetsuo Ono, Michita Imai, and Ryohei Nakatsu. Development and evaluation of an interactive humanoid robot "robovie". In *IEEE International Conference on Robotics and Automation*, pages 1848–1855, 2002. [1](#)
- [22] Zhifeng Kong, Wei Ping, Jiaji Huang, Kexin Zhao, and Bryan Catanzaro. Diffwave: A versatile diffusion model for audio synthesis. In *International Conference on Learning Representations*, 2020. [1](#), [2](#)
- [23] Mihee Lee, Samuel S Sohn, Seonghyeon Moon, Sejong Yoon, Mubbasir Kapadia, and Vladimir Pavlovic. Muse-vae: Multi-scale vae for environment-aware long term trajectory prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2221–2230, 2022. [2](#)
- [24] Jesse Levinson, Jake Askeland, Jan Becker, Jennifer Dolson, David Held, Soeren Kammel, J Zico Kolter, Dirk Langer, Oliver Pink, Vaughan Pratt, et al. Towards fully autonomous driving: Systems and algorithms. In *IEEE Intelligent Vehicles symposium*, pages 163–168. IEEE, 2011. [1](#)
- [25] Jiachen Li, Fan Yang, Masayoshi Tomizuka, and Chiho Choi. Evolvegraph: Multi-agent trajectory prediction with dynamic relational reasoning. In *Proceedings of the Neural Information Processing Systems*, 2020. [7](#)
- [26] Maosen Li, Siheng Chen, Xu Chen, Ya Zhang, Yanfeng Wang, and Qi Tian. Symbiotic graph neural networks for 3d skeleton-based human action recognition and motion prediction. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(6):3316–3333, 2021. [1](#)

- [27] Karttikeya Mangalam, Yang An, Harshayu Girase, and Jitendra Malik. From goals, waypoints & paths to long term human trajectory forecasting. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 15233–15242, 2021. 1, 2
- [28] Karttikeya Mangalam, Harshayu Girase, Shreyas Agarwal, Kuan-Hui Lee, Ehsan Adeli, Jitendra Malik, and Adrien Gaidon. It is not the journey but the destination: Endpoint conditioned trajectory prediction. In *European Conference on Computer Vision*, pages 759–776, 2020. 1, 2, 6, 7
- [29] Wei Mao, Miaomiao Liu, and Mathieu Salzmann. History repeats itself: Human motion prediction via motion attention. In *European Conference on Computer Vision*, pages 474–489, 2020. 2
- [30] Wei Mao, Miaomiao Liu, Mathieu Salzmann, and Hongdong Li. Learning trajectory dependencies for human motion prediction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9489–9497, 2019. 2
- [31] Ramin Mehran, Alexis Oyama, and Mubarak Shah. Abnormal crowd behavior detection using social force model. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 935–942. IEEE, 2009. 2
- [32] Abdulla Mohamed, Kun Qian, Mohamed Elhoseiny, and Christian Claudel. Social-stgcnn: A social spatio-temporal graph convolutional neural network for human trajectory prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14424–14432, 2020. 6
- [33] Jeremy Morton, Tim A Wheeler, and Mykel J Kochenderfer. Analysis of recurrent neural networks for probabilistic modeling of driver behavior. *IEEE Transactions on Intelligent Transportation Systems*, 18(5):1289–1298, 2016. 2
- [34] Alexander Quinn Nichol and Prafulla Dhariwal. Improved denoising diffusion probabilistic models. In *International Conference on Machine Learning*, pages 8162–8171, 2021. 1, 2
- [35] 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. 6
- [36] Kashif Rasul, Calvin Seward, Ingmar Schuster, and Roland Vollgraf. Autoregressive denoising diffusion models for multivariate probabilistic time series forecasting. In *International Conference on Machine Learning*, pages 8857–8868, 2021. 2
- [37] Amir Sadeghian, Vineet Kosaraju, Ali Sadeghian, Noriaki Hirose, Hamid Rezatofighi, and Silvio Savarese. Sophie: An attentive gan for predicting paths compliant to social and physical constraints. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1349–1358, 2019. 2, 7
- [38] Tim Salimans and Jonathan Ho. Progressive distillation for fast sampling of diffusion models. In *International Conference on Learning Representations*, 2021. 2, 7
- [39] Tim Salzmann, Boris Ivanovic, Punarjay Chakravarty, and Marco Pavone. Trajectron++: Dynamically-feasible trajectory forecasting with heterogeneous data. In *European Conference on Computer Vision*, pages 683–700, 2020. 1, 2, 6, 7
- [40] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International Conference on Machine Learning*, pages 2256–2265, 2015. 2
- [41] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. In *International Conference on Learning Representations*, 2020. 2, 7
- [42] Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. In *Advances in Neural Information Processing Systems*, volume 32, 2019. 2
- [43] 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. 2
- [44] Bohan Tang, Yiqi Zhong, Ulrich Neumann, Gang Wang, Ya Zhang, and Siheng Chen. Collaborative uncertainty in multi-agent trajectory forecasting. *Advances in Neural Information Processing Systems*, 34, 2021. 1
- [45] Yusuke Tashiro, Jiaming Song, Yang Song, and Stefano Ermon. Csd: Conditional score-based diffusion models for probabilistic time series imputation. In *Advances in Neural Information Processing Systems*, pages 24804–24816, 2021. 1, 2
- [46] Maria Valera and Sergio A Velastin. Intelligent distributed surveillance systems: a review. *IEE Proceedings-Vision, Image and Signal Processing*, 152(2):192–204, 2005. 1
- [47] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, 2017. 1
- [48] Anirudh Vemula, Katharina Muelling, and Jean Oh. Social attention: Modeling attention in human crowds. In *2018 IEEE international Conference on Robotics and Automation*, pages 4601–4607, 2018. 2
- [49] Pengxiang Wu, Siheng Chen, and Dimitris N Metaxas. Motionnet: Joint perception and motion prediction for autonomous driving based on bird’s eye view maps. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11385–11395, 2020. 1
- [50] 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. 1, 2, 6, 7
- [51] 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. 6, 7
- [52] Chenxin Xu, Yuxi Wei, Bohan Tang, Sheng Yin, Ya Zhang, and Siheng Chen. Dynamic-group-aware networks for multi-agent trajectory prediction with relational reasoning. *arXiv preprint arXiv:2206.13114*, 2022. 2
- [53] Cunjun Yu, Xiao Ma, Jiawei Ren, Haiyu Zhao, and Shuai Yi. Spatio-temporal graph transformer networks for pedestrian trajectory prediction. In *European Conference on Computer Vision*, pages 507–523, 2020. 6, 7

- [54] Ye Yuan, Xinshuo Weng, Yanglan Ou, and Kris M Kitani. Agentformer: Agent-aware transformers for socio-temporal multi-agent forecasting. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9813–9823, 2021. [1](#), [7](#)
- [55] Yiqi Zhong, Zhenyang Ni, Siheng Chen, and Ulrich Neumann. Aware of the history: Trajectory forecasting with the local behavior data. In *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXII*, pages 393–409. Springer, 2022. [1](#)