End-to-End Trajectory Distribution Prediction Based on Occupancy Grid Maps

Ke Guo[1], Wenxi Liu[2], Jia Pan[1] *

[1]The University of Hong Kong   [2]College of Computer and Data Science, Fuzhou University

{kguo,jpan}@cs.hku.hk, wenxi.liu@hotmail.com

Abstract

In this paper, we aim to forecast a future trajectory distribution of a moving agent in the real world, given the social scene images and historical trajectories. Yet, it is a challenging task because the ground-truth distribution is unknown and unobservable, while only one of its samples can be applied for supervising model learning, which is prone to bias. Most recent works focus on predicting diverse trajectories in order to cover all modes of the real distribution, but they may despise the precision and thus give too much credit to unrealistic predictions. To address the issue, we learn the distribution with symmetric cross-entropy using occupancy grid maps as an explicit and scene-compliant approximation to the ground-truth distribution, which can effectively penalize unlikely predictions. In specific, we present an inverse reinforcement learning based multi-modal trajectory distribution forecasting framework that learns to plan by an approximate value iteration network in an end-to-end manner. Besides, based on the predicted distribution, we generate a small set of representative trajectories through a differentiable Transformer-based network, whose attention mechanism helps to model the relations of trajectories. In experiments, our method achieves state-of-the-art performance on the Stanford Drone Dataset and Intersection Drone Dataset.

1. Introduction

Trajectory prediction has gained increasing attention due to its emerging applications such as robot navigation and self-driving cars. Due to the inherent multimodal uncertainty from an agent's intention or environment, a large number of works have been proposed to learn a multimodal distribution of the future trajectories. For example, in [20, 26], the multimodal distribution is explicitly modeled using the Gaussian mixture model, though it is hard to
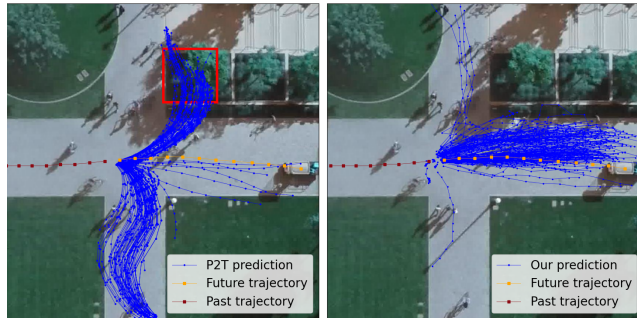
Figure 1. Illustration of trajectory prediction distributions of P2T and our method on Stanford Drone Dataset. Although the prediction of P2T is more diverse, it predicts many infeasible outcomes (e.g., those trajectories intersecting with the parterre) and assigns too high probability to the turning action.

optimize and prone to overfitting. Others have attempted to model the trajectory distribution implicitly using generative models such as conditional variational autoencoder (CVAE) [6, 19, 28, 44], normalizing flow (NF) [34, 38], or generative adversarial network (GAN) [2, 8, 9, 11, 43].

However, most previous works focus on the diversity of the predicted trajectories rather than the more important precision, except a few works (*e.g.* [34, 38]). The issue is that if the model is only encouraged to cover all modes of real distribution, it may assign too many probabilities to unrealistic predictions and cannot accurately reflect the real probability density. One such example is shown in Fig. 1 where a large portion of the diverse trajectories predicted by P2T [6] turn and intersect with obstacles, which are certainly implausible and inconsistent with common knowledge that moving straight ahead is more likely than turning. In such circumstances, a navigation decision based on the predictions will overreact to less likely futures, while underestimating the more likely ones.

Specifically, to learn a diverse trajectory distribution, previous works usually minimize the variety loss [6, 9, 13] or the forward cross-entropy [23, 26, 44]. Yet, the variety loss does not penalize bad predictions as long as there exists one prediction close to the ground-truth, and it does not lead to ground-truth distribution but approximately its square root [46]. On the other hand, the forward cross-

entropy also fails to adequately penalize the unlikely predictions [34,38] and exhibits noise sensitivity [48]. To overcome the limitations of these losses, our solution is to learn a distribution minimizing the symmetric cross-entropy, *i.e.* the combination of forward and reverse cross-entropy between the predictive distribution and ground-truth distribution. Compared with the forward cross-entropy, the reverse cross-entropy can penalize the prediction with low likelihood, but it requires ground-truth distribution as a reference, which unfortunately is not available in many cases. An effective solution is to employ an occupancy grid map (OGM), which divides the social space into grid cells with an occupancy probability in each cell. Thus, the trajectory probability can be approximated as the product of all future position probabilities conditioned on the OGM. In [38], an OGM, parameterized as a cost map, is embedded from spatial scene features by a convolutional neural network (CNN) to assign proper probabilities to different social areas. However, representing all future position distributions with a single OGM is inaccurate, since it neglects the spatial-temporal correspondence of trajectories. Instead, we predict an OGM for each future position with a convolutional long short-term memory (ConvLSTM) [51] network based on our novel deconvolution parameterization of the position probability flow. The resulting dynamic OGMs can help not only the trajectory prediction [23] but also downstream planning tasks [4,53].

When minimizing the symmetric cross-entropy, previous approaches [34,38] usually make use of the normalizing flow, which transforms a simple Gaussian distribution into the target trajectory distribution through a sequence of auto-regressive mappings. These mappings are required to be invertible, differentiable, and easy for computing Jacobian determinants, which are difficult to be satisfied in practice. In addition, the latent variable sampled from the Gaussian distribution is hard to interpret. To address these issues, we develop an end-to-end interpretable model to backpropagate the symmetric cross-entropy loss. In particular, we construct a CVAE model using a coarse future trajectory plan within neighboring grids as the interpretable latent variable, similar to P2T [6]. However, P2T cannot be trained in an end-to-end manner, because it learns the planning policy using the maximum-entropy inverse reinforcement learning (MaxEnt IRL) [50,58] by matching feature expectation. Instead, we implement value iteration in IRL by differentiable value iteration network (VIN) [45] and incorporate Gumbel-Softmax [15] into the discrete planning policy sampling. In our VIN-based IRL, planning and trajectory generation policy can be learned simultaneously by maximizing the data likelihood.

Even though a large number of possible future trajectories can be sampled from the learned distribution, many downstream applications often demand a small set of representative predictions. This requirement is traditionally accomplished by learning the distribution model with the variety loss [5,9,13] or post-processing with heuristic methods like greedy approximation [36] or K-means [6,7]. Motivated by the insight that clustering like K-means can be regarded as paying different attention to different samples, we propose a Transformer-based refinement network, whose attention mechanism can also ensure sampling diversity, to attentively obtain a small set of representative samples from the over-sampled outcomes of our prediction model. The representative properties can be conveniently adjusted by its loss, *e.g.* the variety loss for diversity. In experiments, we compare our method with a set of state-of-the-art approaches on the Stanford Drone Dataset [40] and Intersection Drone Dataset [3] and demonstrate the superiority of our method in both trajectory diversity and quality.

In summary, the main contributions are as follows.

- We propose a VIN-based IRL method, simplifying the learning process while allowing the gradients from trajectory generation to flow back to the planning module.
- We improve the approximation of ground-truth with OGMs in learning trajectory distribution using symmetric cross-entropy.;
- We introduce a Transformer-based refinement network for sampling from trajectory distribution to obtain representative and realistic trajectories;
- We demonstrate the state-of-the-art performance of our framework on two real-world datasets: Stanford Drone dataset [40] and Intersection Drone dataset [3].

2. Related Work

2.1. Trajectory Distribution Prediction

We focus on trajectory distribution prediction approaches based on deep learning. Refer to [41] for a survey of more classical approaches. In the early literature, the trajectory distribution is usually modeled as a simple unimodal distribution, *e.g.* a bivariate Gaussian distribution [1,25,54]. However, the unimodal models tend to predict the average of all possible modes, which may be not valid.

Recently, various generative models such as GAN, NF and CVAE, have been proposed to address the multi-modality, which capture the stochasticity with a latent variable. GAN-based methods [2,8,9,11,43] use a discriminator to generate diverse realistic trajectories but are difficult to train and suffer from the mode collapse. NF methods [34,38] sample the latent variable from a standard Gaussian distribution and map it to the target trajectory through a sequence of transformations. Some CVAE approaches such as DESIRE [19], Trajectron++ [44], learn a Gaussian or categorical latent distribution using the constraint between prior and posterior distribution. Others leverage an interpretable latent variable to incorporate prior knowledge. For

example, PECNet [28], TNT [55] consider the destination position as the latent variable. Further, LB-EBM [33] takes positions at several future steps as the latent variable which is sampled from an energy-based model. P2T [6] samples a coarse grid plan from a policy learned by deep MaxEnt IRL [50] as the latent variable. Even though our model also leverages the plan as the latent variable, we learn the plan and trajectory distributions in a unified framework.

2.2. Occupancy Grid Maps Prediction

OGMs prediction works aim at predicting the categorical occupancy distribution over a grid at each future time step. Even though there are extensive studies forecasting OGMs of a crowd [21, 31] or all objects [24, 32] in a scene, we focus on reviewing the literature predicting one agent's OGMs like our work.

Kim *et al*. [16] directly outputs the future probability of each grid cell using the LSTM network. Y-net [27] yields the OGM at each future step directly from different channels of the feature map output by a CNN. Similarly, in MP3 [4], the feature map at each channel is embedded into the temporal motion fields at each future step, which obtains the probability transition flow between consecutive OGMs by bilinear interpolation of motion vectors on the field. To take advantage of the temporal and spatial patterns in the sequential OGMs, ConvLSTM [51] is widely applied. In [39], they directly derive an OGM from the hidden map of ConvLSTM at each time step. To increase time consistency, DRT-NET [14] learns the residual provability flow between the consecutive OGMs. To incorporate the prior knowledge of local movement, Multiverse [23] uses a graph attention network to aggregate neighborhood information on the hidden map of ConvLSTM. Similarly, SA-GNN [24] considers the interactions with neighbors by graph neural networks. Based on the ConvLSTM and deconvolution parameterization, our method is not only computationally efficient but also explicitly models the local transition probability.

Furthermore, some of these works attempt to obtain trajectories by sampling the OGMs. But the positions independently sampled from each OGM suffer from discretization errors and lack spatio-temporal correspondence in trajectories. To address this problem, [39] leverages the OGMs as input to another ConvLSTM which outputs the coordinates of a fixed number of future trajectories. Multiverse [23] predicts a continuous offset at each cell to mitigate the discretization error and applies a diverse beam search to generate multiple distinct trajectories. Y-net [27] samples intermediate positions conditioned on the sampled goal and waypoints. Instead of sampling OGMs like all previous works, we use the OGMs as auxiliary information in training loss to generate more feasible trajectories.

2.3. Trajectory Sample Refinement

Trajectories sampled from the predicted trajectory distribution usually do not satisfy the downstream requirement. The most common two requirements are precision and diversity to cover all future scenarios accurately [34, 38]. To improve accuracy, previous works [19, 29, 55] usually score the samples using a neural network and refine the top samples. For diversity, the relation between samples needs to be considered. Most literature [5, 9, 13] directly use a variety loss to improve diversity. In addition, P2T [6], PGP [7] and Y-net [27] use K-means to cluster samples while CoverNet [36] employs a greedy approximation algorithm to create a diverse set. To capture both diversity and quality, DSF [52] learns a diverse sampling function to sample the latent variable of CVAE using a diversity loss based on a determinantal point process at test time while DiversityGAN [12] samples distinct latent semantic variables to predict diverse trajectories. Different from previous work, our sample refinement network based on Transformer is an independent and differentiable module and is flexible with the downstream requirement and trajectory sample number.

3. Background

3.1. Problem Formulation

Given an observation Ω including a context and history trajectory $X = \{X_t \in \mathbb{R}^2 \mid t = -t_p + 1, \ldots, 0\}$ of a target agent, our objective is to predict the distribution $p(Y|\Omega)$ of its future trajectory $Y = \{Y_t \in \mathbb{R}^2 \mid t = 1, \ldots, t_f\}$. The context consists of neighbors' history trajectories and an image $\mathbf{I}$, which is a bird's eye view (BEV) perception of the local scene centered at the agent's current position.

We assume that an agent has a grid-based plan on which its future trajectory is conditioned. An agent's planning process is modeled using a Markov decision process (MDP) $\mathcal{M} = \{\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathbf{r}\}$, with a time horizon N. A state set $\mathcal{S}$ consists of all cells over a 2D grid and an absorbing end state of zero value. An action set $\mathcal{A}$ includes 4 adjacent movements $up, down, left, right$ and an end action leading to the absorbing state. A deterministic transition function $\mathcal{T} : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$ describes system dynamics. A non-stationary reward function $\mathbf{r}^n : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ determines a reward for each state and action per step n. We assume that the agent uses a non-stationary stochastic policy $\boldsymbol{\pi}^n(a|s)$ to determine the probability of selecting an action a at a state s at MDP step n, and finally make a plan in terms of the state sequence $S = \{s^n \in \mathcal{S} \mid n = 1, \ldots, N\}$. Note that here we are using the superscript n as the MDP step n, to distinguish with the time step t as subscript.

To relieve the difficulty of modeling the multi-modal future trajectory distribution $p(Y|\Omega)$, we introduce the plau-

sible plan as the latent variable and decompose it as:

$$p(Y|\Omega) = \int_{S \in \mathbb{S}(\Omega)} p(S|\Omega) \, p(Y|S, \Omega) \, dS,$$

where $\mathbb{S}(\Omega)$ is the space of plausible plans conditioned on the observation. In this way, since the plan uncertainty can well capture the multimodality, trajectory conditioned on a plan can be well approximated as a unimodal distribution.

3.2. Trajectory Distribution Learning

We predict the future trajectory distribution by minimizing the discrepancy between the distribution $q_\theta(\hat{Y}|\Omega)$ of the predicted trajectory $\hat{Y}$ and the ground-truth distribution $p(Y|\Omega)$. As a straightforward distance metric between these two distributions, forward cross-entropy (a.k.a, negative log-likelihood (NLL)) is computed as:

$$\mathcal{H}(p, q_\theta) = - \mathop{\mathbb{E}}_{\Omega \sim \Psi, Y \sim p(\cdot|\Omega), S \in \mathbb{S}(Y)} [\log q_\theta(S|\Omega) q_\theta(Y|\Omega, S)],$$

where Ψ denotes the ground-truth observations' distribution and $\mathbb{S}(Y)$ is the space containing the ground-truth plan S, *i.e.* the grid state sequence the trajectory Y goes through.

Although the NLL loss encourages the predicted distribution to cover all plausible modes of the ground-truth distribution, it assigns a low penalty to the implausible predictions which are less likely to take place under the ground-truth distribution [34, 38]. The reverse cross-entropy $\mathcal{H}(q_\theta, p)$ can evaluate the likelihood of the prediction under the ground-truth distribution and penalize unlikely predictions, but the ground-truth distribution p is unknown in the real world with only one sample observed. To address this issue, we approximate the continuous joint distribution $p(Y|\Omega)$ of future trajectory as a product of future positions' categorical marginal distributions $\mathbf{O} = \{\mathbf{O}_t \mid t = 1, \ldots, t_f\}$, represented as OGMs:

$$p(Y|\Omega) \approx p(\mathbf{O}|\Omega) \prod_{t=1}^{t_f} \mathbf{O}_t(Y_t),$$

where $\mathbf{O}_t(Y_t)$ denotes the agent's location probability at Y_t at time t, which is bilinearly interpolated from nearby probabilities on $\mathbf{O}_t$ and $p(\mathbf{O}|\Omega)$ is assumed to be deterministic and parameterized by neural networks $\mathbf{O} = o_\alpha(\Omega)$. Thus, the reverse cross-entropy $\mathcal{H}(q_\theta, p)$ can be approximated as:

$$\mathcal{H}(q_\theta, \mathbf{O}) = - \mathop{\mathbb{E}}_{\Omega \sim \Psi, \hat{Y} \sim q_\theta(\cdot|\Omega)} \log p(\mathbf{O}|\Omega) \prod_{t=1}^{t_f} \mathbf{O}_t(\hat{Y}_t).$$

4. Approach

As shown in Fig. 2, our model is composed of five modules that can be learned in an end-to-end manner: an **Observation Encoder**, a **Policy Network**, an **Occupancy Grid Maps Decoder (OGMs Decoder)**, a **Trajectory Decoder** and a **Refinement Network**.

4.1. Observation Encoder

The first component of our approach is an observation encoder composed of a motion encoder to extract motion features from the past trajectories of the target and its neighbors and a scene encoder to extract scene features from the BEV image of the surrounding environment.

Motion encoder: The motion encoder is designed to embed the past trajectories of the target agent and its neighbors into a feature vector and a feature map. To represent the neighbors' state succinctly, we leverage a directional pooling grid from [18], where each cell contains the relative velocity of a neighbor located in that cell with respect to the target agent. At each past time step t, we first flatten the grid into a vector d_t and then concatenate the vector with the agent velocity $X_t - X_{t-1}$ as input to an RNN. The hidden state of the RNN at time t is given by:

$$m_t = \text{RNN}_{\text{m}}(m_{t-1}, \phi[d_t, X_t - X_{t-1}]),$$

where ϕ is a linear embedding layer and the brackets indicate concatenation. The first hidden state m_{-t_p+1} is set to zero and the last hidden state m_0 is regarded as the motion feature. The m_0 is duplicated over all cells in the scene and then is concatenated with each cell's agent-centered, world-aligned coordinate to construct a motion feature map $\mathbf{M}$:

$$\mathbf{M}(x, y) = [m_0, x, y].$$

Scene encoder: We apply a CNN to extract a scene feature map from the BEV image $\mathbf{I}$ of the neighborhood:

$$\mathbf{F} = \text{CNN}_{\text{f}}(\mathbf{I}),$$

where the spatial dimensions of the scene feature map $\mathbf{F}$ are the same as that of the MDP grid for simplicity.

4.2. Policy Network

We generate a policy in two steps end-to-end: mapping the observation features into rewards and then computing a policy with a value iteration network.

We adopt non-stationary rewards to capture the dynamic agent-to-scene and agent-to-agent interaction. Based on the scene and motion feature maps, a ConvLSTM architecture is applied to yield the reward map at each step. The ConvLSTM hidden map and the reward map at MDP step n are:

$$\mathbf{H}^n = \text{ConvLSTM}_{\text{r}}(\mathbf{H}^{n-1}, \mathbf{F}), \quad \mathbf{r}^n = \Phi(\mathbf{H}^n),$$

where Φ is a fully connected convolutional layer. The initial hidden map $\mathbf{H}^0$ is the embedded motion feature map $\Phi(\mathbf{M})$.

Based on the reward maps, we use the approximate value iteration to generate a policy map $\boldsymbol{\pi}^n$ at each step n. To back-propagate the loss through the value iteration, we take advantage of the value iteration network as [35, 37, 45],
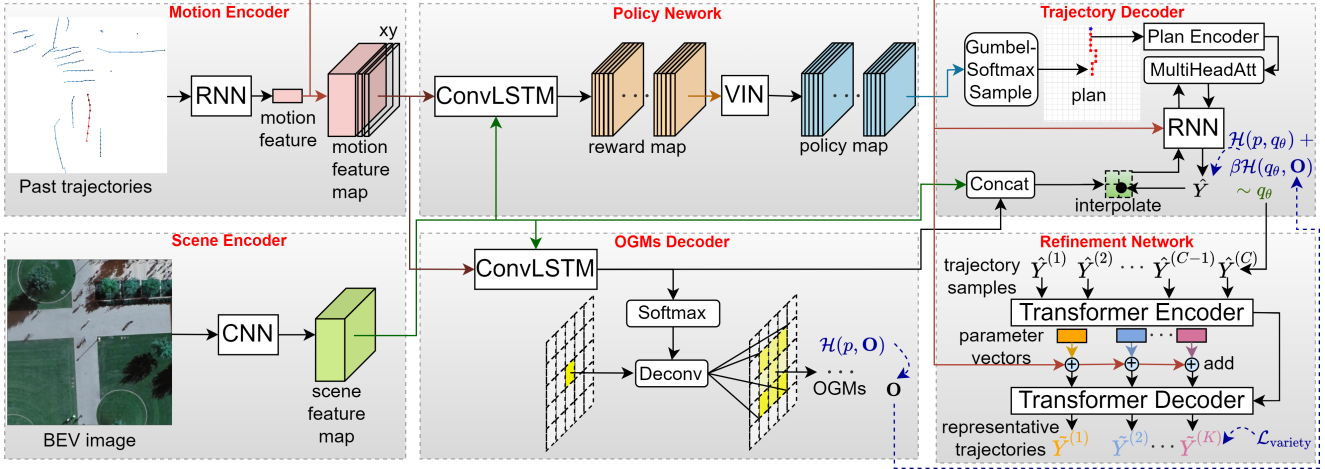
Figure 2. Overview of our approach.

Algorithm 1 Approximate Value Iteration Network

Input: $\mathbf{r}^n(s, a)$
Output: $\boldsymbol{\pi}^n(a|s)$
1: $\mathbf{V}^N(s) = 0, \forall s \in \mathcal{S}$;
2: **for** $n = N, \ldots, 2, 1$ **do**
3:      $\mathbf{Q}^n(s, a) = \mathbf{r}^n(s, a) + \mathbf{V}^n_{s' = \mathcal{T}(s,a)}(s'), \ \forall s \in \mathcal{S}, \ \forall a \in \mathcal{A}$;
4:      $\mathbf{V}^{n-1}(s) = \text{logsumexp}_a \mathbf{Q}^n(s, a), \forall s \in \mathcal{S}$;
5:      $\boldsymbol{\pi}^n(a|s) = \text{softmax}_a \mathbf{Q}^n(s, a), \forall s \in \mathcal{S}$;
6: **end for**

which recursively computes the next value map by a convolution of the current value map with transition filters. To improve the value iteration network's performance, we utilize an approximate value iteration in the MaxEnt IRL formulation [50,58] with non-stationary rewards. Algo. 1 describes the overall computation process of this network.

4.3. OGMs Decoder

To provide an explicit approximation of the ground-truth trajectory distribution, we predict a sequence of dynamic OGMs based on the observation features using a ConvLSTM network. With the scene feature map as input, the hidden map of the ConvLSTM network at time t is:

$$\mathbf{H}_t = \text{ConvLSTM}_o\left(\mathbf{H}_{t-1}, \mathbf{F}\right),$$

The hidden map is initialized with the embedding of the motion feature map $\mathbf{H}_0 = \Phi(\mathbf{M})$.

Then, instead of directly outputting an OGM from each hidden map, we derive a pixel-adaptive normalized deconvolution filter whose weights are spatially varying, nonnegative and sum to one. The deconvolution is subsequently applied to the last OGM to obtain the next one:

$$\mathbf{O}_t = \text{Deconv}\left(\mathbf{O}_{t-1}, \text{softmax}(\Phi(\mathbf{H}_t))\right),$$

where the initial OGM $\mathbf{O}_0$ is a probability matrix to be learned. Our deconvolution method can directly model the

probability density transition process. Besides, the limited size of the normalized deconvolution kernel ensures that the probability mass diffuses into nearby grid cells in a conservative manner, reflecting the prior knowledge that agents do not suddenly disappear or jump between distant locations.

4.4. Trajectory Decoder

Conditioned on a plan from the policy roll-out or the data, an RNN decoder is applied to generate the future position distribution recursively based on local features.

Plan sampling: We generate a plan $\hat{S} = \{\hat{s}^n \in \mathbb{R}^2 \mid n = 1, \ldots, N\}$ by sampling the non-stationary policy outputted by the policy network. However, directly sampling the policy with discrete state and action spaces will introduce difficulty in loss back-propagation. To overcome this difficulty, we sample the policy with the Gumbel-Softmax trick [15], resulting in continuous action and state. Besides, we obtain the policy at continuous state $\hat{s}^n$ by bilinear interpolation.

Plan encoder: Given a ground-truth plan S (or a sampled plan $\hat{S}$), we first collect the local scene feature from scene feature map $\mathbf{F}$ and non-stationary feature from the corresponding hidden map of ConvLSTM_r at each plan state. Then we concatenate these features with the state's coordinates as input to an RNN, whose hidden state at step n is:

$$h^n = \text{RNN}_s\left(h^{n-1}, \phi\left[s^n, \mathbf{F}(s^n), \mathbf{H}^n(s^n)\right]\right).$$

Since the sampled plan's state $\hat{s}^n$ is on the continuous plane, the local features like $\mathbf{F}(s^n)$ are gathered by bilinear interpolation at the spatial dimensions of the feature map $\mathbf{F}$ corresponding to the physical position s^n. Fig. 3 illustrates how the plan encoder extracts the plan features $h^{1:N} = \{h^n \mid n = 1, \ldots, N\}$.

Multi-head attention based decoder: Since different dimensions of the plan features at different steps may have different impacts on current hidden state [30], we utilize a multi-head scaled dot product attention module [47] to ag-
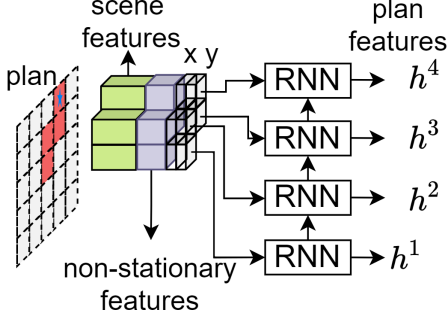
2246

Figure 3. The local scene and non-stationary features at each plan state are concatenated with its location coordinates and then fed into an RNN to obtain all plan features.

gregate the plan information:

$$\text{MultiHead}(Q, K, V) = [\text{Att}(QW_i^Q, KW_i^K, VW_i^V)_{i=1}^H]W^O,$$

$$\text{where } \text{Att}(Q_i, K_i, V_i) = \text{softmax}(\frac{Q_i K_i^T}{\sqrt{d_k}})V_i,$$

where d_k is the dimension of each head. At each future time t, we linearly project the trajectory decoder's previous hidden state h_{t-1} into the query Q_i and the plan features into the key K_i and value V_i through linear layers W_i^Q, W_i^K and W_i^V. The attention module output a_t is then concatenated with coordinates and local bilinearly interpolated features on scene feature map and corresponding OGM hidden map at the previous position Y_{t-1} as input to an RNN decoder:

$$a_t = \text{MultiHead}(h_{t-1}, h^{1:N}, h^{1:N}),$$
$$h_t = \text{RNN}_t\left(h_{t-1}, \phi\left[a_t, Y_{t-1}, \mathbf{F}(Y_{t-1}), \mathbf{H}_t(Y_{t-1})\right]\right),$$

where the initial hidden state h_0 is the embedded motion feature $\phi(m_0)$. Then, the hidden state h_t is utilized to predict the position $\hat{Y}_t$ distribution which is assumed to be a bivariate Gaussian distribution parameterized with the mean $\mu_t + Y_{t-1}$, standard deviation σ_t, and correlation ρ_t:

$$[\mu_t, \sigma_t, \rho_t] = h_t W^P, \quad \hat{Y}_t \sim \mathcal{N}(\mu_t + Y_{t-1}, \sigma_t, \rho_t).$$

During generating predictions $\hat{Y}$, the above ground-truth position Y_{t-1} is substituted by the position $\hat{Y}_{t-1}$ sampled from the predicted distribution with the reparameterization trick [17] to ensure differentiability.

4.5. Refinement Network

We design a refinement network to present a succinct representation of a trajectory distribution with several representative trajectories. The network is an encoder-decoder framework based on Transformer [47] but without positional embedding and auto-regressive decoding because the multi-head attention module in Transformer can well capture the relation between unordered samples to ensure diversity. We first over-sample a large number of trajectory samples $\{\hat{Y}^{(1)}, \hat{Y}^{(2)}, \dots, \hat{Y}^{(C)}\}$ to cover the trajectory distribution, e.g. $C = 200$. Then, all trajectory samples are flattened into vectors and embedded as input to the

Transformer encoder without the positional embedding. To save inference time, we utilize a generative style decoder like [57] but the inputs to our decoder are the summations of the embedded motion features and K different parameter vectors instead of fixed tokens. Finally, we embed the decoder output to obtain a few representative trajectories $\{\tilde{Y}^{(1)}, \tilde{Y}^{(2)}, \dots, \tilde{Y}^{(K)}\}$, e.g. $K = 20$.

4.6. Training Process

To achieve different goals including a good OGM, distribution and representative sets at different steps, our training process has the following four steps:

1. **OGMs learning:** The observation encoder and OGMs decoder are trained to predict OGMs by minimizing the NLL loss:

$$\mathcal{H}(p, \mathbf{O}) = -\mathop{\mathbb{E}}_{\Omega \sim \Psi, Y \sim p(\cdot|\Omega), \mathbf{O} = o_\alpha(\Omega)} \log \prod_{t=1}^{t_f} \mathbf{O}_t(Y_t).$$

2. **Trajectory distribution learning:** Based on the learned observation encoder and OGMs decoder, we train the policy network and trajectory decoder to induce a trajectory distribution that minimizes the approximated symmetric cross-entropy loss:

$$\mathcal{L}_{\text{sce}} = \mathcal{H}(p, q_\theta) + \beta \mathcal{H}(q_\theta, \mathbf{O}).$$

3. **Representative trajectories learning:** Using the trajectories sampled from the learned distribution, we train the refinement network to generate representative trajectories with the variety (MoN) loss [9]:

$$\mathcal{L}_{\text{variety}} = \min_{k \in \{1, \dots, K\}} \|Y - \tilde{Y}^{(k)}\|_2.$$

4. **End-to-end fine-tuning:** We fine-tune the whole network in an end-to-end manner with the variety loss.

Only the first two steps are required for learning a trajectory distribution while all four steps are for obtaining a compact set of representative trajectories.

5. Experimental Results

5.1. Implementation Details

We augment all trajectories and scene images in the training data by $90°$ rotations and flipping. All data fed into and generated from our model are in the world coordinates rather than the pixel coordinates as in previous works [6, 28, 42]. The BEV image fed to the scene encoder is a 200×200 crop of the RGB image around the agent's location. The scene encoder CNN_f consists of the first two layers of ResNet34 [10] and a convolutional layer with kernel size 2 and stride 2, which outputs scene feature map of 32 channels and size 25×25 as the MDP grid. The RNNs

are implemented by gated recurrent units (GRU) with hidden size 64. ConvLSTM_r and ConvLSTM_d have 1 and 2 layers each, with kernel size 3 and 32 hidden states respectively, and the deconvolution kernel size is 5. The multi-head attention module in the trajectory decoder has 4 heads of 16 dimensions. The Transformer encoder and decoder consist of 3 layers with hidden size 64 and 8 self-attention heads and a dropout rate of 0.1. We train using Adam optimizer with a learning rate of 0.001 in the first three steps and 0.0001 in the last step. We have released our code at https://github.com/Kguo-cs/TDOR.

5.2. Datasets and Metrics

Datasets. We evaluate our method on two datasets. Most of our tests are conducted on the **Stanford Drone Dataset** (SDD) [40] provides top-down RGB videos captured on the Stanford University campus by drones at 60 different scenes, containing annotated trajectories of more than 20,000 targets such as pedestrians, bicyclists and cars. Early works [5, 23, 43] consider all trajectories in SDD and subsequent works [27–29, 56] focus on pedestrian trajectories using the *TrajNet* benchmark [42]. On these two splits, we report the results of predicting the 12-step future with the 8-step history with 0.4 seconds step interval. Besides, we report our long-term prediction results on the **Intersection Drone Dataset** (inD) [3], which consists of longer drone recorded trajectories of road users than SDD collected at four German intersections. To evaluate our method's long-term forecasting performance, we use data in [27], including 1222 training and 174 test trajectories with 5-second history and 30-second future with the 1Hz sampling rate.

Metrics. We evaluate our performance of representative samples with three metrics. The first two are commonly used sample-based diversity metrics [9]: minADE_K, *i.e.*, the *minimum average*, and minFDE_K, *i.e.*, the *final displacement errors* between K predictions and ground-truth trajectory in pixels. Following P2T [6], we also report results on the quality metric, *Offroad Rate*, which measures the fraction of the predicted positions falling outside road while ground-truth positions inside road.

5.3. Performance Evaluation

We benchmark against the following state-of-the-arts. **Social GAN** [9] proposes a GAN-based method to predict diverse and socially acceptable trajectories. **Desire** [19] uses CVAE to generate trajectory samples, which are then recursively ranked and refined. **Multiverse** [23] selects multiple coarse trajectories from predicted OGMs by beam search and then refines them with continuous displacement vectors. **SimAug** [22] improves Multiverse [23]'s robustness by utilizing simulated multi-view data. **P2T** [6] predicts the future trajectory conditioned on a plan generated by the deep MaxEnt IRL. **PECNet** [28] is a goal-

Model	minADE_{20}	minFDE_{20}	Offroad Rate
S-GAN [9]	27.25	41.44	-
Desire [19]	19.25	34.05	-
Multiverse [23]	14.78	27.09	-
SimAug [22]	10.27	19.71	-
P2T [6]	10.97	18.40	0.065
Ours	**8.60**	**13.90**	**0.050**
PECNet [28]	9.96	15.88	0.071
LB-EBM [33]	8.87	15.61	0.070
P2T [6]	8.76	14.08	-
Y-Net [27]	7.85	11.85	**0.048**
V [49]	7.34	11.53	-
Ours	**6.77**	**10.46**	0.066

Table 1. Comparison with state-of-the-art methods on the entire SDD dataset (above) and its *TrajNet* split (below) in predicting short-term 4.8-second future.

Model	minADE_{20}	minFDE_{20}
S-GAN [9]	38.57	84.61
PECNet [28]	20.25	32.95
Y-net [27]	14.99	21.13
Ours	**13.09**	**19.39**

Table 2. Results on the inD in the long-term 30-second prediction.

conditioned model dividing the task into goal estimation and trajectory prediction. **LB-EBM** [33] infers intermediate waypoints using latent vector sampled from a cost-based history. **Y-Net** [27] models the future position's multimodality with heatmaps and samples a trajectory from the heatmap conditioned on sampled goal and waypoints. **V** [49] is a concurrent method proposing a two-stage Transformer network to model the trajectory and its Fourier spectrum in the keypoints and interactions levels, respectively.

The performance of our model in the short-term trajectory prediction compared with state-of-the-art methods on the SDD is reported in Tab. 1. minADE_{20} and minFDE_{20} values follow the original papers, while the offroad rates are computed using the released codes and models of different approaches. On both data splits, our model achieves the best performance according to the metrics of minADE_{20} and minADE_{20} and offroad rate. Notably, our results are achieved without manually labeled semantic maps in Y-Net [27] or simulation data in SimAug [22].

We also report our long-term prediction results on the inD in Tab. 2. Our results are again achieved without the manually annotated semantic maps in Y-net [27].

A set of qualitative examples are presented in the supplementary material, which demonstrates that our models are able to learn a diverse and feasible distribution and predict diverse representative trajectories.

5.4. Ablation Study

Ablation experiments on *TrajNet* split are used to expose the significance of different components of our model:

OGMs Prediction Model	$\mathcal{H}(p, \mathbf{O})$
CNN [38]	17.52
ConvLSTM [39]	10.52
ConvLSTM+DiscreteResidualFlow [14]	10.64
ConvLSTM+GraphAttentionNetwork [34]	10.40
ConvLSTM+Deconvolution (Ours)	**10.31**

Table 3. Comparison of four baselines and our method with close parameter numbers in predicting OGMs.

OGMs decoder: First, we consider the ground-truth distribution approximation using one OGM obtained by a CNN acting on scene and motion features as R2P2 [38]. Then, we study how effective our deconvolution parameterization in the ConvLSTM is for OGMs prediction. We implement three baseline OGMs prediction networks inspired by [14, 34, 39]: ConvLSTM directly outputs OGMs from hidden maps; ConvLSTM+DiscreteResidualFlow outputs the log-probability residual between OGMs from hidden maps; ConvLSTM+GraphAttentionNetwork processes hidden maps with a graph attention network at each step. We train these models as our first training step. Results in Tab. 3 about the OGM decoding loss measured by the NLL loss show that our approximation with different OGMs using deconvolution parameterization is the most effective.

Hyperparameter β: To investigate how the hyperparameter β in the symmetric cross-entropy loss affects learning trajectory distribution, we train the policy network and the trajectory decoder with various β values. To measure the learned distribution diversity, we leverage the RF_K metric from [34], *i.e.* the ratio of the average FDE to the minimum FDE among K predictions ($\mathrm{avgFDE}_K / \mathrm{minFDE}_K$). A large avgFDE_K implies that predictions spread out while a small minFDE_K ensures predictions not being arbitrarily stochastic. The off-road rate metric is also applied to evaluate the distribution's precision. As shown in Tab. 4, with increasing β values, the off-road rate and reverse cross-entropy decrease, implying a more precise distribution model, while the forward cross-entropy increases and RF_{20} decreases, meaning the distribution is getting less diverse. It shows that the hyperparameter β can balance the predicted distribution's diversity and accuracy while a distribution only minimizing the forward cross-entropy can cover the data well but will produce implausible samples.

Reward layers: First, we study how the IRL learning method is beneficial compared to behavior cloning (BC). In the BC method, we ablate the value iteration network and directly output the non-stationary policy in replace of the non-stationary reward. Then, our non-stationary reward is compared with the stationary reward (SR) used in previous works [6, 35]. The SR method is implemented by mapping the concatenation of the motion and scene feature maps into one reward map through two fully connected convolutional layers. Results in Tab. 4 show that a non-stationary reward

Model	β	$\mathcal{H}(p, q_\theta)$	$\mathcal{H}(q_\theta, \mathbf{O})$	RF_{20}	Offroad Rate
Ours	0	**-32.19**	24.13	**7.52**	0.035
Ours	0.1	-31.95	8.56	4.99	0.034
Ours	0.2	-31.75	7.536	4.23	0.030
Ours	1	-31.52	4.21	2.70	0.022
Ours	10	-29.76	**3.52**	2.13	**0.020**
BC	0.2	-31.62	8.10	4.80	0.034
SR	0.2	-31.72	9.52	6.95	0.033

Table 4. Effect of β and reward in trajectory distribution forecasting based on the predicted OGMs.

Method	minADE_{20}	minFDE_{20}	Offroad Rate
w/o refinement	8.78	14.34	**0.045**
K-means	7.64	12.12	0.058
w/o end-to-end	7.36	11.51	0.077
multi-task	6.94	10.68	0.066
variety loss	8.16	13.04	0.084
Ours	**6.77**	**10.46**	0.066

Table 5. Effect of the refinement and training methods in predicting representative trajectories based on the learned trajectory distribution with $\beta = 0.2$.

outperforms no reward or a stationary one in terms of forward and reverse cross-entropy and off-road rate.

Refinement network: We consider two models without the refinement network. One removes the refinement network from our method. The other replaces our refinement network with K-means in [6] and outputs K cluster centers of trajectory samples as representatives. Both models are trained end-to-end using the variety loss based on the pretrained distribution model. The comparison between Tab. 5 and Tab. 1 bottom part shows that the refinement network is indispensable and more effective than K-means while increasing the offroad rate due to the diversity loss.

Training process: Firstly, we show the result of only completing the first three training processes without the end-to-end fine-tuning process. Besides, we also consider two other training processes. One is to train the network with the sum of all losses like multi-task. The other one is with the variety loss only. Tab. 5 demonstrates that our training process with end-to-end fine-tuning can improve performance in predicting representative trajectories. We find that training only with variety loss is unstable and may not converge.

6. Conclusion

We have proposed an end-to-end interpretable trajectory distribution prediction model based on a grid plan. Our model can learn to produce a diverse and admissible trajectory distribution by minimizing the symmetric cross-entropy loss. We also design a flexible refinement network to generate a small set of representative trajectories. Finally, we demonstrate the effectiveness of our approach in two real-world datasets with state-of-the-art performance.

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

[2] Javad Amirian, Jean-Bernard Hayet, and Julien Pettré. Social Ways: Learning multi-modal distributions of pedestrian trajectories with gans. In *CVPRW*, pages 0–0, 2019. 1, 2

[3] Julian Bock, Robert Krajewski, Tobias Moers, Steffen Runde, Lennart Vater, and Lutz Eckstein. The inD Dataset: A drone dataset of naturalistic road user trajectories at german intersections. 2019. 2, 7

[4] Sergio Casas, Abbas Sadat, and Raquel Urtasun. MP3: A unified model to map, perceive, predict and plan. In *CVPR*, pages 14403–14412, 2021. 2, 3

[5] Patrick Dendorfer, Aljosa Osep, and Laura Leal-Taixé. Goal-GAN: Multimodal trajectory prediction based on goal position estimation. In *ACCV*, 2020. 2, 3, 7

[6] Nachiket Deo and Mohan M Trivedi. Trajectory forecasts in unknown environments conditioned on grid-based plans. *arXiv preprint arXiv:2001.00735*, 2020. 1, 2, 3, 6, 7, 8

[7] Nachiket Deo, Eric M Wolff, and Oscar Beijbom. Multimodal trajectory prediction conditioned on lane-graph traversals. *arXiv preprint arXiv:2106.15004*, 2021. 2, 3

[8] Stuart Eiffert, Kunming Li, Mao Shan, Stewart Worrall, Salah Sukkarieh, and Eduardo Nebot. Probabilistic crowd gan: Multimodal pedestrian trajectory prediction using a graph vehicle-pedestrian attention network. *IEEE Robotics and Automation Letters*, 5(4):5026–5033, 2020. 1, 2

[9] 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. 1, 2, 3, 6, 7

[10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778, 2016. 6

[11] Yue Hu, Siheng Chen, Ya Zhang, and Xiao Gu. Collaborative motion prediction via neural motion message passing. In *CVPR*, pages 6319–6328, 2020. 1, 2

[12] Xin Huang, Stephen G McGill, Jonathan A DeCastro, Luke Fletcher, John J Leonard, Brian C Williams, and Guy Rosman. DiversityGAN: Diversity-aware vehicle motion prediction via latent semantic sampling. *IEEE Robotics and Automation Letters*, 5(4):5089–5096, 2020. 3

[13] 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. 1, 2, 3

[14] Ajay Jain, Sergio Casas, Renjie Liao, Yuwen Xiong, Song Feng, Sean Segal, and Raquel Urtasun. Discrete residual flow for probabilistic pedestrian behavior prediction. In *Conference on Robot Learning*, pages 407–419, 2020. 3, 8

[15] Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. In *ICLR*, 2017. 2, 5

[16] ByeoungDo Kim, Chang Mook Kang, Jaekyum Kim, Seung Hi Lee, Chung Choo Chung, and Jun Won Choi. Probabilistic vehicle trajectory prediction over occupancy grid map via recurrent neural network. In *International Conference on Intelligent Transportation Systems*, pages 399–404, 2017. 3

[17] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. In *ICLR*, 2014. 6

[18] Parth Kothari, Sven Kreiss, and Alexandre Alahi. Human trajectory forecasting in crowds: A deep learning perspective. *IEEE Transactions on Intelligent Transportation Systems*, 2021. 4

[19] Namhoon Lee, Wongun Choi, Paul Vernaza, Christopher B Choy, Philip HS Torr, and Manmohan Chandraker. Desire: Distant future prediction in dynamic scenes with interacting agents. In *CVPR*, pages 336–345, 2017. 1, 2, 3, 7

[20] Karen Leung, Edward Schmerling, and Marco Pavone. Distributional prediction of human driving behaviours using mixture density networks. *Technical report, Stanford University*, 2016. 1

[21] Yuke Li. Pedestrian path forecasting in crowd: A deep spatio-temporal perspective. In *ACMMM*, pages 235–243, 2017. 3

[22] Junwei Liang, Lu Jiang, and Alexander Hauptmann. SimAug: Learning robust representations from simulation for trajectory prediction. In *ECCV*, pages 275–292, 2020. 7

[23] Junwei Liang, Lu Jiang, Kevin Murphy, Ting Yu, and Alexander Hauptmann. The garden of forking paths: Towards multi-future trajectory prediction. In *CVPR*, pages 10508–10518, 2020. 1, 2, 3, 7

[24] Katie Luo, Sergio Casas, Renjie Liao, Xinchen Yan, Yuwen Xiong, Wenyuan Zeng, and Raquel Urtasun. Safety-oriented pedestrian occupancy forecasting. In *International Conference on Intelligent Robots and Systems*, pages 1015–1022, 2021. 3

[25] Wenjie Luo, Bin Yang, and Raquel Urtasun. Fast and furious: Real time end-to-end 3d detection, tracking and motion forecasting with a single convolutional net. In *CVPR*, pages 3569–3577, 2018. 2

[26] Osama Makansi, Eddy Ilg, Ozgun Cicek, and Thomas Brox. Overcoming limitations of mixture density networks: A sampling and fitting framework for multimodal future prediction. In *CVPR*, pages 7144–7153, 2019. 1

[27] Karttikeya Mangalam, Yang An, Harshayu Girase, and Jitendra Malik. From goals, waypoints & paths to long term human trajectory forecasting. In *ICCV*, 2021. 3, 7

[28] Karttikeya Mangalam, Harshayu Girase, Shreyas Agarwal, Kuan-Hui Lee, Ehsan Adeli, Jitendra Malik, and Adrien Gaidon. It is not the journey but the destination: Endpoint conditioned trajectory prediction. In *ECCV*, pages 759–776, 2020. 1, 3, 6, 7

[29] Francesco Marchetti, Federico Becattini, Lorenzo Seidenari, and Alberto Del Bimbo. Mantra: Memory augmented networks for multiple trajectory prediction. In *CVPR*, pages 7143–7152, 2020. 3, 7

[30] Jean Mercat, Thomas Gilles, Nicole El Zoghby, Guillaume Sandou, Dominique Beauvois, and Guillermo Pita Gil. Multi-head attention for multi-modal joint vehicle motion forecasting. In *International Conference on Robotics and Automation*, pages 9638–9644, 2020. 5

[31] Hiroaki Minoura, Ryo Yonetani, Mai Nishimura, and Yoshi-taka Ushiku. Crowd density forecasting by modeling patch-based dynamics. *IEEE Robotics and Automation Letters*, 6(2):287–294, 2020. 3

[32] Nima Mohajerin and Mohsen Rohani. Multi-step predic-tion of occupancy grid maps with recurrent neural networks. *CVPR*, pages 10592–10600, 2019. 3

[33] Bo Pang, Tianyang Zhao, Xu Xie, and Ying Nian Wu. Tra-jectory prediction with latent belief energy-based model. In *CVPR*, pages 11814–11824, 2021. 3, 7

[34] Seong Hyeon Park, Gyubok Lee, Jimin Seo, Manoj Bhat, Minseok Kang, Jonathan Francis, Ashwin Jadhav, Paul Pu Liang, and Louis-Philippe Morency. Diverse and admissi-ble trajectory forecasting through multimodal context under-standing. In *ECCV*, pages 282–298, 2020. 1, 2, 3, 4, 8

[35] Max Pflueger, Ali Agha, and Gaurav S Sukhatme. Rover-IRL: Inverse reinforcement learning with soft value iteration networks for planetary rover path planning. *IEEE Robotics and Automation Letters*, 4(2):1387–1394, 2019. 4, 8

[36] Tung Phan-Minh, Elena Corina Grigore, Freddy A Boulton, Oscar Beijbom, and Eric M Wolff. CoverNet: Multimodal behavior prediction using trajectory sets. In *CVPR*, pages 14074–14083, 2020. 2, 3

[37] Eike Rehder, Florian Wirth, Martin Lauer, and Christoph Stiller. Pedestrian prediction by planning using deep neu-ral networks. In *International Conference on Robotics and Automation*, pages 5903–5908, 2018. 4

[38] Nicholas Rhinehart, Kris M Kitani, and Paul Vernaza. R2p2: A reparameterized pushforward policy for diverse, precise generative path forecasting. In *ECCV*, pages 772–788, 2018. 1, 2, 3, 4, 8

[39] Daniela Ridel, Nachiket Deo, Denis Wolf, and Mohan Trivedi. Scene compliant trajectory forecast with agent-centric spatio-temporal grids. *IEEE Robotics and Automa-tion Letters*, 5(2):2816–2823, 2020. 3, 8

[40] Alexandre Robicquet, Amir Sadeghian, Alexandre Alahi, and Silvio Savarese. Learning social etiquette: Human tra-jectory understanding in crowded scenes. In *ECCV*, pages 549–565, 2016. 2, 7

[41] 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. 2

[42] Amir Sadeghian, Vineet Kosaraju, Agrim Gupta, Silvio Savarese, and A Alahi. Trajnet: Towards a benchmark for human trajectory prediction. *arXiv preprint*, 2018. 6, 7

[43] 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. 1, 2, 7

[44] Tim Salzmann, Boris Ivanovic, Punarjay Chakravarty, and Marco Pavone. Trajectron++: Dynamically-feasible trajec-tory forecasting with heterogeneous data. In *ECCV*, pages 683–700, 2020. 1, 2

[45] Aviv Tamar, Yi Wu, Garrett Thomas, Sergey Levine, and Pieter Abbeel. Value iteration networks. In *NeurIPS*, pages 2154–2162, 2016. 2, 4

[46] Luca Anthony Thiede and Pratik Prabhanjan Brahma. Ana-lyzing the variety loss in the context of probabilistic trajec-tory prediction. In *ICCV*, pages 9954–9963, 2019. 1

[47] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszko-reit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NeurIPS*, pages 5998–6008, 2017. 5, 6

[48] Yisen Wang, Xingjun Ma, Zaiyi Chen, Yuan Luo, Jinfeng Yi, and James Bailey. Symmetric cross entropy for robust learning with noisy labels. In *ICCV*, pages 322–330, 2019. 2

[49] Conghao Wong, Beihao Xia, Ziming Hong, Qinmu Peng, and Xinge You. View Vertically: A hierarchical network for trajectory prediction via fourier spectrums. *arXiv preprint arXiv:2110.07288*, 2021. 7

[50] Markus Wulfmeier, Dushyant Rao, Dominic Zeng Wang, Pe-ter Ondruska, and Ingmar Posner. Large-scale cost function learning for path planning using deep inverse reinforcement learning. *The International Journal of Robotics Research*, 36(10):1073–1087, 2017. 2, 3, 5

[51] Shi Xingjian, Zhourong Chen, Hao Wang, Dit-Yan Yeung, Wai-Kin Wong, and Wang-chun Woo. Convolutional LSTM network: A machine learning approach for precipitation nowcasting. In *NeurIPS*, pages 802–810, 2015. 2, 3

[52] Ye Yuan and Kris Kitani. Diverse trajectory forecasting with determinantal point processes. In *ICLR*, 2020. 3

[53] Wenyuan Zeng, Wenjie Luo, Simon Suo, Abbas Sadat, Bin Yang, Sergio Casas, and Raquel Urtasun. End-to-end inter-pretable neural motion planner. In *CVPR*, pages 8660–8669, 2019. 2

[54] Pu Zhang, Wanli Ouyang, Pengfei Zhang, Jianru Xue, and Nanning Zheng. SR-LSTM: State refinement for lstm to-wards pedestrian trajectory prediction. In *CVPR*, pages 12085–12094, 2019. 2

[55] Hang Zhao, Jiyang Gao, Tian Lan, Chen Sun, Benjamin Sapp, Balakrishnan Varadarajan, Yue Shen, Yi Shen, Yun-ing Chai, Cordelia Schmid, Congcong Li, and Dragomir Anguelov. TNT: Target-driven trajectory prediction. In *Con-ference on Robot Learning*, 2020. 3

[56] He Zhao and Richard P Wildes. Where are you heading? dynamic trajectory prediction with expert goal examples. In *ICCV*, pages 7629–7638, 2021. 7

[57] Haoyi Zhou, Shanghang Zhang, Jieqi Peng, Shuai Zhang, Jianxin Li, Hui Xiong, and Wancai Zhang. Informer: Beyond efficient transformer for long sequence time-series forecast-ing. In *AAAI*, pages 11106–11115, 2021. 6

[58] Brian D Ziebart, Andrew L Maas, J Andrew Bagnell, and Anind K Dey. Maximum entropy inverse reinforcement learning. In *AAAI*, volume 8, pages 1433–1438, 2008. 2, 5