2024 IEEE Intelligent Vehicles Symposium (IV)
June 2-5, 2024. Jeju Island, Korea

TrajFlow: Learning Distributions over Trajectories for Human Behavior Prediction

Anna Mészáros*, Julian F. Schumann, Javier Alonso-Mora, Arkady Zgonnikov, and Jens Kober

2024 IEEE Intelligent Vehicle Symposium (IV) | 979-8-3503-4881-1/24/$31.00 ©2024 IEEE | DOI: 10.1109/IV55156.2024.10588386

***Abstract*— Predicting the future behavior of human road users is an important aspect for the development of risk-aware autonomous vehicles. While many models have been developed towards this end, effectively capturing and predicting the variability inherent to human behavior still remains an open challenge. This paper proposes TrajFlow—a new approach for probabilistic trajectory prediction based on Normalizing Flows. We reformulate the problem of capturing distributions over trajectories into capturing distributions over abstracted trajectory features using an autoencoder, simplifying the learning task of the Normalizing Flows. TrajFlow outperforms state-of-the-art behavior prediction models in capturing full trajectory distributions in two synthetic benchmarks with known true distributions, and is competitive on the naturalistic datasets ETH/UCY, rounD, and nuScenes. Our results demonstrate the effectiveness of TrajFlow in probabilistic prediction of human behavior.**

I. INTRODUCTION

Autonomous vehicles (AVs) have become an important field of research due to many promised benefits which include, but are not limited to, improved safety, accessibility, as well as reduced traffic congestion [1], [2], [3]. Yet they are still not widespread, in big part due to their inability to effectively resolve interactions with humans [4], [5]. Being able to reliably and accurately predict human behavior would allow for more efficient and safe AV path planning [6].

However, predicting human behavior in traffic is complicated by the fact that such behavior is generally not deterministic, but instead stochastic, with potentially complex and multi-modal distributions [7]. An example of such multi-modality can be seen at roundabouts, where vehicles have the option to enter the roundabout directly or to wait for an oncoming car to pass. While these two options are the most obvious high-level behaviors, there can also be other distinct modes, such as deciding whether or not to slow down before entering the roundabout (Fig. 1). Such modes are scenario-dependent and may get overlooked by methods that rely on a predefined number of modes [8], [9].

Several methodologies for providing probabilistic predictions over traffic agents' future trajectories have been proposed, ranging from Gaussian Mixture Models (GMMs) [8], [9] to generative networks. Generative networks such as Generative Adversarial Networks (GANs) [10], [11], Variational Autoencoder (VAE) based networks [12], [13], [14], and diffusion models [15], are particularly interesting due to their potential to learn complex multi-modal distributions without specifying the number of expected modes, unlike methods that rely on GMMs. While these state-of-the-art approaches already achieve good results in prediction accuracy, they have the fundamental problem of being trained to reproduce the *only* true future trajectory available for each past trajectory in the dataset, thereby ignoring the underlying stochasticity of human behavior. This training approach can result in mode collapse, which is especially problematic for GAN-based methods [10], [11]. Additionally, many state-of-the-art models predict distributions at individual time steps [12], [16], ignoring the correlation between different time steps. These kinds of predictions can lead to more conservative strategies within the subsequent motion planning [17].

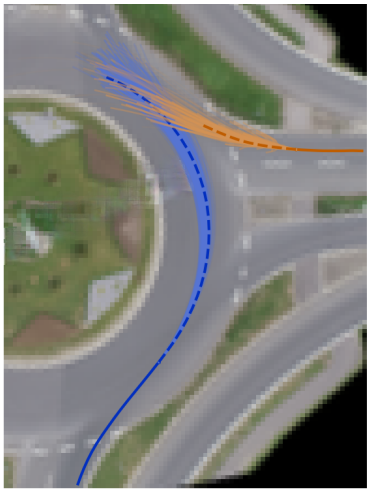

Fig. 1. An example of the predictions generated by TrajFlow on the rounD dataset. Level of opacity indicates the likelihood of a given prediction.

To overcome these issues, one promising approach is Normalizing Flows (NFs) [18], [19], which are specifically designed to learn underlying distributions in data and have been shown to have the capability of capturing multi-modal distributions. While NFs can be used to learn distributions of positions at individual time steps [20], [21], more recent work has expanded to providing distributions over complete trajectories [22], [23], [24]. Even though the above NF methods already demonstrate good qualitative results in predicting multiple future trajectories, it remains unclear how well these models capture the true underlying distribution of the data. Furthermore, the previously mentioned NF models which predict over the complete trajectories require one to set the

*Corresponding author

This research was supported by NWO-NWA project "Acting under uncertainty" (ACT), NWA.1292.19.298.

All authors are with the Cognitive Robotics Department, TU Delft, 2628 CB Delft, The Netherlands {A.Meszaros, J.F.Schumann, J.AlonsoMora, A.Zgonnikov, J.Kober}@tudelft.nl

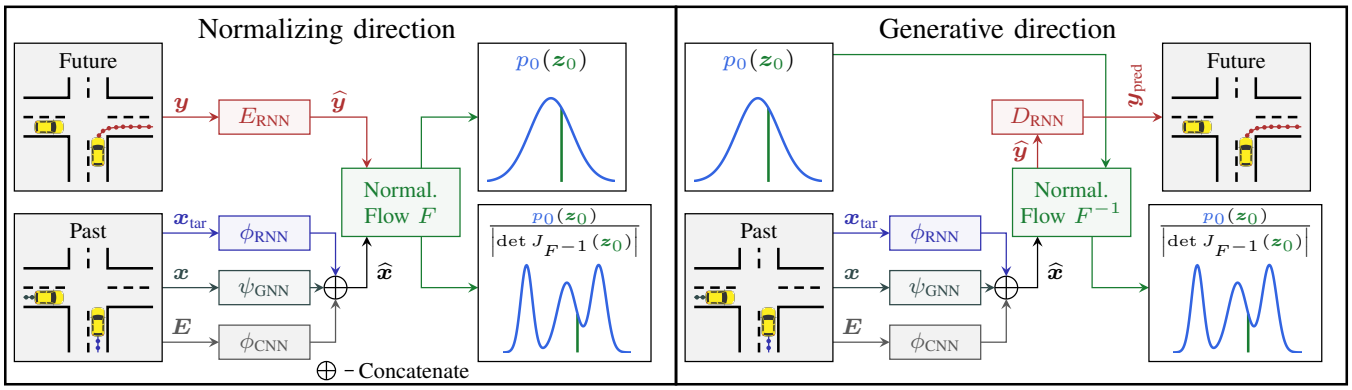

Fig. 2. Architecture of TrajFlow. During training we use the normalizing direction in which we encode the future trajectories $\boldsymbol{y}$ with E_{RNN} and transform the abstracted features $\widehat{\boldsymbol{y}}$ to a sample $\boldsymbol{z}_0 = F(\widehat{\boldsymbol{y}})$ assumed to follow a standard normal distribution with the probability density function p_0. For inference we then use the generative direction, in which a sample $\boldsymbol{z}_0 \sim p_0$ is inversely transformed by the Normalizing Flow to generate the abstracted future trajectories $\widehat{\boldsymbol{y}} = F^{-1}(\boldsymbol{z}_0)$ that are decoded with D_{RNN} into the actual trajectories $\boldsymbol{y}_{\text{pred}}$. The likelihood of the encoded trajectory is obtained with $p_0(\boldsymbol{z}_0)|\det J_{F^{-1}}(\boldsymbol{z}_0)|^{-1}$. The encoding ϕ_{CNN} of map E, and the encoding ψ_{GNN} of social interactions are optional blocks, which can provide richer context information.

number of predicted time steps during their design, which might limit their applicability and usefulness in an online setting.

In response to these challenges, the main contribution of this work is *TrajFlow*—a prediction model with an improved capability for fitting distributions present in underlying training samples. The model builds on top of *FloMo* [24], which we extend with a key component in the form of a Recurrent Neural Network Autoencoder (Fig. 2). This extension generates an intermediate representation of the trajectories, which captures the most relevant features of the trajectories and in turn also simplifies the learning of the underlying distribution. The decoder of the Autoencoder is built in an auto-regressive manner, which additionally gives the model the flexibility to predict trajectories beyond the length of the seen training data. We validated our approach on a synthetic dataset for which we know the underlying distribution, as well as on an augmented version of the multi-modal Forking Paths dataset [25], and several popular real-world datasets (ETH/UCY [26], [27], rounD [28], and nuScenes [29]).

II. Background: Normalizing Flows

Normalizing Flows constitute a family of generative methods which enable exact likelihood computation. They are based on the concept of transforming distributions through a series of differentiable bijective functions into a simple known "base" distribution Z_0 – most commonly a standard normal distribution.

A number of ways for constructing flow models have been proposed [30]. One possible way is by using auto-regressive flows, consisting of a series of K normalizing layers. The main components of these layers are the conditioner c_k and the transformer τ_k, which are often accompanied by an additional permutation layer ϵ_k. The latter two functions (τ_k and ϵ_k) are bijective – and therefore invertible. In the generative direction, these functions then enable the transformation of a sample $\boldsymbol{z}_0$ from the base distribution Z_0 towards the desired distribution Z_K:

$$\boldsymbol{z}_{k+1} = \epsilon_k\left(\tau_k(\boldsymbol{z}_k;\boldsymbol{\theta}_k)\right), \quad \text{with} \quad \boldsymbol{\theta}_k = c_k\left(\boldsymbol{z}_k;\widehat{\boldsymbol{x}}\right),$$

where $\boldsymbol{z}_{k+1}$ is the result of the k-th intermediate transformation. Meanwhile, $\widehat{\boldsymbol{x}}$ is a conditioning input [31] that can take the form of e.g. an encoding of observations like past trajectories, static environment, and social interactions. In the normalizing direction, F is then a composition of all K layers, where it is possible to exploit the property of c_k that $\boldsymbol{\theta}_k = c_k\left(\epsilon_k^{-1}(\boldsymbol{z_{k+1}});\widehat{\boldsymbol{x}}\right)$:

$$F(\boldsymbol{z}_K) = \left(\tau_0^{-1}\circ\epsilon_0^{-1}\cdots\circ\tau_K^{-1}\circ\epsilon_K^{-1}\right)(\boldsymbol{z}_K) = \boldsymbol{z}_0$$

In the generative direction, this then allows the drawing of a sample $\boldsymbol{z}_K = F^{-1}(\boldsymbol{z}_0)$ from the desired non-normal distribution over outputs Z_K, using a sample $\boldsymbol{z}_0$ from Z_0. The Probability Density Function (PDF) p_K of Z_K can then also be obtained in terms of the PDF p_0:

$$\begin{aligned} p_K(\boldsymbol{z}_K) &= p_0(F(\boldsymbol{z}_K))\,|\det J_F(\boldsymbol{z}_K)| \\ &= p_0(\boldsymbol{z}_0)\,|\det J_{F^{-1}}(\boldsymbol{z}_0)|^{-1}, \end{aligned}$$

The absolute determinant of the Jacobian $|\det J_F(\boldsymbol{z}_K)|$ quantifies the relative change of volume within a small neighborhood of $\boldsymbol{z}_K$ when transforming it to a sample $\boldsymbol{z}_0$ using F. This ensures that the probability mass remains the same between the two distributions.

The parameters of F are learned by minimizing the KL-divergence between the target distribution Z_K^* with PDF $p_K^*(\boldsymbol{z}_K)$ and the learned distribution Z_K with the PDF $p_K(\boldsymbol{z}_K)$:

$$\begin{aligned} \mathcal{L} =& D_{\text{KL}}[p_K^*(\boldsymbol{z}_K)||p_K(\boldsymbol{z}_K)] \\ =& -\mathbb{E}_{\boldsymbol{z}_K\sim Z_K^*}\left[\log p_0(F(\boldsymbol{z}_K)) + \log|\det J_F(\boldsymbol{z}_K)|\right. \\ & \left. - \log p_K^*(\boldsymbol{z}_K)\right] \end{aligned}$$

With only a finite number N of samples $\boldsymbol{z}_{K,n}$ representing the underlying distribution Z_K^* and ignoring the constant part $\log p_K^*(\boldsymbol{z}_K)$, this loss can be approximated with:

$$\mathcal{L} \approx -\frac{1}{N}\sum_{n=1}^{N}\log p_0(F(\boldsymbol{z}_{K,n})) + \log|\det J_F(\boldsymbol{z}_{K,n})|.$$

III. METHOD: TRAJFLOW

We build up on the *FloMo* approach [24], in which NFs are employed for learning distributions directly on two-dimensional trajectories $\boldsymbol{y} \in \mathbb{R}^{n_O \times 2}$ defined at n_O future time steps (where $\boldsymbol{y} = \boldsymbol{z}_K$). However, attempting to learn a distribution over the trajectories directly makes it susceptible to overfitting to the variability inherent in human behavior [32] as well as noise in its measurements. Additionally, as n_O in this design is fixed, the model has a limited prediction horizon, hindering its general applicability. Furthermore, intuitively people do not observe trajectories as a series of precise positions at each time step. Instead, they perceive a trajectory more abstractly in terms of general direction, length, and shape. Therefore, learning the distribution over such abstracted characteristics might be better suited to mimic human decision making, an approach that has shown itself to be promising in improving prediction models [33], [34].

To overcome these challenges and to facilitate the learning of underlying distributions, we constructed our proposed model *TrajFlow* to let the NFs reason over trajectory abstractions rather than the raw trajectories.

A. Normalizing Flow

In our specific case, we chose to use a *Coupling Layer* for c_k, a *Rational Quadratic Spline* for τ_k, and a permutation layer ϵ_k, similar to the *FloMo* model [24]. However, unlike [24], we did not augment the trajectories in the training data. Furthermore, we also did not inject added noise into the Normalizing Flow as was done in FloMo using the β and γ hyperparameters. Lastly, we incorporated a learning rate decay lr for the training of the Normalizing Flow, as this has proven beneficial for achieving a better distribution fit.

B. Encoding Trajectories

To capture the abstracted characteristics of a trajectory, we utilized a Recurrent Neural Network Autoencoder (RNN-AE) with encoder E_{RNN} and decoder D_{RNN}. This allows us to create an abstraction of a trajectory $\widehat{\boldsymbol{y}} = E_{\text{RNN}}(\boldsymbol{y}) \in \mathbb{R}^m$. This addition results in the novel *TrajFlow* model (Fig. 2) that consequently learns the distribution of the encoded future trajectories $\widehat{Y}$ (with $\widehat{\boldsymbol{y}} = \boldsymbol{z}_K$) rather than the raw future trajectories Y.

1) Gated Recurrent Unit: The RNN-AE uses as its main component a so-called Gated Recurrent Unit (GRU) [35], one of the main RNNs used for encoding time series events. In its most basic single-layered form with embedding dimensionality M and hidden dimensionality d, it can be depicted as a function $\phi_{\text{GRU}} : \mathbb{R}^M \times \mathbb{R}^d \to \mathbb{R}^d$, which takes at time step t an input $\boldsymbol{a}_t \in \mathbb{R}^M$ and uses it to change its internal hidden state $\boldsymbol{h} \in \mathbb{R}^d$:

$$\boldsymbol{h}_t = \phi_{\text{GRU}}\left(\boldsymbol{a}_t, \boldsymbol{h}_{t-1}\right)$$

If no hidden layer is provided at the beginning of a sequence, those can be assumed to be zero. However, *TrajFlow* employs a multi-layered version using multiple recurrent units $\phi^{(l)}_{\text{GRU}}$, with $l \in \{1, \ldots, L\}$:

$$\boldsymbol{h}^{(l)}_t = \phi^{(l)}_{\text{GRU}}\left(\boldsymbol{h}^{(l-1)}_t, \boldsymbol{h}^{(l)}_{t-1}\right) \quad \text{with} \quad \boldsymbol{h}^{(0)}_t = \boldsymbol{a}_t$$

This can then be combined in a multilayer function $\phi_{\text{L-GRU}} : \mathbb{R}^M \times \mathbb{R}^{L \times d} \to \mathbb{R}^{L \times d}$ with $\boldsymbol{H}_t = \{\boldsymbol{h}^{(1)}_t, \ldots, \boldsymbol{h}^{(L)}_t\}$:

$$\boldsymbol{H}_t = \phi_{\text{L-GRU}}\left(\boldsymbol{a}_t, \boldsymbol{H}_{t-1}\right) \tag{1}$$

2) The RNN-Encoder: In the first step of the encoder E_{RNN}, we created a transformed trajectory $\widetilde{\boldsymbol{y}} = \langle \widetilde{\boldsymbol{y}}_1, \ldots, \widetilde{\boldsymbol{y}}_{n_O} \rangle$ with

$$\widetilde{\boldsymbol{y}}_t = \boldsymbol{y}_t - \boldsymbol{y}_{t-1} \tag{2}$$

This is based on previous results showing that displacement information is more useful for trajectory prediction tasks [36]. We then used a linear layer $\phi_{\text{em}} : \mathbb{R}^2 \to \mathbb{R}^M$ that embeds a displacement $\widetilde{\boldsymbol{y}}_t$. The embedded time steps are then run in sequence through a multi-layered GRU $\phi_{\text{E-L-GRU}}$ (1), setting $\boldsymbol{a}_t = \phi_{\text{em}}(\widetilde{\boldsymbol{y}}_t)$. Using a second linear layer $\phi_{\text{E}} : \mathbb{R}^d \to \mathbb{R}^m$, we get our final encoded trajectory $\widehat{\boldsymbol{y}} = \phi_{\text{E}}\left(\boldsymbol{h}^{(L)}_{E,n_O}\right)$.

3) The RNN-Decoder: Our decoder D_{RNN}, uses as its first step a liner layer $\phi_{\text{D}} : \mathbb{R}^m \to \mathbb{R}^d$ to pre-process an encoded trajectory $\widehat{\boldsymbol{y}}$. We then again used a multilayer GRU $\phi_{\text{D-L-GRU}}$ (Equation (1)) to construct a new trajectory. Here, the initial hidden states are set to $\boldsymbol{h}^{(l)}_{D,0} = \widehat{\boldsymbol{y}}$. Meanwhile, our input is auto-regressive, i.e. $\boldsymbol{a}_1 = \phi_{\text{D}}(\widehat{\boldsymbol{y}})$ and $\boldsymbol{a}_t = \phi_{\text{D}}\left(\boldsymbol{h}^{(L)}_{D,t-1}\right)$ for $t > 1$. We constructed the final displacements using a linear layer $\phi_{\text{out}} : \mathbb{R}^d \to \mathbb{R}^2$:

$$\widetilde{\boldsymbol{y}}_{t,\text{pred}} = \phi_{\text{out}}\left(\boldsymbol{h}^{(L)}_{D,t}\right)$$

As a last step, we used the cumulative sum over $\widetilde{\boldsymbol{y}}_{\text{pred}}$ to construct the predicted trajectory $\boldsymbol{y}_{\text{pred}}$ (inverting (2)). While we used the same hidden dimension d and embedding size M for both $\phi_{\text{E-L-GRU}}$ and $\phi_{\text{D-L-GRU}}$, we did not use any weight sharing between them.

4) Training: The RNN-AE is trained separately before the rest of the network with a root mean square error reconstruction loss on the reconstructed trajectories:

$$\mathcal{L}_{\text{AE}} = \frac{1}{N} \sum_{n=1}^{N} \|\boldsymbol{y}_n - \boldsymbol{y}_{\text{pred},n}\|.$$

The choice to calculate the loss on the reconstructed trajectories instead of the decoded displacements was made to penalize cumulative errors that can arise from summing over the displacements. During the later training of the remaining parts of the model, the weights of the RNN-AE were frozen.

C. Encoding Context Information

For the observations $\widehat{\boldsymbol{x}}$, which are used for conditioning the distributions learned by the NF, we used the target agent's past trajectory $\boldsymbol{x}_{\text{tar}}$, the past trajectories of all agents $\boldsymbol{x}$, and optionally images of the static environment E. In order to encode these pieces of information, we used the

neural networks ϕ_{RNN}, ψ_{GNN}, and ϕ_{CNN} respectively and concatenated their outputs:

$$\widehat{\boldsymbol{x}} = \phi_{\text{RNN}}(\boldsymbol{x}_{\text{tar}}) \oplus \psi_{\text{GNN}}(\boldsymbol{x}) \oplus \phi_{\text{CNN}}(\boldsymbol{E})$$

The exact implementation of these components can be found in the Appendix.

IV. Experimental Setup

We performed a number of tests, two on synthetic datasets with known ground truth distributions (Sec. V) and three on real-world datasets (Sec. VI). To facilitate those tests, we utilized an existing benchmarking framework [6].

A. Models

We used four state-of-the-art behavior prediction models as baselines:

- *Trajectron++ (T++)* [12]—selected as it continues to act as a strong baseline in trajectory prediction tasks. At the same time, it provides a good illustration of the potential drawbacks of fitting distributions per time step.
- *PECNet* [14]—a state-of-the-art model which captures multi-modality by predicting distributions over goal points and then regressing the trajectories to them.
- *Motion Indeterminacy Diffusion (MID)* [15]—a recent diffusion-based method for probabilistic trajectory prediction. As of late, diffusion based models have been showing promise in generating probabilistic predictions [37].
- *FloMo* [24] (FM)—since we build on this model, it is most directly comparable to our approach.
- *TrajFlow without the RNN-AE (TF w/o AE)*—to showcase the importance of the RNN-AE we tested against an ablation of TrajFlow.

For the RNN-AE in *TrajFlow* we used a $L = 3$ layered GRU with a hidden dimensionality $d = 20$, embedding dimensionality $M = 20$, and latent space dimensionality $m = 20$ (Sec. III-B.2).

For the past trajectory encoding ϕ_{RNN}, we set the parameters in accordance to those described in the Appendix. Meanwhile, where applicable we employed the same ψ_{GNN} and ϕ_{CNN} structures for *TrajFlow*, *TF w/o AE* and *FloMo*.

The learning rate decay was set to $lr = 0.98$. These same parameters were used for *TF w/o AE*.

B. Metrics

To evaluate the distance of the predicted trajectories w.r.t. the ground truth trajectory we use:

- **minADE/minFDE**: Average/Final L_2 distance (measured in meters) between the best-predicted trajectory and the ground truth, based on 20 predicted samples. We chose this metric primarily to obtain an interpretable measure of how closely the predictions of the models capture the single ground truth sample available in real-world test cases, since the usefulness of distribution specific metrics such as Negative Log-Likelihood (NLL) is limited when evaluating on singular ground truth samples.

In order to obtain insight into the learned distribution over trajectories, we make use of:

- $\mathbf{D}_{\text{JS}}$: Average Jensen-Shannon divergence [38] between the ground truth distribution and the learned distribution. A perfect fit of the distribution is characterized by a divergence value of 0 whereas two dissimilar distributions would result in a divergence value of 1. As this metric requires a known ground truth distribution, it is only applicable for the synthetic cases.
- **(indep.) NLL**: The average NLL of the ground truth according to the learned distribution of each individual agent. This gives us insight into the fit over the marginal distributions of the trajectories within the scene. This metric is commonly used in cases with any number of ground truth trajectories for a given scenario [12].
- **joint NLL**: The average NLL of the ground truth, based on the joint distribution of predicted trajectories for all agents in the scene. This metric gives additional information about how well the model learned the interactions between the agents in the scene.

To obtain the density estimates needed for the above metrics, we used a non-parametric density estimation approach proposed in [39] so as to ensure a more reliable comparison between models. For estimating the predicted trajectory distribution, 100 sampled trajectories were used.

V. Experiments: Synthetic Datasets

A. Datasets

We tested our approach on two synthetically generated datasets, one of which was generated using a single bimodal distribution, while the other is an augmented version of the Forking Paths dataset [25].

The **synthetic bimodal dataset** was used to test the models' ability to capture the underlying distribution of the observed data. We constructed this dataset based on two recorded pedestrian trajectories with distinct directions to obtain a set of future trajectories over which we know the underlying distribution. This synthetic dataset has only a single agent and no static environment information for the observations. The trajectories were split into a past sequence $\boldsymbol{x}_k^*$ with $n_I = 10$ and future sequence $\boldsymbol{y}_k^*$ with $n_O = 14$ recorded time steps of $0.25\,\text{s}$ each. However, for both future trajectories, we used the same past trajectory $\boldsymbol{x}_1^*$ so that the true output distribution is guaranteed to be bimodal. With this, we avoid the learned likelihoods becoming skewed due to slight differences in the past trajectories which could in turn make it more difficult to evaluate the predicted distributions. The set of future trajectories $\boldsymbol{Y}$ with 3000 samples was then created by multiplying the original two future trajectories with a random scaling factor $s \sim \mathcal{N}(1, 0.15)$:

$$\boldsymbol{Y} = \{s_{1,i}\boldsymbol{y}_1^*, s_{2,i}\boldsymbol{y}_2^* \mid i \in \{1, \dots, 1500\}\}$$

The **augmented Forking Paths dataset** was used to test the methods on a more complex, multi-modal dataset. The Forking Paths dataset [25] originally included multiple human-predicted pedestrian trajectories. Within this dataset, for each past trajectory $\boldsymbol{x}^*$ with $n_I = 8$, there are K annotated future trajectories $\boldsymbol{y}_k^*$ with $n_O = 12$, recorded

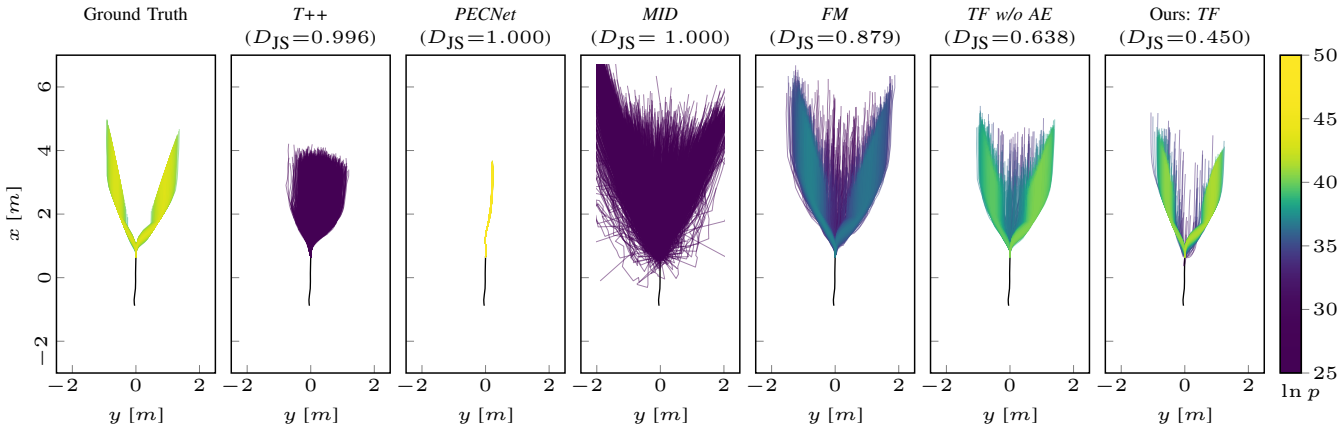

Fig. 3. Results of the experiments on the synthetic bimodal dataset. The left-most plot depicts the ground truth distribution; the other panels are the best out of the ten distributions learned by *Trajectron++ (T++)*, *PECNet*, *Motion Indeterminacy Diffusion (MID)*, *FloMo (FM)*, an ablation of *TrajFlow* without the RNN-AE *(TF w/o AE)*, and *TrajFlow (TF)*, along with the D_{JS} values for the specific distributions that are depicted. The colors provided in the distributions are determined based on the density values obtained through density estimation on 3000 samples obtained from the respective models.

TABLE I

SYNTHETIC BIMODAL DATASET: AVERAGE RESULTS ACROSS 10 SEEDS

Models	minADE	minFDE	NLL	D_{JS}
T++	0.43 $\pm$0.03	0.66 $\pm$0.09	-19.08 $\pm$3.92	0.998 $\pm$0.001
PECNet	0.90 $\pm$0.01	1.29 $\pm 4e^{-3}$	$0.8e^{3}$ $\pm$10.77	1.000 $\pm 1e^{-16}$
MID	0.72 $\pm$0.03	0.81 $\pm$0.20	-3.79 $\pm$1.88	1.000 $\pm 5e^{-7}$
FM	0.19 $\pm$0.03	0.32 $\pm$0.05	-35.83 $\pm$0.72	0.916 $\pm$0.016
TF w/o AE	0.13 $\pm$0.02	0.22 $\pm$0.03	-38.34 $\pm$1.05	0.811 $\pm$0.081
TF (Ours)	0.12 $\pm$0.01	0.20 $\pm$0.02	-39.22 $\pm$0.68	0.683 $\pm$0.132

TABLE II

FORKING PATHS: AVERAGE RESULTS ACROSS ALL SPLITS & SEEDS.

Models	minADE	minFDE	NLL	D_{JS}
T++	0.56 $\pm$0.06	1.02 $\pm$0.13	-5.73 $\pm$4.21	0.985 $\pm$0.005
PECNet	1.05 $\pm$0.09	1.89 $\pm$0.28	$1.3e^{3}$ $\pm 0.5e^{3}$	1.000 $\pm 1.3e^{-7}$
MID	0.89 $\pm$0.09	2.06 $\pm$0.38	-7.41 $\pm$2.17	1.000 $\pm 5.7e^{-5}$
FM	0.41 $\pm$0.04	0.70 $\pm$0.10	-21.86 $\pm$0.44	0.986 $\pm$0.003
TF w/o AE	0.40 $\pm$0.05	0.69 $\pm$0.11	-22.65 $\pm$0.69	0.982 $\pm$0.004
TF (Ours)	0.42 $\pm$0.06	0.71 $\pm$0.11	-23.69 $\pm$0.99	0.984 $\pm$0.004

with a sampling frequency of 2.5 Hz. We then generated 100 augmented trajectories for each k:

$$\boldsymbol{y}_{k,i} = s_{k,i}\boldsymbol{y}_k^* \boldsymbol{R}_{\theta_{k,i}}^T, \text{ with } k \in \{1,\dots,K\}, i \in \{1,\dots,100\}.$$

Here, $\boldsymbol{R}_\theta \in \mathbb{R}^{2\times 2}$ is a rotation matrix rotating $\boldsymbol{y}_k^*$ by $\theta \sim \mathcal{N}(0, \frac{\pi}{180})$, while $s \sim \mathcal{N}(1, 0.03)$ is a scaling factor.

B. Training and evaluation

Considering that the **synthetic bimodal dataset** contains a single scenario, we trained 10 instances of each model, using different random seeds, to decrease the effect of the random initialization of the models' trainable parameters.

Meanwhile, for the **augmented Forking Paths** dataset training and evaluation were performed using five-fold cross-validation, which was each repeated for 5 random seeds.

C. Results

On the synthetic bimodal dataset, we found that out of the tested models, the methods which did not employ Normalizing Flows exhibited the poorest performance. This can be attributed to different factors, from the lack of correlation between time steps (*T++*) to complete mode collapse (*PECNet*). The NF-based methods, meanwhile, were all able to capture the general shape of the underlying distribution. Out of these, our approach *TrajFlow (TF)* was able to provide the best fit. A key factor to this is the use of the RNN-AE, which becomes clear when comparing the distributions learned by *TF* with its ablation (Fig. 3). These qualitative results are further supported by the low NLL and D_{JS} values (Tab. I) attained by *TrajFlow*.

On the augmented Forking Paths dataset, we observed that none of the models could obtain distributions identical to the ones in the evaluation set – as indicated by D_{JS} values which are close to the maximum divergence value of 1 (Tab. II). This is likely due to the fact that even though each of the past inputs has a ground truth distribution over the future trajectories, similar inputs may have different output distributions which could be merged by the models and thus result in dissimilar distributions from the actual ground truth distribution. Nevertheless, NLL values show clear differences in the models' capability to capture the ground truth distributions; the Normalizing Flow methods are able to achieve better performance, with *TrajFlow* achieving the best performance.

VI. EXPERIMENTS: REAL-WORLD DATASETS

A. Datasets

We tested our approach on three real-world datasets, ETH/UCY [26], [27], rounD [28], and nuScenes [29], all of which are widely used for trajectory prediction.

For testing the models on **ETH/UCY** (mostly including pedestrian crowds), we used $n_I = 8$ and $n_O = 12$ with a sampling frequency of 2.5 Hz, resulting in 3.2 s and 4.8 s of past and future data respectively. Like in the majority of prior works, we did not make use of static environment information for the sake of comparability.

TABLE III

ETH/UCY: AVERAGE RESULTS ACROSS THE FIVE LOCATIONS.

Models	minADE	minFDE	indep. NLL	joint NLL
T++	0.41 ± 0.17	0.66 ± 0.26	-6.77 ± 7.93	$0.2e^3$ $\pm 0.5e^3$
PecNet	2.39 ± 3.00	3.34 ± 3.99	$1.2e^4$ $\pm 2.2e^4$	$2.7e^4$ $\pm 4.3e^4$
MID	0.59 ± 0.16	1.08 ± 0.30	-0.27 ± 10.21	$0.1e^3$ $\pm 0.2e^3$
FM	0.32 ± 0.13	0.55 ± 0.22	-19.65 ± 4.82	-50.05 ± 20.84
TF w/o AE	0.33 ± 0.12	0.54 ± 0.21	-18.02 ± 4.11	-42.24 ± 16.50
TF (Ours)	0.33 ± 0.15	0.55 ± 0.24	-21.05 ± 5.35	-75.79 ± 47.42

TABLE IV

ROUND: AVERAGE RESULTS ACROSS THE FIVE CROSS SPLITS.

Models	minADE	minFDE	indep. NLL	joint NLL
T++	0.69 ± 0.02	1.66 ± 0.07	-42.04 ± 1.10	-79.67 ± 1.98
PECNet	1.56 ± 0.11	4.49 ± 0.20	$3.4e^3$ $\pm 0.3e^3$	$6.1e^3$ $\pm 0.7e^3$
MID	4.57 ± 0.01	7.93 ± 0.07	$0.3e^3$ $\pm 0.1e^3$	$0.5e^3$ $\pm 0.2e^3$
FM	0.85 ± 0.03	1.80 ± 0.06	-34.98 ± 3.65	-69.35 ± 4.98
TF w/o AE	0.80 ± 0.01	1.72 ± 0.03	-36.86 ± 1.32	-71.62 ± 2.43
TF (Ours)	0.67 ± 0.03	1.38 ± 0.09	-42.06 ± 2.67	-81.23 ± 4.69

For testing the models on **rounD** (drone-captured roundabouts), we set $n_I = 15$ and $n_O = 25$ with a sampling frequency of 5 Hz, which amounts to 3 s and 5 s of past and future data respectively. For our evaluation, we used the scenarios extracted from the original dataset as done in [40], which focused on the gap acceptance scenario of a vehicle entering the roundabout. There, both the trajectories of the vehicle entering the roundabout and the trajectory of the vehicle already inside the roundabout, which might be cut off by the former vehicle, have to be predicted. As it is important to predict if the other vehicle might yield when trying to plan for such scenarios, it can be necessary to predict more than $n_O = 25$ time steps. This is the case in 5.9% of the scenes in rounD, with the longest predictions requiring 35 time steps.

Lastly, on **nuScenes** (general street traffic), we used $n_I = 4$ and $n_O = 12$ with a sampling frequency of 2 Hz. For both nuScenes and rounD, full context information is available.

B. Training and Evaluation

For **ETH/UCY**, training and evaluation were performed using a leave-one-out strategy [12], using the five recording locations (ETH-univ, ETH-hotel, UCY-univ, UCY-zara01, UCY-zara02).

For **rounD**, training and evaluation were performed using five-fold cross-validation. While we evaluated the normal minADE metric based on the first 25 time steps, we also checked the models' capability to predict beyond that to test its ability of extending predictions until the point by which a yielding decision had to be reached. If a model was unable to predict beyond the 25 future time steps that it had been trained on, the values for the remaining time steps were obtained through a simple constant velocity extrapolation.

Lastly, training and evaluation for **nuScenes** were performed using nuScenes' predefined training and validation splits. To decrease the effect of random parameter initialization, we trained 5 different versions of each model, using different random seeds.

TABLE V

NUSCENES: VALIDATION SPLIT RESULTS ACROSS THE FIVE SEEDS.

Models	minADE	minFDE	indep. NLL	joint NLL
T++	1.96 ± 0.02	4.05 ± 0.07	10.28 ± 0.29	43.18 ± 1.01
PECNet	26.73 ± 2.30	40.04 ± 2.36	$6.7e^5$ $\pm 1.0e^5$	$1.8e^6$ $\pm 2.9e^5$
MID	6.47 ± 0.29	13.48 ± 0.82	71.04 ± 7.97	$0.3e^3$ ± 23.27
FM	12.61 ± 1.07	23.55 ± 2.02	$0.3e^3$ $\pm 0.2e^3$	$0.9e^3$ $\pm 0.7e^3$
TF w/o AE	13.15 ± 0.18	24.71 ± 0.29	$0.9e^3$ $\pm 0.6e^3$	$1.9e^3$ $\pm 1.3e^3$
TF (Ours)	1.86 ± 0.06	3.72 ± 0.19	16.54 ± 1.04	45.55 ± 2.72

C. Results

On ETH/UCY, we observed the same trend as in the synthetic datasets. The three NF-based methods were able to better fit the underlying data distribution compared to the methods which do not employ NFs. Out of these, *TrajFlow* clearly outperformed existing methods as well as its ablation in terms of distribution fit as captured by both NLL metrics. (Tab. III) This is especially clear in the joint NLL, indicating that the learned distributions were also better able to capture the interactions among agents in a scene. We found that the state-of-the-art methods also performed worse even compared to the originally reported values, such as in the case of *T++* [12]. It is, however, important to note that this is not the first time the original *T++* results could not be replicated (see e.g. [41]), and compared to common practice, we used a stricter method for extracting testing samples, necessitating the existence of all 12 future positions.

On rounD (Tab. IV), *TrajFlow* was able to achieve the best results. When compared to its ablation case, there is a clear performance boost both in terms of the learned distribution but also in terms of the distance metrics. This is especially notable since in rounD the agents being predicted are vehicles, not pedestrians. As a result, the distance crossed is generally larger and thus prediction errors tend to accumulate faster. In terms of the extrapolation performance, out of all the tested models, *TrajFlow* achieved the best minADE value with $2.27\,\mathrm{m}^{\pm 0.30}$. This was closely followed by *TF w/o AE* and *T++* with values of $2.53\,\mathrm{m}^{\pm 0.52}$ and $2.58\,\mathrm{m}^{\pm 0.55}$ respectively. *FloMo* achieved an error of $2.62\,\mathrm{m}^{\pm 0.59}$, while *MID* and *PECNet* achieved an error of $10.10\,\mathrm{m}^{\pm 1.26}$ and $19.09\,\mathrm{m}^{\pm 1.10}$ respectively. It is worth noting that out of these methods, only *TrajFlow* and *T++* have auto-regressive capability while the remaining models require a separate extrapolation method.

Finally, on nuScenes (Tab. V), *TrajFlow* outperformed all of the models in terms of the distance metrics minADE and minFDE. It was, however, outperformed on the distribution metrics by *T++*. What is notable, however, is that although *TrajFlow* had poorer performance than *T++* on the independent NLL, the performance of the two models on the joint NLL was comparable. This indicates that while *TrajFlow* does not capture the individual distributions as precisely as *T++* on nuScenes, the distributions learned manage to reflect the overall behavior in a scene to a similar extent as *T++*.

VII. CONCLUSION

In this work, we proposed *TrajFlow*, a novel model for predicting the trajectories of human agents in traffic by

applying Normalizing Flows to the latent abstraction of the future trajectories to be predicted.

Tests carried out on synthetic examples, which contained sets of grounds truths for a single input, showed that Normalizing Flow-based methods outperformed several state-of-the-art methods in terms of the distribution fits. Furthermore, among these NF-based models, *TrajFlow* had the best performance thanks to incorporating a Recurrent Neural Network-based Autoencoder.

Through evaluations on the ETH/UCY, rounD, and nuScenes datasets, we found that our model can successfully learn distributions within real-world datasets. *TrajFlow* achieved competitive results compared to state-of-the-art methods on all three datasets, and was able to clearly outperform existing methods in terms of the distribution fit on the pedestrian dataset ETH/UCY. This is particularly noteworthy since pedestrian behavior is less structured than that of vehicles and one can in turn expect a higher amount of variability.

We further observed that the introduction of an RNN-AE does not lead to a severe degradation of our predicted trajectories despite the loss of information inherent to data compression, and in fact, for rounD and nuScenes, learning over abstracted trajectory features proved to be beneficial. Tests on rounD also showed that the auto-regressive nature of our decoder provides further flexibility in terms of the possible length of the predictions and is even able to outperform state-of-the-art models in terms of prediction accuracy. This is particularly useful for cases that need a longer planning horizon to ensure safety and comfort such as when approaching a roundabout or when interacting with pedestrians close to a crosswalk.

Future work will explore ways to improve prediction quality for more structured environments such as in the case of vehicle trajectory prediction. Another point of focus will be to expand the model towards being able to provide joint predictions of all agents in a given scene. Finally, *TrajFlow*'s capacity to capture multi-modal distributions can be utilized in contingency planners which account for multiple possible outcomes [42]. The extent to which better distribution fitting is beneficial to such planners was outside of the scope of this work and should be investigated in future work.

Overall, our results indicate that *TrajFlow* compares favorably to state-of-the-art behavior prediction models in learning trajectory distributions both from highly variable data (e.g., pedestrian trajectories) as well as more constrained scenarios (such as in the case of vehicle trajectories). This model thus has potential for application in settings where AVs have to navigate in settings with human road users in order to generate more natural and safer plans.

APPENDIX

THE ENCODING OF PAST BEHAVIOR

A. Past Trajectories

The encoder of the past behavior ϕ_{RNN} encodes the past trajectory x_{tar} of the single target agent whose future is to be predicted. This function was taken from the implementation of [24] and is identical to the encoder E_{RNN} (see Sec. III-B.2), except that instead of $d = M = m = 20$, we used $d = M = m = 64$ for the TrajFlow variants and $d = M = m = 16$ in FloMo, as per the original implementation.

B. Static Environment

For encoding a gray-scale image of the static environment E, which has been rotated to align with the target agent's heading, we used a CNN function ϕ_{CNN}. For this, $L_{\text{CNN}} = 3$ convolutional layers $\phi^{(l)}_{\text{CNN}}$ are used within this network with a kernel of size 5 and a stride of 4. The first two layers additionally have a zero-padding of size 1 around the image. With this, an initial input of size $h^{(0)} \times w^{(0)} = 156 \times 257$ and $c^{(0)} = 1$ channel is transformed first into a representation with $c^{(1)} = 8$ and $h^{(1)} \times w^{(1)} = 39 \times 64$, then into a representation with $c^{(2)} = 16$ and $h^{(2)} \times w^{(2)} = 10 \times 16$ and lastly into an output representation with $c^{(3)} = 32$ and $h^{(3)} \times w^{(3)} = 2 \times 3$. This output $\phi^{(3)}_{\text{CNN}}$ is then flattened and passed through a two-layer dense network. The first linear layer transforms the input into a hidden state of length 128, while the second linear layer produces the final encoding of the image of size $M_{\text{CNN}} = 64$.

C. Social Interactions

To encode interactions, we use a GNN function ψ_{GNN} that processes all past trajectories $\boldsymbol{x} = \{\boldsymbol{x}_{\text{tar}}, \boldsymbol{x}_1, \dots \boldsymbol{x}_{n-1}\}$. There, in the first step, the past trajectory $\boldsymbol{x}_a$ of each agent a of the n agents is encoded using a GRU-based function $\psi_{\text{RNN},c}$ (structure is identical to ϕ_{RNN}; Sec. VII-A). This network is shared between all agents of each class $c \in C = \{\text{veh.}, \text{ped.}, \dots\}$, i.e. there is for instance one network $\psi_{\text{RNN, veh.}}$ to encode the past of vehicles. A linear embedding layer $\psi_{\text{em}} : \mathbb{R}^m \to \mathbb{R}^{M_{\text{GNN}}}$ is then applied to each encoded past trajectory, with $\widetilde{\boldsymbol{x}}^{(0)}_a = \psi_{\text{em}}\left(\psi_{\text{RNN},c_a}\left(\boldsymbol{x}_a\right)\right)$.

In the GNN, each of the n agents is seen as a node, with n^2 unidirectional edges being established between all nodes. Based on this, L_{GNN} layers $\psi^{(l)}_{\text{GNN}}$ are applied to this network to update the node states $\widetilde{\boldsymbol{x}}^{(l)} = \left\{\widetilde{\boldsymbol{x}}^{(l)}_{\text{tar}}, \widetilde{\boldsymbol{x}}^{(l)}_1, \dots, \widetilde{\boldsymbol{x}}^{(l)}_{n-1}\right\}$:

$$\widetilde{\boldsymbol{x}}^{(l)} = \psi^{(l)}_{\text{GNN}}\left(\widetilde{\boldsymbol{x}}^{(l-1)}\right)$$

The update starts with calculating the message $\boldsymbol{m}^{(l)}_{b,a}$ from agent b to agent a for every possible connection, using the message network $\psi_{\text{M}} : \mathbb{R}^{2\,M_{\text{GNN}}+2\,|C|+1} \to \mathbb{R}^{M_{\text{GNN}}}$:

$$\boldsymbol{m}^{(l)}_{b,a} = \psi_{\text{M}}\left(\widetilde{\boldsymbol{x}}^{(l-1)}_b \oplus \widetilde{\boldsymbol{x}}^{(l-1)}_a \oplus \mathcal{C}_b \oplus \mathcal{C}_a \oplus \|\boldsymbol{x}_a - \boldsymbol{x}_b\|\right),$$

where the last three terms are the graph's edge features between agents a and b, with $\mathcal{C}_a, \mathcal{C}_b \in \mathbb{R}^{|C|}$ being the one-hot encoding of class c_a and c_b respectively. Those incoming messages are then aggregated at each node:

$$\boldsymbol{m}^{(l)}_a = \sum_b \boldsymbol{m}^{(l)}_{b,a}$$

Lastly, the state of each node is updated, using an update network $\psi_{\text{U}} : \mathbb{R}^{2\,M_{\text{GNN}}} \to \mathbb{R}^{M_{\text{GNN}}}$

$$\widetilde{\boldsymbol{x}}^{(l)}_a = \psi^{(l)}_{\text{U}}\left(\widetilde{\boldsymbol{x}}^{(l-1)}_a \oplus \boldsymbol{m}^{(l)}_a\right) + \widetilde{\boldsymbol{x}}^{(l-1)}_a$$

After being propagated through all L_{GNN} layers $\psi_{\text{GNN}}^{(l)}$, the final output of ψ_{GNN} is

$$\frac{1}{n}\sum_i \widetilde{\boldsymbol{x}}_a^{(L_{\text{GNN}})}$$

For our work, we chose to set $L_{\text{GNN}} = 4$ and $M_{\text{GNN}} = 32$.

REFERENCES

[1] J. S. Brar and B. Caulfield, "Impact of autonomous vehicles on pedestrians' safety," in *IEEE 20th Int. Conf. on Intell. Transp. Syst. (ITSC)*, 2017.

[2] J. Meyer, H. Becker, P. M. Bösch, and K. W. Axhausen, "Autonomous vehicles: The next jump in accessibilities?," *Research Transp. Econ.*, vol. 62, pp. 80–91, 2017.

[3] J. Pisarov and G. Mester, "The future of autonomous vehicles," *FME Trans.*, vol. 49, no. 1, pp. 29–35, 2021.

[4] M. Milford, S. Anthony, and W. Scheirer, "Self-driving vehicles: Key technical challenges and progress off the road," *IEEE Potentials*, vol. 39, no. 1, pp. 37–45, 2019.

[5] B. Brown, M. Broth, and E. Vinkhuyzen, "The halting problem: Video analysis of self-driving cars in traffic," in *Proc. 2023 CHI Conf. on Hum. Factors Comput. Syst.*, pp. 1–14, 2023.

[6] J. F. Schumann, J. Kober, and A. Zgonnikov, "Benchmarking behavior prediction models in gap acceptance scenarios," *IEEE Trans. on Intell. Veh.*, 2023.

[7] G. M. Ferro and D. Sornette, "Stochastic representation decision theory: How probabilities and values are entangled dual characteristics in cognitive processes," *Plos one*, vol. 15, no. 12, p. e0243661, 2020.

[8] B. Varadarajan, A. Hefny, A. Srivastava, K. S. Refaat, N. Nayakanti, A. Cornman, K. Chen, B. Douillard, C. P. Lam, D. Anguelov, *et al.*, "MultiPath++: Efficient information fusion and trajectory aggregation for behavior prediction," in *Int. Conf. on Robotics Autom.*, 2022.

[9] K. Messaoud, N. Deo, M. M. Trivedi, and F. Nashashibi, "Trajectory prediction for autonomous driving based on multi-head attention with joint agent-map representation," in *IEEE Intell. Veh. Symp.*, 2021.

[10] A. Gupta, J. Johnson, L. Fei-Fei, S. Savarese, and A. Alahi, "Social GAN: Socially acceptable trajectories with generative adversarial networks," in *Conf. on Comput. Vis. Pattern Recognit.*, 2018.

[11] J. Amirian, J.-B. Hayet, and J. Pettré, "Social ways: Learning multi-modal distributions of pedestrian trajectories with GANs," in *Proc. IEEE/CVF Conf. on Comput. Vis. Pattern Recognit. Work.*, 2019.

[12] T. Salzmann, B. Ivanovic, P. Chakravarty, and M. Pavone, "Trajectron++: Dynamically-feasible trajectory forecasting with heterogeneous data," in *Eur. Conf. on Comput. Vis.*, 2020.

[13] Y. Yuan, X. Weng, Y. Ou, and K. M. Kitani, "AgentFormer: Agent-aware transformers for socio-temporal multi-agent forecasting," in *Proc. IEEE/CVF Int. Conf. on Comput. Vis.*, 2021.

[14] K. Mangalam, H. Girase, S. Agarwal, K.-H. Lee, E. Adeli, J. Malik, and A. Gaidon, "It is not the journey but the destination: Endpoint conditioned trajectory prediction," in *Eur. Conf. on Comput. Vis.*, 2020.

[15] T. Gu, G. Chen, J. Li, C. Lin, Y. Rao, J. Zhou, and J. Lu, "Stochastic trajectory prediction via motion indeterminacy diffusion," in *Proc. IEEE/CVF Conf. on Comput. Vis. Pattern Recognit.*, 2022.

[16] B. F. de Brito, H. Zhu, W. Pan, and J. Alonso-Mora, "Social-VRNN: One-shot multi-modal trajectory prediction for interacting pedestrians," in *Conf. on Robot Learn.*, 2021.

[17] L. Janson, E. Schmerling, and M. Pavone, "Monte Carlo motion planning for robot trajectory optimization under uncertainty," in *Robotics Research*, pp. 343–361, Springer, 2018.

[18] E. G. Tabak and E. Vanden-Eijnden, "Density estimation by dual ascent of the log-likelihood," *Commun. Math. Sci.*, vol. 8, no. 1, pp. 217–233, 2010.

[19] E. G. Tabak and C. V. Turner, "A family of nonparametric density estimation algorithms," *Commun. on Pure Appl. Math.*, vol. 66, no. 2, pp. 145–164, 2013.

[20] N. Rhinehart, K. M. Kitani, and P. Vernaza, "R2p2: A reparameterized pushforward policy for diverse, precise generative path forecasting," in *Proc. Eur. Conf. on Comput. Vis. (ECCV)*, 2018.

[21] N. Rhinehart, R. McAllister, K. Kitani, and S. Levine, "Precog: Prediction conditioned on goals in visual multi-agent settings," in *Proc. IEEE/CVF Int. Conf. on Comput. Vis.*, 2019.

[22] A. Bhattacharyya, C.-N. Straehle, M. Fritz, and B. Schiele, "Haar wavelet based block autoregressive flows for trajectories," in *DAGM German Conf. on Pattern Recognit.*, 2020.

[23] J. Sun, Z. Wang, J. Li, and C. Lu, "Unified and fast human trajectory prediction via conditionally parameterized normalizing flow," *IEEE Robotics Autom. Lett.*, vol. 7, no. 2, pp. 842–849, 2021.

[24] C. Schöller and A. Knoll, "FloMo: Tractable motion prediction with normalizing flows," in *Int. Conf. Intell. Robot. Syst.*, 2021.

[25] J. Liang, L. Jiang, K. Murphy, T. Yu, and A. Hauptmann, "The garden of forking paths: Towards multi-future trajectory prediction," in *Proc. IEEE/CVF Conf. on Comput. Vis. Pattern Recognit.*, 2020.

[26] S. Pellegrini, A. Ess, K. Schindler, and L. Van Gool, "You'll never walk alone: Modeling social behavior for multi-target tracking," in *2009 IEEE 12th Int. Conf. on Comput. Vis.*, 2009.

[27] A. Lerner, Y. Chrysanthou, and D. Lischinski, "Crowds by example," in *Comput. Graph. Forum*, vol. 26, 2007.

[28] R. Krajewski, T. Moers, J. Bock, L. Vater, and L. Eckstein, "The rounD dataset: A drone dataset of road user trajectories at roundabouts in Germany," in *IEEE 23rd Int. Conf. on Intell. Transp. Syst.*, 2020.

[29] H. Caesar, V. Bankiti, A. H. Lang, S. Vora, V. E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan, and O. Beijbom, "nuScenes: A multimodal dataset for autonomous driving," in *Proc. IEEE/CVF Conf. on Comput. Vis. Pattern Recognit.*, 2020.

[30] G. Papamakarios, E. T. Nalisnick, D. J. Rezende, S. Mohamed, and B. Lakshminarayanan, "Normalizing flows for probabilistic modeling and inference.," *J. Mach. Learn. Res.*, vol. 22, no. 57, pp. 1–64, 2021.

[31] Y. Lu and B. Huang, "Structured output learning with conditional generative flows," in *Proc. AAAI Conf. on Artif. Intell.*, 2020.

[32] O. Siebinga, A. Zgonnikov, and D. Abbink, "Uncovering variability in human driving behavior through automatic extraction of similar traffic scenes from large naturalistic datasets," in *2023 IEEE Int. Conf. on Syst. Man, Cybern. (SMC)*, pp. 4790–4796, 2023.

[33] Z. Cao, E. Biyik, G. Rosman, and D. Sadigh, "Leveraging smooth attention prior for multi-agent trajectory prediction," in *2022 Int. Conf. on Robotics Autom. (ICRA)*, 2022.

[34] Q. Song, W. Wang, W. Fu, Y. Sun, D. Wang, and Z. Gao, "Research on quantum cognition in autonomous driving," *Sci. reports*, vol. 12, no. 1, p. 300, 2022.

[35] K. Cho, B. van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using RNN encoder–decoder for statistical machine translation," in *Conf. on Empir. Methods Natural Language Process.*, 2014.

[36] J. Martinez, M. J. Black, and J. Romero, "On human motion prediction using recurrent neural networks," in *Proc. IEEE Conf. on Comput. Vis. Pattern Recognit.*, 2017.

[37] L. Yang, Z. Zhang, Y. Song, S. Hong, R. Xu, Y. Zhao, W. Zhang, B. Cui, and M.-H. Yang, "Diffusion models: A comprehensive survey of methods and applications," *ACM Comput. Surv.*, vol. 56, no. 4, pp. 1–39, 2023.

[38] J. Lin, "Divergence measures based on the Shannon entropy," *IEEE Trans. on Inf. Theory*, vol. 37, no. 1, pp. 145–151, 1991.

[39] A. Mészáros, J. F. Schumann, J. Alonso-Mora, A. Zgonnikov, and J. Kober, "Rome: Robust multi-modal density estimator," *Proc. 33rd Int. Jt. Conf. on Artif. Intell.*, 2024, in press.

[40] J. F. Schumann, A. R. Srinivasan, J. Kober, G. Markkula, and A. Zgonnikov, "Using models based on cognitive theory to predict human behavior in traffic: A case study," in *2023 IEEE 20th Int. Conf. on Intell. Transp. Syst. (ITSC)*, 2023.

[41] F. S. Westerhout, J. F. Schumann, and A. Zgonnikov, "Smooth-Trajectron++: Augmenting the Trajectron++ behaviour prediction model with smooth attention," in *2023 IEEE 20th Int. Conf. on Intell. Transp. Syst. (ITSC)*, 2023.

[42] N. Rhinehart, J. He, C. Packer, M. A. Wright, R. McAllister, J. E. Gonzalez, and S. Levine, "Contingencies from observations: Tractable contingency planning with learned behavior models," in *2021 IEEE Int. Conf. on Robotics Autom. (ICRA)*, pp. 13663–13669, IEEE, 2021.