Fast Inference and Update of Probabilistic Density Estimation on Trajectory Prediction

Takahiro Maeda   Norimichi Ukita

Toyota Technological Institute, Japan

{sd21601, ukita}@toyota-ti.ac.jp

Abstract

Safety-critical applications such as autonomous vehicles and social robots require fast computation and accurate probability density estimation on trajectory prediction. To address both requirements, this paper presents a new normalizing flow-based trajectory prediction model named FlowChain. FlowChain is a stack of conditional continuously-indexed flows (CIFs) that are expressive and allow analytical probability density computation. This analytical computation is faster than the generative models that need additional approximations such as kernel density estimation. Moreover, FlowChain is more accurate than the Gaussian mixture-based models due to fewer assumptions on the estimated density. FlowChain also allows a rapid update of estimated probability densities. This update is achieved by adopting the newest observed position and reusing the flow transformations and its log-det-jacobians that represent the motion trend. This update is completed in less than one millisecond because this reuse greatly omits the computational cost. Experimental results showed our FlowChain achieved state-of-the-art trajectory prediction accuracy compared to previous methods. Furthermore, our FlowChain demonstrated superiority in the accuracy and speed of density estimation. Our code is available at `https://github.com/meaten/FlowChain-ICCV2023`.

1. Introduction

Human trajectory prediction is a challenging problem because human movements are not deterministic, unlike the billiard balls on a pool table. Starting from deterministic approaches [26, 49], to tackle its indeterministic nature, many stochastic approaches are proposed, such as Gaussian distribution-based [1, 42, 66], generative adversarial networks (GANs)-based [9, 10, 25, 57], variational autoencoder (VAE)-based [58, 71, 73], diffusion-based [24], and normalizing flow-based [59] methods. These methods
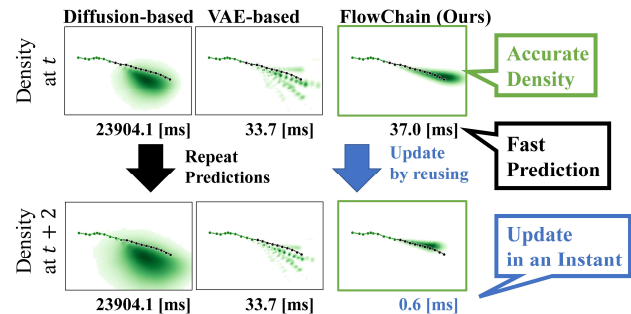

Figure 1. **Effects of our FlowChain.** Our FlowChain can rapidly estimate accurate density than VAE-based methods as shown in green. Furthermore, we propose an update of estimated density that only requires an instant of time, as shown in blue.

can be applied to human-interactive systems, including autonomous vehicles [8,11,38,55,69] and social robots [4,39].

In these safety-critical applications, autonomous systems must infer the existence probability density distribution ("density" for brevity) of other moving objects such as pedestrians around on all prediction steps for risk assessment (i.e., collision avoidance). However, most of the indeterministic approaches mentioned above merely generate possible future trajectories and cannot estimate the density alone. We need to further apply kernel density estimation (KDE) to these future trajectories for density estimation.

Thus, trajectory prediction with density estimation usually takes a larger computational cost than only generating possible trajectories. This computational cost accumulates upon every single prediction update along with highly-frequent observations. This is unacceptable to autonomous systems that work in rapidly changing environments.

To address both computational cost and density estimation, we propose a new normalizing flow-based model named FlowChain (Fig. 1). This speed and accuracy are because normalizing flows need no additional approximation such as Gaussian mixture modeling or KDE with limited samples. FlowChain can estimate a density on each prediction time step thanks to our stacked normalizing flow archi-

tecture unlike Flomo [59], which is closest to FlowChain. While Flomo uses one flow module that only estimates the spatial density of a trajectory, FlowChain further predicts its spatio-temporal density so that the spatial density of an agent position at each time step is also predicted. This stacking architecture also allows a rapid update of an estimated density in an instant of time *e.g.* less than one millisecond. This update is achieved by reusing the flow transformations and its log-det-jacobians that represent the *motion trend* such as going straight or turning right. This update with the reuse significantly omits the computational cost while maintaining the accuracy of density estimation because the *motion trend* doesn't change for several time steps.

Our contributions are three-fold as follows:

- We propose a normalizing flow-based trajectory prediction network named FlowChain that rapidly estimates the accurate density of future positions on each prediction time step.

- Our update procedure for the density requires less than one millisecond by adopting new observations and reusing the *motion trend* as the flow transformations.

- Our FlowChain achieved comparable prediction accuracy to other computationally-intense state-of-the-art methods on several benchmarks.

2. Related Work

Human Trajectory Prediction. The earliest works on human trajectory prediction [26, 49] are deterministic regression models using attractive and repulsive forces between humans and goal positions. However, human movements are indeterministic in nature. Therefore, many approaches are proposed to model this uncertainty. Social-LSTM [1] and Social-STGCNN [42] regress the parameters of the bivariate Gaussian distributions to model the uncertainty of future trajectories. These approaches [1, 42] are simple and require low computational cost but cannot catch the future multi-modality we often see at road intersections.

To address this multi-modality, many stochastic prediction models are proposed. Trajectron++ [58], Agent-Former [71], and ExpertTraj [73] combined the aforementioned Gaussian regression with VAE [32]. These models predict a bivariate Gaussian mixture based on stochastically sampled latent variables [28, 52]. We can obtain the density by summing up each estimated Gaussian density map. However, these Gaussian mixture methods have limited expressive power and thus predict inaccurate distributions.

On the other hand, GAN [18]-based approaches implicitly handle this uncertainty. Social-GAN [25], SoPhie [57], Goal-GAN [10], and MG-GAN [9] use a GAN architecture to predict diverse trajectories conditioned on random noises. While they cannot estimate the density alone, we can estimate the density by applying KDE [48, 54] on sampled trajectories

The denoising diffusion probabilistic model [27] can also implicitly handle uncertainty by generating multiple trajectories. Recently, denoising diffusion attracts much attention for its state-of-the-art performances in several generative tasks, including image synthesis [12] and audio generation [33]. MID [24] is a trajectory prediction model incorporating denoising diffusion, which achieves state-of-the-art performances. We can also estimate the density of MID by applying KDE. However, GAN-based and diffusion-based approaches cannot estimate accurate densities because KDE is performed on limited samples.

On the contrary, normalizing flow [51, 64, 65]-based trajectory prediction method named Flomo [59] can rapidly generate future trajectories and their accurate probabilities thanks to the analytical computation of normalizing flows. However, this approach only estimates the probability of each future trajectory and thus cannot estimate the density of positions on each prediction step alone.

Normalizing Flows. Thanks to its bijective process, normalizing flow can compute the density analytically. This analytical computation allows us fast and accurate density estimation because it does not need additional approximation such as KDE with limited samples. For more expressive power, several bijective families were proposed such as affine coupling layers [13, 14], autoregressive maps [17, 47], invertible linear transformations [31], and invertible ResNet block [2, 6]. All flows mentioned above have a topology constraint between the base and estimated densities. This topology constraint limits the expressive power of normalizing flows. To address this constraint, some approaches incorporated stochasticity into flows [68]. However, this stochasticity makes analytical density calculation impossible. Therefore, further density approximation such as KDE is required for this stochastic normalizing flow. Other approaches [5, 7, 44] use surjective layers, which relax the topology constraint but make analytical density computation with a little looseness. In summary, the optimal normalizing flow model architecture for future trajectory density estimation is still unknown.

Density Estimation. Kernel density estimation [48, 54] is the most representative non-parametric method that estimates a continuous density from a finite set of data. Suppose that we have A input data and B points evaluated, the computational cost is $\mathcal{O}(AB)$. This is a huge cost for autonomous systems. Therefore, several accelerations are proposed such as data binning [19, 60, 62, 67], fast sum updating [15, 16, 35, 61], fast Gauss transform [22, 23, 34], and dual-tree method [20, 21, 36]. However, we confirmed that the KDE implementation utilizing GPU is faster than these accelerations including [3, 45, 46]. We use the KDE utilizing

GPU for comparison in Sec. 4.

3. Proposed Approach

In this section, we present our normalizing flow-based trajectory prediction named FlowChain and its update of estimated density. First, we formulate the trajectory prediction task and conditional CIFs in Sec. 3.1. and Sec. 3.2 respectively. Then, we present our FlowChain model architecture in Sec. 3.3. After that, we describe its fast update procedure of estimated density in Sec.3.4. Finally, we explain how to obtain the density map in Sec. 3.5.

3.1. Problem Formulation

We aim to estimate agents' plausible future trajectory densities from their past trajectories. The trajectory of each agent is defined as a sequence $(\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_T)$ of positions $\mathbf{x}_t = (x_t, y_t)$ over discrete timesteps $t \in \{0, \ldots, T\}$. Given the current time step t, we predict future T_f-step trajectory densities $\{p(\tilde{\mathbf{x}}_{t+1}^i|\mathbf{o}_t), \ldots, p(\tilde{\mathbf{x}}_{t+T_\mathrm{f}}^i|\mathbf{o}_t)\}$ given T_o-step observed trajectory $\mathbf{o}_t = \{(\mathbf{x}_{t-T_\mathrm{o}-1}^i, \ldots, \mathbf{x}_t^i)| \quad i \in \{1, 2, \ldots, N_a\}\}$, where N_a and i denote the number of agents and the index of each agent, respectively. This index will be omitted for brevity in later use. We denote $\mathbf{x}$ as an observed position, $\tilde{\mathbf{x}}$ as a predicted future position, and $\hat{\mathbf{x}}$ as a ground truth future position.

3.2. Conditional Continuously-indexed Flow

Conditional normalizing flow is a density estimation method that uses a parameterized bijective transformation f conditioned on additional information $\mathbf{c}$ such as an encoded vector from observed trajectories, i.e., $\mathbf{o}_t$, for trajectory prediction. The bijection f transforms a base density $p(\mathbf{z})$ to an expressive density $p(\mathbf{y})$ by the analytical computation of the change-of-variables formula as follows:

$$\mathbf{y} = f(\mathbf{z}) \tag{1}$$
$$p(\mathbf{y}) = p(\mathbf{z})|\det \nabla_\mathbf{y} f^{-1}(\mathbf{y}, \mathbf{c})| \tag{2}$$

The parameters of bijective transformation f can be learned by maximizing the likelihood of samples $\hat{\mathbf{y}}$ from datasets, or minimizing the negative log-likelihood as follows:

$$NLL = -\log p(\hat{\mathbf{y}}|\mathbf{c}) \tag{3}$$

A normalizing flow with only bijective transformations has a constraint that the topology of a base density is preserved. For example, if the base density is a simple unimodal Gaussian distribution, the normalizing flow fails to fit a distinct multi-modal density. To avoid this problem of bijective transformation, we use continuously-indexed flows (CIFs) [7], akin to an infinite mixture of normalizing flows. While CIFs potentially avoid the topology constraint, CIFs still enable analytical density computation with a little

looseness $\mathcal{E}$ because CIFs incorporate inference and generative surjections according to the survey [44]. We investigated the tradeoff between the expressive power and exactness of estimated density on CIFs. We concluded this looseness of estimated density can be ignored in trajectory prediction, as we show in the supplemental materials. Based on the above discussion, We use conditional CIFs as the base normalizing flow model of our FlowChain.

3.3. FlowChain model

The overview of the FlowChain model is shown in Fig. 2. Given the current time step t, we estimate future trajectory densities $\{p(\tilde{\mathbf{x}}_{t+1}|\mathbf{o}_t), \ldots, p(\tilde{\mathbf{x}}_{t+T_\mathrm{f}}|\mathbf{o}_t)\}$ by chaining the conditional CIFs, each of which transforms the density of the former time step $p(\tilde{\mathbf{x}}_{t+n-1}|\mathbf{o}_t)$ to the density of the next time step $p(\tilde{\mathbf{x}}_{t+n}|\mathbf{o}_t)$, shown in the upper side of Fig. 2. Our formulation consists of two equations: the initial condition for the first step prediction (Eq.(4)) and the recurrence relation between the latter steps (Eq.(5)) as follows.

$$p(\tilde{\mathbf{x}}_{t+1}|\mathbf{o}_t) = \mathcal{N}(\mathbf{z}; \mathbf{x}_t, \sigma)|\det \nabla_{\tilde{\mathbf{x}}_{t+1}} f_1^{-1}(\tilde{\mathbf{x}}_{t+1}, \mathbf{o}_t)| \tag{4}$$

$$p(\tilde{\mathbf{x}}_{t+n}|\mathbf{o}_t) = p(\tilde{\mathbf{x}}_{t+n-1}|\mathbf{o}_t)|\det \nabla_{\tilde{\mathbf{x}}_{t+n}} f_n^{-1}(\tilde{\mathbf{x}}_{t+n}, \mathbf{o}_t)| \tag{5}$$

$$= \exp\left(\log \mathcal{N}(\mathbf{z}; \mathbf{x}_t, \sigma) \right.$$
$$\left. + \sum_{n'=1}^{n} \log |\det \nabla_{\tilde{\mathbf{x}}_{t+n'}} f_{n'}^{-1}(\tilde{\mathbf{x}}_{t+n'}, \mathbf{o}_t)| \right) \tag{6}$$

where $n \in \{2, 3, \ldots, T_\mathrm{f}\}$. f_n is a conditional CIF that transforms the sample $\tilde{\mathbf{x}}_{t+n-1}$ from a former density $p(\tilde{\mathbf{x}}_{t+n-1}|\mathbf{o}_t)$ to the sample $\tilde{\mathbf{x}}_{t+n}$ from a next density $p(\tilde{\mathbf{x}}_{t+n}|\mathbf{o}_t)$ as $\tilde{\mathbf{x}}_{t+n} = f_n(\tilde{\mathbf{x}}_{t+n-1}, \mathbf{o}_t)$. f_1 transforms the sample $\mathbf{z}$ from a base Gaussian density $\mathcal{N}(\cdot; \mathbf{x}_t, \sigma)$. $\mathcal{N}(\cdot; \mathbf{x}_t, \sigma)$ is created from the *newest observed position* $\mathbf{x}_t$ as a mean position. The standard deviation σ is a trainable parameter of FlowChain model. All of f_1 and f_n are independently trained with the loss function (Eq.(3)) in our method.

From f_n, we can sample multiple trajectories for best-of-N metrics [25] by sampling multiple $\mathbf{z}$ as follows:

$$\{\tilde{\mathbf{x}}_{t+1}, \ldots, \tilde{\mathbf{x}}_{t+n}\} = \{f_1(\mathbf{z}, \mathbf{o}_t), \ldots, f_n \circ \cdots \circ f_2 \circ f_1(\mathbf{z}, \mathbf{o}_t)\} \tag{7}$$

3.4. Update of estimated density

For accurate prediction by reusing the previous predictions, we focused on two features of the observed trajectory: the *newest observed position* and *motion trend*, shown in Fig. 2. Our FlowChain generates future trajectories by transforming the *newest observed position* $\mathbf{x}_t$ based on the *motion trend e.g.* go straight or turning left/right, modeled
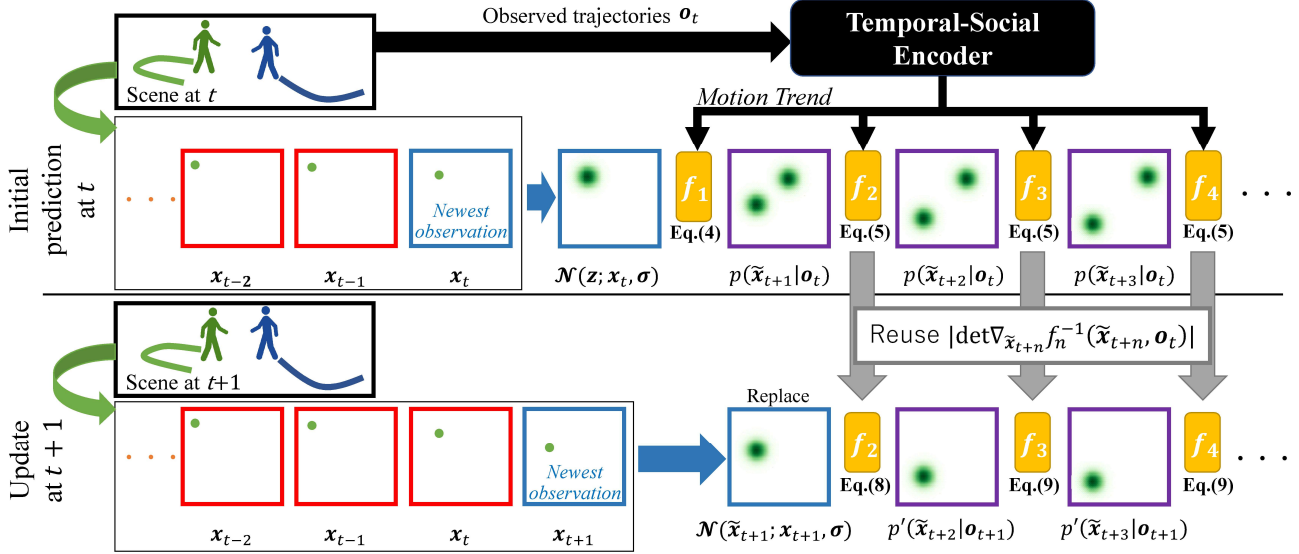
Figure 2. **Overview of our FlowChain model.** FlowChain consists of sequential conditional CIFs denoted by f_n. f_n is conditioned on the feature vector denoted by *Motion Trend* from the Temporal-Social Encoder, which encodes the observed trajectories $\mathbf{o}_t$. Since $\mathbf{o}_t$ includes the trajectories of other agents as well as an agent of interest (i.e, the target of prediction), the future trajectory is predicted while taking into account social interaction with the others. f_n transforms the density of the former time step $p(\tilde{\mathbf{x}}_{t+n-1}|\mathbf{o}_t)$ to the density of the next time step $p(\tilde{\mathbf{x}}_{t+n}|\mathbf{o}_t)$ as shown in the upper side. However, for the first step, the Gaussian distribution $\mathcal{N}(\cdot; \mathbf{x}_t, \sigma)$, which is created from the *newest observed position* $\mathbf{x}_t$, is transformed. The boxes of past, current (i.e., newest), and future densities are colored by red, blue, and purple, respectively. In the update procedure shown on the bottom side, the Gaussian distribution $\mathcal{N}(\cdot; \mathbf{x}_{n+1}, \sigma)$, created from *newest observed position* $\mathbf{x}_{t+1}$ at $t+1$, replace the estimated density from the former time step. The other calculations are reused as shown by gray arrows. We can properly update the density by using the *newest observed position* at $t+1$, $\mathbf{x}_{t+1}$, instead of $p(\tilde{\mathbf{x}}_{t+1}|\mathbf{o}_t)$. In a toy example shown in this figure, another incorrect mode observed in the predicted densities, $p(\tilde{\mathbf{x}}_{t+2}|\mathbf{o}_t)$ and $p(\tilde{\mathbf{x}}_{t+3}|\mathbf{o}_t)$, is successfully suppressed in the updated predictions $p'(\tilde{\mathbf{x}}_{t+2}|\mathbf{o}_{t+1})$ and $p'(\tilde{\mathbf{x}}_{t+3}|\mathbf{o}_{t+1})$. As shown in Sec. 4, our experiments validate this advantage of our proposed fast update procedure is greater than its disadvantage (i.e., a decrease in the prediction accuracy due to the reuse of the motion trend) so that the updated predictions are better than the initial predictions (e.g., $p'(\tilde{\mathbf{x}}_{t+3}|\mathbf{o}_{t+1})$ is better than $p(\tilde{\mathbf{x}}_{t+3}|\mathbf{o}_t)$ in the figure).

by each flow $f_n(\mathbf{x}_{t+n}, \mathbf{o}_t)$. We cannot reuse the *newest observed position* $\mathbf{x}_t$ at t for the next time step because this position is a starting point of the future trajectories and thus greatly affect the accuracy of probability density estimation. Therefore, FlowChain adopt the *newest observed position* $\mathbf{x}_{t+n}$ at $t+n$ by replacing the estimated density $p(\tilde{\mathbf{x}}_{t+n}|\mathbf{o}_t)$ with the Gaussian density $\mathcal{N}(\cdot; \mathbf{x}_{t+n}, \sigma)$, as shown in the bottom of Fig. 2. This replacement propagates the information of the *newest observed position* properly to the latter time steps.

On the other hand, the *motion trend* can be reused because it doesn't change for several time steps unless a sudden strong external force is applied to a prediction target. Therefore, our FlowChain model reuses its trend in each flow transformation $f_n(\tilde{\mathbf{x}}_{t+n}, \mathbf{o}_t)$. Specifically, we use the previously generated trajectories $\{\tilde{\mathbf{x}}_{t+1}, \ldots, \tilde{\mathbf{x}}_{t+n}\}$. Thus, we can reuse the corresponding log-det-jacobians $\log |\det \nabla_{\tilde{\mathbf{x}}_{t+n}} f_n^{-1}(\tilde{\mathbf{x}}_{t+n+1}, \mathbf{o}_t)|$ which is the variation of the estimated densities, as shown by the gray arrows in Fig. 2. The equation of FlowChain's update is expressed

similar to Eq.(4) and Eq.(5) as follows:

$$p'(\tilde{\mathbf{x}}_{t+2}|\mathbf{o}_{t+1}) = \mathcal{N}(\tilde{\mathbf{x}}_{t+1}; \mathbf{x}_{t+1}, \sigma)|\det \nabla_{\tilde{\mathbf{x}}_{t+2}} f_2^{-1}(\tilde{\mathbf{x}}_{t+2}, \mathbf{o}_t)|$$
$$(8)$$

$$p'(\tilde{\mathbf{x}}_{t+n}|\mathbf{o}_{t+1}) = p'(\tilde{\mathbf{x}}_{t+n-1}|\mathbf{o}_{t+1})|\det \nabla_{\tilde{\mathbf{x}}_{t+n}} f_n^{-1}(\tilde{\mathbf{x}}_{t+n})|$$
$$(9)$$

$$= \exp \Bigg(\log \mathcal{N}(\tilde{\mathbf{x}}_{t+1}; \mathbf{x}_{t+1}, \sigma) $$
$$+ \sum_{n'=2}^{n} \log |\det \nabla_{\tilde{\mathbf{x}}_{t+n'}} f_{n'}^{-1}(\tilde{\mathbf{x}}_{t+n'}, \mathbf{o}_t)| \Bigg)$$
$$(10)$$

where, $n \in \{3, 4, \ldots, T_f\}$. Thanks to reusing each flow transformation $f_n(\mathbf{x}_{t+n}, \mathbf{o}_t)$, we don't need to evaluate log-det-jacobians $\log |\det \nabla_{\tilde{\mathbf{x}}_{t+n}} f_n^{-1}(\tilde{\mathbf{x}}_{t+n})|$ again. This will greatly omit the computational cost. As a result, we can update the probability density by only calculating the Gaussian base distribution, summing up the log-det-jacobians, and taking exponentials, which take less than $1ms$ in total. This rapid update is beneficial, especially for autonomous
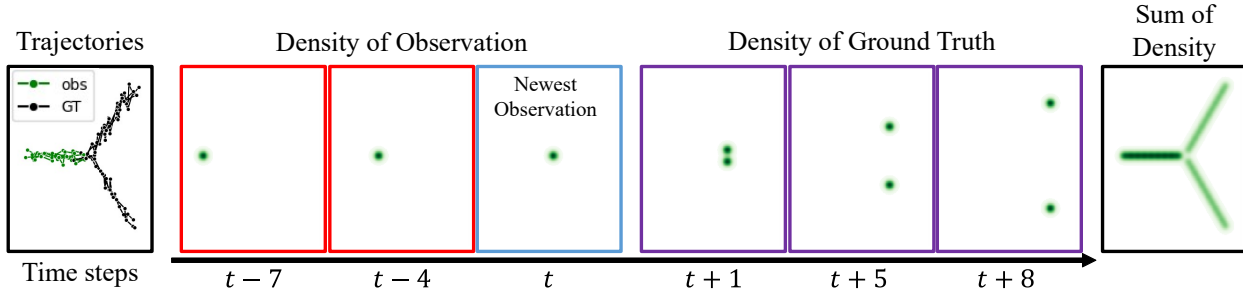
Figure 3. Overview of *Simfork* dataset. The trajectories shown in the left figure are examples of the observed and ground truth trajectories for training. The position of each step is sampled from the distributions of the middle figures. The sum of each distribution is shown in the right figure. The boxes of past, current (i.e., newest), and future densities are colored by red, blue, and purple, respectively.

systems where computational resources are limited and often become busy by executing other computationally-intense tasks such as object detection, mapping, and path planning.

3.5. How to obtain the density map

The density map, not like the mere one-point estimate, is needed when autonomous systems perform visualization of density or successive tasks such as navigation. We can correct the possible future points with estimated density by sampling multiple trajectories from Eq.(7) and its density from Eq.(6). The density map is obtained by collecting these points with density at the same time step.

4. Experiments and Results Analysis

4.1. Experimental Setup

Datasets. Experiments were conducted on various datasets including our developed dataset with synthesized data *Simfork*, and real-image benchmarks including *ETH/UCY* [37, 50] and *SDD* [53]. On all the datasets, the numbers of past observed and future predicted steps are $T_o = 8$ and $T_f = 12$, respectively. All steps have an interval of 0.4 seconds in accordance with the literature such as [25, 58].

Simfork is a synthesized dataset for comparison on the accuracy of predicted densities, which are produced as explained in Sec. 3.5. In all trajectories in this dataset, either of the two types of traveling directions (i.e., turning left or right) is observed after a straight trajectory, as shown in Fig. 3. Positions before and after the fork (colored by green and black, respectively, in the leftmost figure of Fig. 3) are used as past observed and future ground-truth positions, respectively. Each position of a trajectory is independently perturbed by the Gaussian noise, which results in the density at each time step.

Two pedestrian trajectory datasets, *ETH* [50] and *UCY* [37], are merged to *ETH/UCY*. This dataset has five scenes: ETH, HOTEL, UNIV, ZARA1, and ZARA2. For training and testing, we follow the standard leave-one-out

approach, where we train on 4 scenes and test on the remaining one. We follow the train-validation-test split of Social-GAN [25].

SDD consists of 20 scenes captured in bird's eye view using aerial drones. This dataset contains various moving agents such as pedestrians, bicycles, and cars, but we focus on pedestrians for comparison, as with Trajnet [56]. We also follow the train-test split of Trajnet [56].

Metrics Evaluation of accuracy on trajectory prediction is done with the widely-used metrics, Average Displacement Error (ADE) and Final Displacement Error (FDE). ADE is the average L2 distance between the predicted and ground truth trajectories over time. FDE is the L2 distance between the endpoints of the predicted and ground truth trajectories. We follow the Best-of-N procedure [25] with $N = 20$.

For evaluation of the accuracy of density estimation, we use earth mover's distance (EMD) and log-probability. EMD is a measure of the distance between two densities by the minimum cost to move one density pile to the other. EMD is calculated between the estimated density and ground truth density on each prediction step. Thus, we use EMD on *Simfork* dataset, where the ground truth densities are available. The advantage of EMD over the widely-used KL divergence is that EMD works properly even if the two density is disjointed while KL divergence returns a constant value. We use log-probability on the pedestrian trajectory prediction datasets, where the ground truth density is unavailable. The log-probability is calculated on the ground truth trajectory instead of density.

Implementation Details. Each conditional CIFs in our FlowChain used three-layer RealNVP [14] with multilayer perceptions (MLPs) of three hidden layers and 128 hidden units. All the conditional CIFs are with independent weights conditioned on the feature vectors from Trajectron++ [58] encoder as the temporal-social encoder. We used a batch size of 128 and ADAM optimizer [30] with a learning rate of 10^{-4}. All experiments were conducted on a single NVIDIA A100 GPU. We use a KDE implementation utilizing GPU as mentioned in Sec. 2.

Table 1. **Quantitative results of trajectory prediction on *ETH/UCY* dataset with Best-of-20 metrics.** T + I denotes its method takes a trajectory and an image as inputs while T denotes its method takes only a trajectory. Scores are in meters. **Red** and **blue** scores denote the best and the second-best in methods with T. Lower is better. The results of Trajectron++ [58] and MID [24] are updated according to an implementation issue #53 on the Trajectron++ GitHub page.

Method	Input	ETH		HOTEL		UNIV		ZARA1		ZARA2		Mean	
		ADE	FDE	ADE	FDE	ADE	FDE	ADE	FDE	ADE	FDE	ADE	FDE
SoPhie [57]	T + I	0.70	1.43	0.76	1.67	0.54	1.24	0.30	0.63	0.38	0.78	0.54	1.15
Goal-GAN [10]	T + I	0.59	1.18	0.19	0.35	0.60	1.19	0.43	0.87	0.32	0.65	0.43	0.85
MG-GAN [9]	T + I	0.47	0.91	0.14	0.24	0.54	1.07	0.36	0.73	0.29	0.60	0.36	0.71
Social-LSTM [1]	T	1.09	2.35	0.79	1.76	0.67	1.40	0.47	1.00	0.56	1.17	0.72	1.54
Social-GAN [25]	T	0.87	1.62	0.67	1.37	0.76	1.52	0.35	0.68	0.42	0.84	0.61	1.21
STGAT [29]	T	0.65	1.12	0.35	0.66	0.52	1.10	0.34	0.69	0.29	0.60	0.43	0.83
Social-STGCNN [42]	T	0.64	1.11	0.49	0.85	0.44	0.79	0.34	0.53	0.30	0.48	0.44	0.75
PECNet [41]	T	0.61	1.07	**0.22**	0.39	0.34	0.56	0.25	0.45	**0.19**	**0.33**	0.32	0.56
Trajectron++ [58]	T	**0.61**	1.03	**0.20**	**0.28**	**0.30**	**0.55**	**0.24**	**0.41**	**0.18**	**0.32**	**0.31**	**0.52**
MID [24]	T	**0.55**	**0.88**	**0.20**	**0.35**	**0.30**	**0.55**	0.29	0.51	0.20	0.38	**0.31**	**0.53**
Social Implicit [43]	T	0.66	1.44	**0.20**	0.36	0.31	0.60	0.25	0.50	0.22	0.43	0.33	0.67
FlowChain(Ours)	T	**0.55**	**0.99**	**0.20**	**0.35**	**0.29**	**0.54**	**0.22**	**0.40**	0.20	0.34	**0.29**	**0.52**

Table 2. **Quantitative results of trajectory prediction on *SDD* dataset with Best-of-20 metrics.** Scores are in pixels. See the caption of Tab. 1 also for details.

Method	Input	ADE	FDE
SoPhie [57]	T + I	16.27	29.38
Goal-GAN [10]	T + I	12.20	22.10
MG-GAN [9]	T + I	13.60	25.80
Social-LSTM [1]	T	31.19	56.97
Social-GAN [25]	T	27.25	41.44
STGAT [29]	T	14.85	28.17
Social-STGCNN [42]	T	20.76	33.18
PECNet [41]	T	**9.97**	**15.89**
Trajectron++ [58]	T	11.40	20.12
MID [24]	T	10.31	17.37
FlowChain (Ours)	T	**9.93**	**17.17**

4.2. Prediction Accuracy

A wide range of previous trajectory prediction methods is quantitatively compared with our FlowChain on *ETH/UCY* and *SDD* datasets, shown in Tab. 1 and Tab. 2. We didn't include some methods [40, 63, 70–72] due to a different train-validation-test split.

For *ETH/UCY* dataset, FlowChain achieved mean ADE/FDE of $0.29/0.52$ in meters, which are the best among all trajectory prediction methods. On *SDD* dataset, FlowChain achieved average ADE/FDE of $9.93/17.17$ in pixels, which are the best and second-best among all methods taking only trajectories as inputs [1, 24, 25, 41, 58, 63].

Although our focus is on density estimation with a minimal computational cost, FlowChain is comparable with the state-of-the-art methods also in terms of errors. We didn't evaluate the accuracy of future trajectories from FlowChain's update because the trajectories do not change
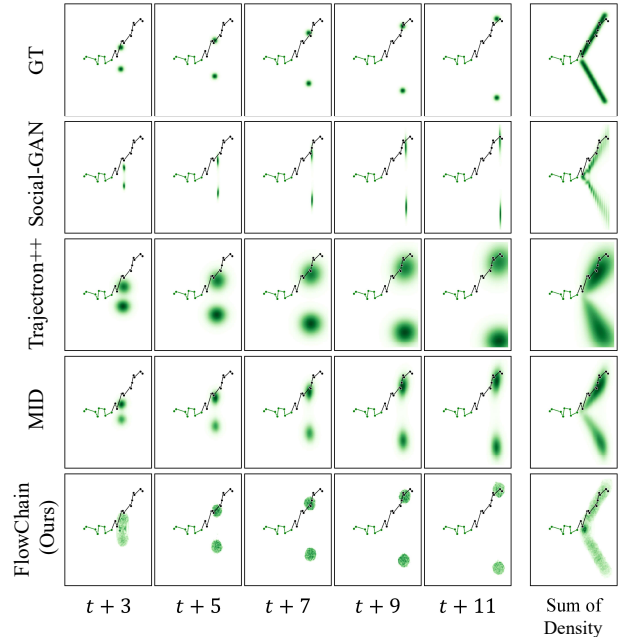

Figure 4. **Temporal estimated densities on *Simfork* dataset.**

in the update as mentioned in Sec. 3.4.

4.3. Analysis on Density Estimation

The results of density estimation are compared between FlowChain and previous methods based on VAE, GAN, and diffusion approaches. Trajectron++ is the representative VAE-based trajectory prediction model with density estimation by GMM. Social-GAN is a minimal model of GAN-based trajectory prediction. MID is the diffusion-based trajectory prediction model with state-of-the-art prediction accuracy. We applied KDE to Social-GAN and MID since
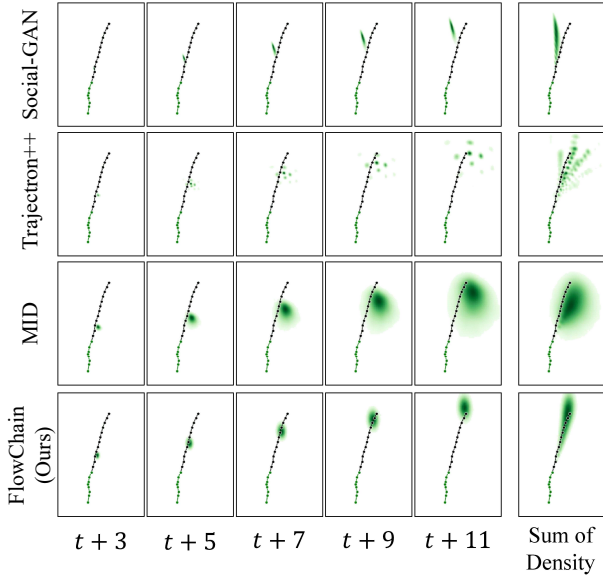
Figure 5. **Temporal estimated densities on *ETH/UCY* dataset.**

Figure 6. **Temporal densities estimated by our update procedure on *Simfork* dataset.**

Figure 7. **Temporal densities estimated by our update procedure on *ETH/UCY* dataset.**

they cannot estimate the density alone. For a fair comparison, FlowChain, Social-GAN, and MID generated the same number of samples for future trajectories; 10^5 samples.

We evaluated the computational time and the accuracy of density estimation of each method, as shown in Tab. 3. Social-GAN and MID take much longer time because KDE is slow in spite of running on GPU. Trajectron++ is a bit faster than our FlowChain due to its simple Gaussian mixture model but suffers from the poor quality of estimated density, as shown by the worst EMD, 2.545, on *Simfork* dataset and worst log-probability, -281.86, on *ETH/UCY* dataset. On the other hand, our FlowChain achieved the best EMD, 1.408, and the best log-probability, -0.26, in spite of the fast computation $37.0ms$. Thus, our method successfully addresses both the computational cost and accurate density estimation.

The estimated densities on *Simfork* are visualized in Fig. 4. The densities estimated by Trajectron++ are too large compared to the ground truth densities. While the densities of Social-GAN and MID are stretched to the vertical axis, our FlowChain successfully estimated the tighter round-shape densities closest to the ground truth densities.

The estimated densities on *ETH/UCY* are shown in Fig. 5. The size of each density represents the positional uncertainty of a predicted position. As within *Simfork*, in *ETH/UCY*, an agent cannot change its traveling direction significantly in a short period (i.e., $T_f = 12$ is $0.4 \times 12 = 4.8$ seconds), the size of the density expected to be tight. The densities estimated by Social-GAN are tight but deviated from the ground truth trajectory. Trajectron++ generated the densities with several disconnected modes due to the approximation as a Gaussian mixture. While the densi-
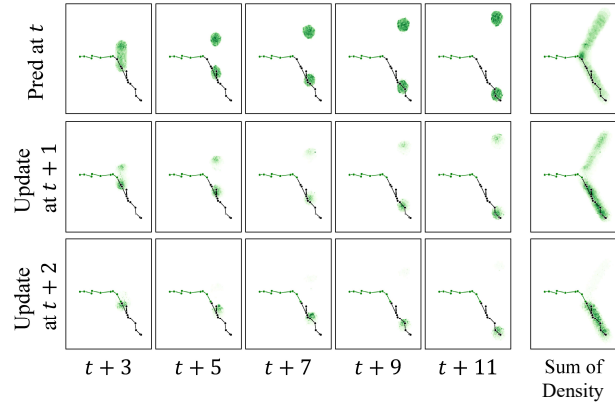
ties from MID are close to the ground truth trajectory, MID could not suggest tight densities and their modes deviated from the ground truth trajectories. On the contrary, the density from our FlowChain covers the ground truth trajectory with tight distributions. thanks to the explicit density estimation of conditional CIFs and our expressive model architecture. We should note that the estimated density of FlowChain has some fluctuation due to the looseness of the density computation of CIFs. However, we did not observe any potential problem or failure of density estimation related to this in our experiments.

4.4. Proposed Update of Density Estimation

As shown in Tab. 3, the accuracy of density estimation degraded along with the number of updates. However, this update is remarkably fast $0.6ms$, also shown in Tab. 3. Thus, autonomous systems can select the accurate prediction from scratch or the super rapid update by reusing predictions depending on the utilization of their computational resources.

The estimated densities of our update procedure are shown in Fig. 6 and Fig. 7, which show the densities up-

Table 3. **Accuracy and computational time of density estimation.** Scores are in milliseconds for computational time. While lower is better for EMD on *Simfork* dataset, higher is better for log-probablity on *ETH/UCY* dataset. We report the averaged EMD and log-probability over the 12 prediction steps.

Method		Comp. Time		Accuracy of Density Estimation	
		All	**w/o KDE**	**EMD** $\downarrow$	**log-probability** $\uparrow$
Social-GAN [25]		11498.8	9.8	1.544	-3.40
Trajectron++ [58]		33.7	-	2.545	-281.86
MID [24]		23904.1	12219.1	2.135	-0.90
	Pred at t	37.0	-	**1.408**	**-0.26**
FlowChain (Ours)	Update from $t-1$	0.6	-	2.539	-0.45
	Update from $t-5$	0.5	-	2.899	-0.53

Table 4. **Accuracy of density estimation with the update.**

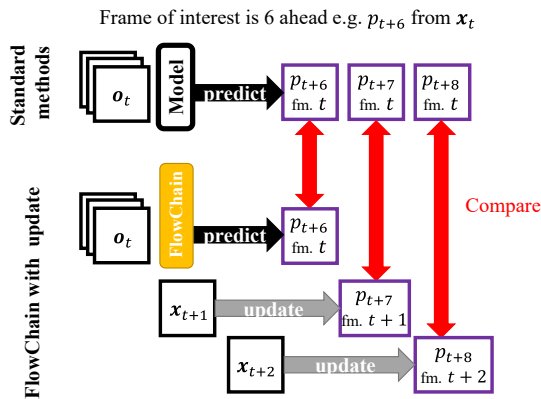| Method		log-probability on each step							
		t+6	**t+7**	**t+8**	**t+9**	**t+10**	**t+11**	**t+12**	**mean**
Social-GAN [25]		-2.65	-2.86	-3.07	-3.26	-3.44	-3.62	-3.80	-3.24
Trajectron++ [58]		-121.19	-93.61	-71.62	-51.36	-36.85	-27.92	-22.72	-97.79
MID [24]		-1.06	-1.43	-1.75	-2.02	-2.26	-2.46	-2.65	-1.95
FlowChain	Without Update	0.21	0.36	**0.44**	-0.08	-0.78	-0.75	-1.19	-0.26
	With Update	**0.20**	**0.50**	0.39	**0.44**	**0.20**	**0.26**	**0.16**	**0.31**

Figure 8. **Comparison of standard methods and FlowChain with updates.**

dated at $t+1$ and $t+2$ from the prediction at t. On *Simfork* dataset, the wrong mode is successfully suppressed by exploiting the information of the new observation. On *ETH/UCY* dataset, the initial prediction deviated from the ground truth because the observed trajectory observed until t does not have the information on turning left. However, the updated densities covered the ground truth well by adopting the *newest observed position* despite the minimal computational cost.

We further perform a comparison with a practical setting. Models usually make predictions with intervals due to the large computational cost. Therefore, we usually cannot obtain the prediction from the latest observation, as shown by the upper side of Fig. 8. For example, assume we need the density of six time-step ahead such as p_{t+6} from $\mathbf{x}_t$, we cannot obtain p_{t+7} from $\mathbf{x}_{t+1}$ or p_{t+8} from $\mathbf{x}_{t+2}$ due to the

computational overhead. However, with FlowChain's rapid update, we can obtain the update (p_{t+7}, p_{t+8}) from the latest observation ($\mathbf{x}_{t+1}$,$\mathbf{x}_{t+2}$) on every time step thanks to the minimal computational cost, as shown by the bottom side of Fig. 8. We conducted the comparison with the frame of interest as six time-step ahead $t+6$ and the log-probability metric. As shown by Tab. 4, our FlowChain with update achieves the highest log-probability. Therefore, we can conclude that FlowChain's update can improve the density estimation with minimal computational cost.

5. Concluding Remarks

In this paper, we proposed a new normalizing flow-based trajectory prediction network named FlowChain that achieves fast and accurate probability density estimation. Furthermore, our update procedure reliably generates the density estimation of the next step based on the density estimation of the previous step in an instant. Experimental results demonstrated that FlowChain achieved comparable accuracy to the state-of-the-art trajectory prediction models and superiority in the accuracy of density estimation.

Our FlowChain has one limitation. Our update procedure can reliably estimate the density for several steps but the performance will degrade along with the time steps because the reused flow transformations do not reflect the new observations. Our future work is determining when should we stop to use the update procedure and make predictions from scratch. Furthermore, FlowChain is the domain-agnostic model that can be applied to a wide range of prediction tasks such as stock price prediction, human motion prediction, and video prediction. We will investigate the applicability of our FlowChain on these other domains.

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 *CVPR*, pages 961–971, 2016.

[2] Jens Behrmann, Will Grathwohl, Ricky TQ Chen, David Duvenaud, and Jörn-Henrik Jacobsen. Invertible residual networks. In *ICML*, pages 573–582, 2019.

[3] Alberto Bernacchia and Simone Pigolotti. Self-consistent method for density estimation. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 73(3):407–422, 2011.

[4] Judith Bütepage, Hedvig Kjellström, and Danica Kragic. Anticipating many futures: Online human motion prediction and generation for human-robot interaction. In *ICRA*, pages 4563–4570, 2018.

[5] Jianfei Chen, Cheng Lu, Biqi Chenli, Jun Zhu, and Tian Tian. Vflow: More expressive generative flows with variational data augmentation. In *ICML*, pages 1660–1669, 2020.

[6] Ricky TQ Chen, Jens Behrmann, David K Duvenaud, and Jörn-Henrik Jacobsen. Residual flows for invertible generative modeling. *NeurIPS*, 32, 2019.

[7] Rob Cornish, Anthony Caterini, George Deligiannidis, and Arnaud Doucet. Relaxing bijectivity constraints with continuously indexed normalising flows. In *ICML*, pages 2133–2143, 2020.

[8] Henggang Cui, Vladan Radosavljevic, Fang-Chieh Chou, Tsung-Han Lin, Thi Nguyen, Tzu-Kuo Huang, Jeff Schneider, and Nemanja Djuric. Multimodal trajectory predictions for autonomous driving using deep convolutional networks. In *ICRA*, pages 2090–2096, 2019.

[9] Patrick Dendorfer, Sven Elflein, and Laura Leal-Taixé. Mg-gan: A multi-generator model preventing out-of-distribution samples in pedestrian trajectory prediction. In *ICCV*, pages 13158–13167, 2021.

[10] Patrick Dendorfer, Aljosa Osep, and Laura Leal-Taixé. Goalgan: Multimodal trajectory prediction based on goal position estimation. In *ACCV*, 2020.

[11] Nachiket Deo and Mohan M Trivedi. Convolutional social pooling for vehicle trajectory prediction. In *CVPRW*, pages 1468–1476, 2018.

[12] Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. *NeurIPS*, 34:8780–8794, 2021.

[13] Laurent Dinh, David Krueger, and Yoshua Bengio. Nice: Non-linear independent components estimation. *ICLR Workshop*, 2015.

[14] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real nvp. *ICLR*, 2017.

[15] Jianqing Fan and James S Marron. Fast implementations of nonparametric curve estimators. *Journal of computational and graphical statistics*, 3(1):35–56, 1994.

[16] Theo Gasser and Alois Kneip. Discussion: linear smoothers and additive models. *The Annals of Statistics*, 17(2):532–535, 1989.

[17] Mathieu Germain, Karol Gregor, Iain Murray, and Hugo Larochelle. Made: Masked autoencoder for distribution estimation. In *ICML*, pages 881–889, 2015.

[18] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *Communications of the ACM*, 63(11):139–144, 2020.

[19] Artur Gramacki and Jarosław Gramacki. FFT-based fast computation of multivariate kernel density estimators with unconstrained bandwidth matrices. *Journal of Computational and Graphical Statistics*, 26(2):459–462, 2017.

[20] Alexander Gray and Andrew Moore. N-body'problems in statistical learning. *Advances in neural information processing systems*, 13, 2000.

[21] Alexander G Gray and Andrew W Moore. Nonparametric density estimation: Toward computational tractability. In *Proceedings of the 2003 SIAM International Conference on Data Mining*, pages 203–211, 2003.

[22] Leslie Greengard and John Strain. The fast Gauss transform. *SIAM Journal on Scientific and Statistical Computing*, 12(1):79–94, 1991.

[23] Leslie Greengard and Xiaobai Sun. A new version of the fast Gauss transform. *Documenta Mathematica*, 3:575–584, 1998.

[24] Tianpei Gu, Guangyi Chen, Junlong Li, Chunze Lin, Yongming Rao, Jie Zhou, and Jiwen Lu. Stochastic trajectory prediction via motion indeterminacy diffusion. In *CVPR*, pages 17113–17122, 2022.

[25] Agrim Gupta, Justin Johnson, Li Fei-Fei, Silvio Savarese, and Alexandre Alahi. Social gan: Socially acceptable trajectories with generative adversarial networks. In *CVPR*, pages 2255–2264, 2018.

[26] Dirk Helbing and Peter Molnar. Social force model for pedestrian dynamics. *Physical review E*, 51(5):4282, 1995.

[27] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *NeurIPS*, 33:6840–6851, 2020.

[28] Xin Huang, Stephen G McGill, Brian C Williams, Luke Fletcher, and Guy Rosman. Uncertainty-aware driver trajectory prediction at urban intersections. In *ICRA*, pages 9718–9724, 2019.

[29] Yingfan Huang, Huikun Bi, Zhaoxin Li, Tianlu Mao, and Zhaoqi Wang. Stgat: Modeling spatial-temporal interactions for human trajectory prediction. In *ICCV*, pages 6272–6281, 2019.

[30] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR Poster*, 2015.

[31] Durk P Kingma and Prafulla Dhariwal. Glow: Generative flow with invertible 1x1 convolutions. *NeurIPS*, 31, 2018.

[32] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *ICLR*, 2014.

[33] Zhifeng Kong, Wei Ping, Jiaji Huang, Kexin Zhao, and Bryan Catanzaro. Diffwave: A versatile diffusion model for audio synthesis. In *ICLR*, 2021.

[34] Christophe G Lambert, Scott E Harrington, Campbell R Harvey, and Arman Glodjo. Efficient on-line nonparametric kernel density estimation. *Algorithmica*, 25:37–57, 1999.

[35] Nicolas Langrené and Xavier Warin. Fast and stable multivariate kernel density estimation by fast sum updating. *Journal of Computational and Graphical Statistics*, 28(3):596–608, 2019.

[36] Dongryeol Lee, Andrew Moore, and Alexander Gray. Dual-tree fast gauss transforms. *Advances in Neural Information Processing Systems*, 18, 2005.

[37] Alon Lerner, Yiorgos Chrysanthou, and Dani Lischinski. Crowds by example. In *Computer graphics forum*, volume 26, pages 655–664, 2007.

[38] Xin Li, Xiaowen Ying, and Mooi Choo Chuah. Grip: Graph-based interaction-aware trajectory prediction. In *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, pages 3960–3966, 2019.

[39] Hongyi Liu and Lihui Wang. Human motion prediction for human-robot collaboration. *Journal of Manufacturing Systems*, 44:287–294, 2017.

[40] Karttikeya Mangalam, Yang An, Harshayu Girase, and Jitendra Malik. From goals, waypoints & paths to long term human trajectory forecasting. In *ICCV*, pages 15233–15242, 2021.

[41] 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 *ECCV*, pages 759–776, 2020.

[42] Abduallah Mohamed, Kun Qian, Mohamed Elhoseiny, and Christian Claudel. Social-stgcnn: A social spatio-temporal graph convolutional neural network for human trajectory prediction. In *CVPR*, pages 14424–14432, 2020.

[43] Abduallah Mohamed, Deyao Zhu, Warren Vu, Mohamed Elhoseiny, and Christian Claudel. Social-implicit: Rethinking trajectory prediction evaluation and the effectiveness of implicit maximum likelihood estimation. In *ECCV*, pages 463–479, 2022.

[44] Didrik Nielsen, Priyank Jaini, Emiel Hoogeboom, Ole Winther, and Max Welling. Survae flows: Surjections to bridge the gap between vaes and flows. *NeurIPS*, 33:12685–12696, 2020.

[45] Travis A O'Brien, William D Collins, Sara A Rauscher, and Todd D Ringler. Reducing the computational cost of the ecf using a nuFFT: A fast and objective probability density estimation method. *Computational Statistics & Data Analysis*, 79:222–234, 2014.

[46] Travis A O'Brien, Karthik Kashinath, Nicholas R Cavanaugh, William D Collins, and John P O'Brien. A fast and objective multidimensional kernel density estimation method: fastkde. *Computational Statistics & Data Analysis*, 101:148–160, 2016.

[47] George Papamakarios, Theo Pavlakou, and Iain Murray. Masked autoregressive flow for density estimation. *NeurIPS*, 30, 2017.

[48] Emanuel Parzen. On estimation of a probability density function and mode. *The annals of mathematical statistics*, 33(3):1065–1076, 1962.

[49] Stefano Pellegrini, Andreas Ess, Konrad Schindler, and Luc Van Gool. You'll never walk alone: Modeling social behavior for multi-target tracking. In *ICCV*, pages 261–268, 2009.

[50] Stefano Pellegrini, Andreas Ess, and Luc Van Gool. Improving data association by joint modeling of pedestrian trajectories and groupings. In *ECCV*, pages 452–465, 2010.

[51] Danilo Rezende and Shakir Mohamed. Variational inference with normalizing flows. In *ICML*, pages 1530–1538. PMLR, 2015.

[52] Nicholas Rhinehart, Rowan McAllister, Kris Kitani, and Sergey Levine. PRECOG: prediction conditioned on goals in visual multi-agent settings. In *ICCV*, 2019.

[53] Alexandre Robicquet, Amir Sadeghian, Alexandre Alahi, and Silvio Savarese. Learning social etiquette: Human trajectory understanding in crowded scenes. In *ECCV*, pages 549–565, 2016.

[54] Murray Rosenblatt. Remarks on some nonparametric estimates of a density function. *The annals of mathematical statistics*, pages 832–837, 1956.

[55] Andrey Rudenko, Luigi Palmieri, Michael Herman, Kris M Kitani, Dariu M Gavrila, and Kai O Arras. Human motion trajectory prediction: A survey. *The International Journal of Robotics Research*, 39(8):895–935, 2020.

[56] Amir Sadeghian, Vineet Kosaraju, Agrim Gupta, Silvio Savarese, and Alexandre Alahi. Trajnet: Towards a benchmark for human trajectory prediction. *arXiv preprint*, 2018.

[57] 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 *CVPR*, pages 1349–1358, 2019.

[58] Tim Salzmann, Boris Ivanovic, Punarjay Chakravarty, and Marco Pavone. Trajectron++: Dynamically-feasible trajectory forecasting with heterogeneous data. In *ECCV*, pages 683–700, 2020.

[59] Christoph Schöller and Alois Knoll. Flomo: Tractable motion prediction with normalizing flows. In *IROS*, pages 7977–7984, 2021.

[60] David W Scott. Averaged shifted histograms: effective nonparametric density estimators in several dimensions. *The Annals of Statistics*, pages 1024–1040, 1985.

[61] Burkhardt Seifert, Michael Brockmann, Joachim Engel, and Theo Gasser. Fast algorithms for nonparametric curve estimation. *Journal of Computational and Graphical Statistics*, 3(2):192–213, 1994.

[62] Bernhard W Silverman. Algorithm as 176: Kernel density estimation using the fast fourier transform. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 31(1):93–99, 1982.

[63] Jianhua Sun, Yuxuan Li, Hao-Shu Fang, and Cewu Lu. Three steps to multimodal trajectory prediction: Modality clustering, classification and synthesis. In *ICCV*, pages 13250–13259, 2021.

[64] Esteban G Tabak and Cristina V Turner. A family of nonparametric density estimation algorithms. *Communications on Pure and Applied Mathematics*, 66(2):145–164, 2013.

[65] Esteban G Tabak and Eric Vanden-Eijnden. Density estimation by dual ascent of the log-likelihood. *Communications in Mathematical Sciences*, 8(1):217–233, 2010.

[66] Anirudh Vemula, Katharina Muelling, and Jean Oh. Social attention: Modeling attention in human crowds. In *ICRA*, pages 4601–4607, 2018.

[67] MP Wand. Fast computation of multivariate kernel estimators. *Journal of Computational and Graphical Statistics*, 3(4):433–445, 1994.

[68] Hao Wu, Jonas Köhler, and Frank Noé. Stochastic normalizing flows. *NeurIPS*, 33:5933–5944, 2020.

[69] 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 *CVPR*, pages 11385–11395, 2020.

[70] Cunjun Yu, Xiao Ma, Jiawei Ren, Haiyu Zhao, and Shuai Yi. Spatio-temporal graph transformer networks for pedestrian trajectory prediction. In *ECCV*, 2020.

[71] Ye Yuan, Xinshuo Weng, Yanglan Ou, and Kris M Kitani. Agentformer: Agent-aware transformers for socio-temporal multi-agent forecasting. In *ICCV*, pages 9813–9823, 2021.

[72] Jiangbei Yue, Dinesh Manocha, and He Wang. Human trajectory prediction via neural social physics. In *ECCV*, pages 376–394, 2022.

[73] He Zhao and Richard P Wildes. Where are you heading? dynamic trajectory prediction with expert goal examples. In *ICCV*, pages 7629–7638, 2021.