Stimulus Verification is a Universal and Effective Sampler in Multi-modal Human Trajectory Prediction

Jianhua Sun*, Yuxuan Li*, Liang Chai, Cewu Lu§
Shanghai Jiao Tong University, China
{gothic,yuxuan_li,chailiang1234,lucewu}@sjtu.edu.cn

Abstract

To comprehensively cover the uncertainty of the future, the common practice of multi-modal human trajectory prediction is to first generate a set/distribution of candidate future trajectories and then sample required numbers of trajectories from them as final predictions. Even though a large number of previous researches develop various strong models to predict candidate trajectories, how to effectively sample the final ones has not received much attention yet. In this paper, we propose stimulus verification, serving as a universal and effective sampling process to improve the multi-modal prediction capability, where stimulus refers to the factor in the observation that may affect the future movements such as social interaction and scene context. Stimulus verification introduces a probabilistic model, denoted as stimulus verifier, to verify the coherence between a predicted future trajectory and its corresponding stimulus. By highlighting prediction samples with better stimulus-coherence, stimulus verification ensures sampled trajectories plausible from the stimulus' point of view and therefore aids in better multi-modal prediction performance. We implement stimulus verification on five representative prediction frameworks and conduct exhaustive experiments on three widely-used benchmarks. Superior results demonstrate the effectiveness of our approach.

1. Introduction

Human trajectory prediction [3, 8, 9, 17, 35, 38, 39, 43] is a vital task towards autonomous driving systems and social robots, bridging the perception and decision modules [4, 26, 27, 42]. Due to the fact that there is no single correct future, one of the most important aspects of trajectory prediction lies in multi-modal prediction, studying to cover multiple distinct future possibilities with finite predictions. Specifically, for an observation, a set/distribution of candidate trajectories is first generated by a base prediction model, after which final ones of required number are sampled from the candidates, seeing Fig. 1-a.

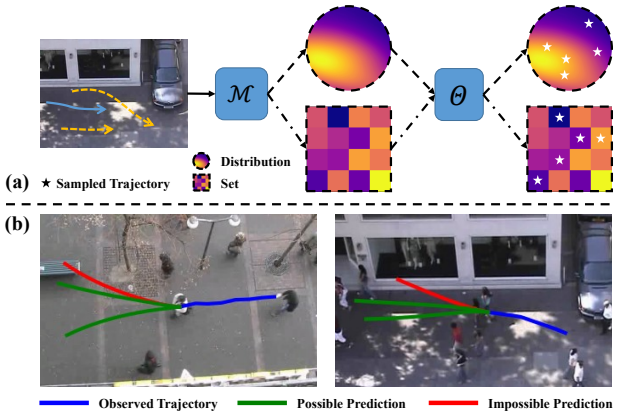

Figure 1. (a) Multi-modal trajectory prediction pipeline, including two major components: base prediction model $\mathcal{M}$ and sampler Θ. (b) Examples of stimulus(context)-inconsistent trajectories sampled from a stimulus(context)-aware base prediction model, Sophie [33]. Predictions in red leads to impossible regions.

To guarantee the performance of multi-modal trajectory prediction, countless efforts have been put into the design of prediction models in prior works using either generative or classification frameworks. For generative ones, some popular architectures such as GAN [7], VAE [14] and Normalizing Flow [29] have been applied and improved in plenty of studies on multi-modal trajectory prediction [8, 23, 33, 34, 40]. In the meantime, impressive results are also achieved with classification models such as MultiPath [3] and PCCSNet [39]. Yet it is often neglected that an effective sampling strategy, which selects the appropriate trajectories from the collection of candidates, also has a great impact on the prediction performance. Typically, random or Monte-Carlo sampling is adopted as the default one. Only a handful of works [2, 22, 46] have studied on

*contributed equally
§Cewu Lu is the corresponding author, the member of Qing Yuan Research Institute and MoE Key Lab of Artificial Intelligence, AI Institute, Shanghai Jiao Tong University, China and Shanghai Qi Zhi institute.

sampling strategy, while they all seek to solve this problem by searching better latent vectors for generative frameworks. Although they have achieved remarkable outcomes, the applications of such approaches are strongly limited as they are confined to generative frameworks only, ignoring the other mainstream frameworks, *i.e.* classification ones.

Besides, there are some factors in the observation that may influence the future movements. These factors, such as social interaction and scene context, are referred as *stimuli* [32] and their significance of guiding accurate predictions has been proven in numerous researches [15, 25, 33, 34]. Although base frameworks can learn from stimulus information to generate candidate predictions, if such information is missing in the sampling process, out-of-distribution predictions are still highly probable to be sampled and harm the overall accuracy, as illustrated in Fig. 1-b.

To raise a universal sampler taking stimulus into consideration, in this paper, we propose an explicit sampling process called stimulus verification. It first verifies the coherence between a candidate trajectory and its corresponding stimulus, and then samples highly stimulus-coherent ones as final results. Since the verification is an external process beyond the base prediction model, it can be universally applied to any multi-modal prediction frameworks. And meanwhile, stimulus information is successfully introduced, ensuring that the selected samples comply with the constrains of the involved stimuli.

Stimulus verification is built on a conditionally parameterized probabilistic model optimized by Maximum Likelihood Estimation, namely stimulus verifier. By feeding the candidate trajectory along with its stimulus into the verifier, corresponding output likelihood reveals the trajectory-stimulus coherence, where a higher likelihood indicates better coherence. To this end, we can map the likelihood to a coherence score, which further serves as the basis to sample more stimulus-coherent predictions from a large group of trajectory candidates.

Our method is simple to implement for any base prediction framework in a plug-and-play manner. We implement our approach on five representative frameworks [8, 29, 33, 39, 43] and conduct exhaustive experiments on three widely-used trajectory prediction benchmarks, *i.e.* ETH [28]/UCY [16] Dataset, Grand Central Station Dataset [47] and NBA SportVU Dataset [20], to validate our approach. Superior results confirm the effectiveness of stimulus verification.

2. Related Work

2.1. Stimulus in Trajectory Prediction

The trajectory prediction task [1, 8, 9, 11, 17, 30, 31, 48] is proposed to forecast a series of possible future movements of dynamic agents according to their past states. It has many important applications such as autonomous vehicles and service robots [9, 32]. Apart from the observed past trajectory, the future movement of an agent is generally made by taking into account the influences of multiple types of stimulus [32] such as scene context and social interaction. Researchers have developed many different ways to better model and learn stimulus information in trajectory prediction frameworks. For example, Social GAN [8], Social-STGCNN [25] and many other works [25, 34, 44] introduce pooling modules or GNNs to learn the social interactions among neighboring agents to generate socially plausible trajectories and avoid collisions. And as for context information, Sophie [33] and NEXT [19] design CNNs to process images of the scene so that the model can be aware of the physical restrictions when making predictions. Besides, some approaches use CNNs to learn human skeletons or appearances [19, 21], which turn out to be helpful as well. In this paper, we introduce commonly used social interaction and context information as examples of stimulus to demonstrate the effectiveness of stimulus verification.

2.2. Sampling in Trajectory Prediction

Despite the efforts put in designing stronger multi-modal trajectory prediction frameworks, not many researchers have paid their attention on better sampling appropriate trajectories from the candidates output by the prediction model. Existing ones mainly focus on increasing the diversity of final samples and solve this problem by searching better latent vector for generative frameworks. For example, DSF [46] maps forecasting context features to a set of latent codes which can be decoded into a set of diverse trajectory samples, and a diversity loss based on a determinantal point process (DPP) is introduced for optimization. LDS [22] leverages the likelihood under the pre-trained generative model and a robust diversity loss to learn a sampling distribution that induces diverse and plausible trajectory predictions. Inspired by Quasi-Monte Carlo, NPSN [2] achieves remarkable performance and remains light-weighted at the same time. A great difference between ours and prior works is that these approaches are only applicable for generative models such as GANs, VAEs or Normalizing Flows, whereas ours can be adopted to any prediction model that is capable of generating multi-modal results.

2.3. Conditional Parameterization.

When modeling a conditional probability relationship, instead of directly learning a conditional distribution [24, 37], conditional parameterization learns a set of parameters that are dynamically generated according to the input conditions. Such formulation has previously been utilized on vision-related researches [12, 41]. In this work, we borrow such technique in our stimulus verifier as it can increase the capacity of the conditional distribution and enable exact

density estimation for an explicit distribution.

3. Stimulus Verification

3.1. Overview

In a typical multi-modal trajectory prediction process as Fig. 1, a prediction framework $\mathcal{M}$ first receives an observation $\mathbb{O}$ regarding an agent over a time period of τ, and generates a set of candidate future trajectories

$$\begin{aligned}\hat{\mathbb{Y}} &= \mathcal{M}(\mathbb{O}) \\ &= \left\{\hat{Y}_i = \{\hat{\mathbf{y}}_1, \hat{\mathbf{y}}_2, \cdots, \hat{\mathbf{y}}_T\} | i = 1, 2, \cdots\right\}\end{aligned} \tag{1}$$

where $\hat{\mathbf{y}}_t = (x_t, y_t)$ denotes the 2-dimensional position of the agent at time t, and T is the prediction horizon. Afterwards, a sampling process Θ is applied to the candidates to get the final predictions $\mathbb{Y}^*$, written as

$$\mathbb{Y}^* = \Theta\left(\hat{\mathbb{Y}}\right) \tag{2}$$

Under this paradigm, stimulus verification performs the process described in Eq. 2. First, trajectories in $\hat{\mathbb{Y}}$ along with the stimuli information $S \in \mathbb{O}$ are fed to the *stimulus verifier* V, which evaluates the likelihood L of each trajectory given its corresponding stimulus

$$L = V(\hat{Y}, S) \tag{3}$$

where $\hat{Y} \in \hat{\mathbb{Y}}$ denotes a predicted trajectory candidate. These likelihoods are then mapped to coherence scores c,

$$c = f(L) \tag{4}$$

where $f(\cdot)$ is the mapping function. With the guidance of these coherence scores, candidates highly compatible with the stimuli can be selected as the final predictions.

In Sec. 3.2, we introduce the structure of the stimulus verifier using two types of stimulus as examples, and we also talk about the training of the stimulus verifier. Then in Sec 3.3, we discuss the detailed step taken to sample trajectories according to the coherence. In Sec. 3.4, we further present the details of our implementation of the stimulus verifier.

Discussion. Most previous sampling approaches [2, 22, 46] follow the same idea, inserting an additional module into the generative network to sample the latent vectors from the prior distribution. These approaches perform sampling in an implicit manner which is only applicable to generative models, and thus their applications are naturally narrowed. In a very different way, our stimulus verification is an explicit sampling process that directly performs sampling on the prediction candidates, output of a multi-modal predictor. Therefore, stimulus verification is model-agnostic, meaning that it is applicable to any base prediction model.

3.2. Stimulus Verifier

According to Eq. 3, stimulus verifier V evaluates the likelihood L of a T-step trajectory $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_T\}$ given its corresponding stimulus S, where $\mathbf{x}_t = (x_t, y_t)$ denotes the position. Therefore, it is implemented as a probabilistic model to learn a conditional probability distribution p

$$L = V(\mathbf{X}, S) = p(\mathbf{X}|S) \tag{5}$$

To increase the capacity of the conditional distribution, we design its architecture as a conditionally parameterized distribution [12,41], where the parameters θ of the distribution p are dynamically generated conditioned on the given stimulus information S by a stimulus feature extractor F and a parameter generator G. Specifically, the stimulus information S is first encoded by the feature extractor F into stimulus representations, and then the representations are fed to the distribution generator G to dynamically generate the parameters θ for the base trajectory distribution p. In this manner, the likelihood can be calculated as

$$L = V(\mathbf{X}, S) = p(\mathbf{X}; \theta) = p(\mathbf{X}; G(F(S))) \tag{6}$$

Under this formulation, the stimulus verifier V consists of three major components, a stimulus feature extractor F, a parameter generator G and a base distribution p. In our implementation, p is defined as the product of all step-wise positional distributions,

$$p(\mathbf{X}; \theta) = \prod_{t=1}^{T} p_t(\mathbf{x}_t; \theta_t) \tag{7}$$

where each step-wise distribution p_t, parameterized by $\theta_t \in \theta$, denotes the positional distribution at timestep t. We adopt 2-dim Gaussian Mixture Models (GMMs) as the step-wise distributions p_t over all timesteps considering the superiority of mixture models in approximating complex distributions. Besides, another key benefit of using GMMs is that they are in the analytic form which enables explicit and exact density estimation. The parameter generator G is implemented with an MLP for all types of stimulus. In contrast, the architecture of the feature extractor F differs for different types of stimulus S. In Sec. 3.2.1 and Sec. 3.2.2, we use two types of the most widely used stimuli, namely *scene context* and *social interaction*, as examples to introduce the design of feature extractors and how the stimulus verifier works as a whole in detail. Fig. 2 demonstrates that the distributions learned by our verifiers have high correspondence to human cognition.

3.2.1 Scene Context as Stimulus

The scene context stimulus S_c is often represented with a semantic map of the scene [19, 23, 33]. It highlights areas

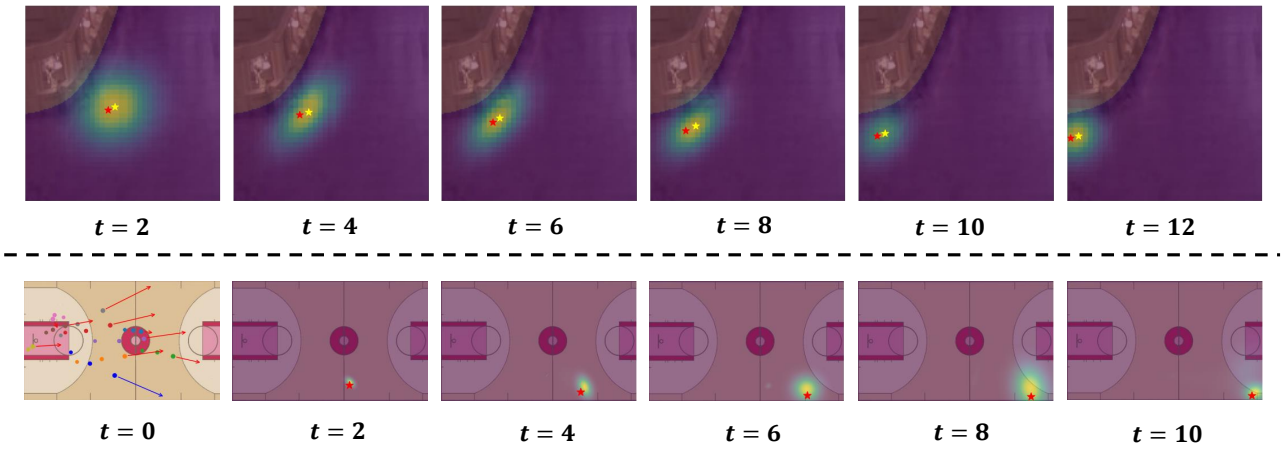

Figure 2. Illustration of distributions learned by the verifiers. The first row is an example of context stimulus and the second row is an example of social stimulus, where the prediction target is marked in blue and the other colors represent social agents. The ground truth position of the agent at time t is marked with a red star, and the yellow star in the first row indicates the center of the cropped context map m.

of different types such as obstacles and guiding lines in the scene. These features can give strong guidance to agents' movements such as areas to avoid and routes to follow. We follow this setting and use the semantic map as the context stimulus in our implementation.

For a certain trajectory $\mathbf{X}$, the context stimulus can intuitively be acquired by cropping a piece of semantic map from the whole scene covering the full journey of $\mathbf{X}$. However, this results in a potential problem. Since the total displacement of a trajectory varies a lot, the size of the cropped semantic map should either be fixed to a very large value in order to cover all the samples, or change dynamically according to the current sample. Both of these settings may cause the trajectory-context pairs to be scale-inconsistent. On one hand, a semantic map that is too large for a trajectory may contain considerable amounts of unrelated information and therefore jeopardize the learning. And on the other hand, semantic maps with dynamic scales could add unnecessary difficulties of fitting the model.

We tackle this issue by referring to the idea of sliding window algorithm. Specifically, the whole trajectory is first split into a series of position-displacement pairs, written as $\mathbf{X}_{sw} = \{\widetilde{\mathbf{x}}_t = (\mathbf{x}_t, \Delta\mathbf{x}_t) | t = 0, 1, 2, \cdots, T-1\}$, where $\Delta\mathbf{x}_t = \mathbf{x}_{t+1} - \mathbf{x}_t$ denotes the displacement over one timestep, and $\mathbf{x}_0$ is the position of the agent one step before $\mathbf{x}_1$. Then, for each pair of data $\widetilde{\mathbf{x}}_t$, we crop a fixed-size square area m_t centered at $\mathbf{x}_t$ from the scene as the semantic map for the trajectory at timestep t. This process can be interpreted as a square window sliding along the whole trajectory. In this manner, the context stimulus can be formulated as

$$S_c = \{m_t | t = 0, 1, 2, \cdots, T-1\} \tag{8}$$

Following previous works [23,33], we adopt a deep CNN as the context feature extractor F_c as it can effectively capture local patterns in the semantic map. For each timestep t, the context semantic map is first encoded by the extractor F_c, then the feature is fed to the generator G, which produces the parameters of the step-wise distribution at the next timestep $t+1$. Finally, the likelihood of the corresponding displacement can be calculated according to the parameterized distribution. The likelihood of the whole trajectory, which indicates its context coherence, can be acquired by multiplying all the displacements' likelihoods together. The whole process can be written as

$$V_c(\mathbf{X}_{sw}, S_c) = \prod_{t=0}^{T-1} p_t\left(\Delta\mathbf{x}_t; G\left(F_c\left(m_t\right)\right)\right) \tag{9}$$

3.2.2 Social Interaction as Stimulus

Similar as prior work [1, 8], we treat the social stimulus information as the states of all neighboring agents along with the target in a τ-step interval before the first timestep of $\mathbf{X}$, *i.e.* $t = 1$. We do not include the neighbors' states within the horizon of $\mathbf{X}$ due to the fact that neighboring agents' states are unknown during the prediction horizon in common trajectory prediction settings.

For each neighboring agent, we represent its state at one timestep by a 4-dim vector $u = (x^{rel}, y^{rel}, v^x, v^y)$, where x^{rel}, y^{rel} are the agent's relative coordinates to the predicted target, v^x, v^y denote its velocity in x, y axis. For each timestep, the states of all n neighboring agents can be collected as $U_t = \{u_i^t | i = 1, 2, \cdots n\}$, where u_i^t is the state vector of the i-th neighboring agent at timestep t. And the

social stimulus can be formulated as

$$S_s = \{U_t | t = -\tau + 1, \cdots, 0\} \in \mathbb{R}^{\tau \times n \times 4} \tag{10}$$

To extract features from social stimulus S_s, we first encode the information at each timestep and then aggregate the features in all τ steps together. We formulate the social states at each timestep as a fully-connected graph $\mathcal{G}$, whose vertices are the state vectors of all social agents and edges indicate interactions between a pair of agents. Then we adopt a Graph Convolutional Network (GCN) [5] to encode $\mathcal{G}$ considering its effectiveness in propagating the social effects of neighbors. After encoding the information at each timestep, we use an LSTM [10] to aggregate the features in the time dimension. In summary, the social representation can be extracted by

$$\begin{aligned} F_s(S_s) = LSTM(\{ & GCN(\mathcal{G}_t) \\ & | \, t = -\tau + 1, \cdots, 0\}) \end{aligned} \tag{11}$$

With the social representation, the generator G can generate the parameters for all the step-wise distributions together, and the likelihood of each step in the trajectory can be calculated using the corresponding parameters $\theta_t = G\left(F_s\left(S_s\right)\right)_t$. In this manner, the likelihood of the whole trajectory can be calculated by

$$V_s(\mathbf{X}_{norm}, S_s) = \prod_{t=1}^{T} p\left(\left(\mathbf{x}_t - \mathbf{x}_0\right); G\left(F_s\left(S_s\right)\right)_t\right) \tag{12}$$

where $\mathbf{X}_{norm} = \{(\mathbf{x}_t - \mathbf{x}_0) | t = 1, 2, \cdots, T\}$ is the normalized version of the original trajectory $\mathbf{X}$.

3.2.3 Training

As is previously mentioned, the goal of the stimulus verifier V is to learn a conditional probability distribution p of trajectories. To achieve this goal, we refer to the Maximum Likelihood Estimation (MLE) algorithm and train the verifier by maximizing the joint likelihood of all the available trajectories in the training set. For computational convenience, we equivalently maximize the logarithm of the joint likelihood and the loss function can be formulated by

$$\mathcal{L} = -\frac{1}{|\mathcal{X}|} \sum_{(\mathbf{X}, S) \in \mathcal{X}} \log V(\mathbf{X}, S) \tag{13}$$

Since both the parameter generator G and feature extractor F are differentiable with respect to the parameters of p, gradients can be back-propagated throughout the network.

3.3. Sampling with Stimulus Coherence

In this section, we introduce how to sample final predictions with stimulus coherence. According to Eq. 1, the base prediction model generates an excessive amount of N prediction candidates $\hat{\mathbb{Y}}$. The likelihood L of each candidate $\hat{Y}_i$ given the corresponding stimulus S can be calculated with the learned verifier and further used to reveal the coherence c_i between them by a mapping function $f(\cdot)$. In practice, we use logarithmic operations as the mapping function for computational convenience. And in the case of multiple types of stimulus being available, we consider the final coherence score c_i as the weighted sum over the coherence scores regarding each type of stimulus. After acquiring the coherence scores for all candidates, we then select the top k stimulus-coherent trajectories as the final results. In this manner, the overall stimulus coherence of the model's final output is significantly increased, which is identical to the goal of using stimulus for prediction, and therefore improves the prediction accuracy.

When more than one trajectory is required as the final predictions, there is one more step that is essential to the overall output quality. According to the definition of stimulus coherence, it is natural for similar trajectory-stimulus pairs to have similar coherence scores. Such property will lead to a tendency of selecting similar trajectories within a large group of samples, which could gravely impair the diversity of final prediction outputs.

We borrow the idea of Non-Maximum Suppression (NMS) [6] from the FDE perspective to tackle this issue. After sorting the N trajectories according to the coherence scores, we start a sequential selection process where a trajectory $\hat{Y}_i$ can only be selected as part of the final result if

$$\min\left(\{FDE(\hat{Y}_i, \hat{Y}') | \hat{Y}' \in \mathbb{Y}'\}\right) > \gamma \tag{14}$$

where $\mathbb{Y}'$ is the set of selected trajectories within the first $i-1$ samples, and γ is the threshold.

3.4. Implementation Details

In our implementation of the stimulus verifier, the GMM for each step-wise distribution has 12 components for both context and social stimulus. For context stimulus, we adopt a 4-layer convolutional neural network with average pooling and $ReLU$ activation as feature extractor, whose output size is 256. The cropped semantic maps are fixed as 50x50 pixel^2 for all datasets. For social stimulus, the GCN has three hidden layers, and uses maxpooling as the aggregation function. We collect the neighboring agents whose distance to the target at $t = 0$ is less than a threshold d, where d is 30 pixels for the Grand Central Station Dataset, 100 ft for the NBA Dataset and 2.5 meters for ETH/UCY Dataset. The number of timesteps used for social stimulus feature extraction is identical to the length of observation in each dataset. The output size of the social feature extractor is 128. The parameter generator is a 2-layer MLP with $ReLU$ activation for both types of stimulus. γ is picked by cross-validation.

All models are trained on one RTX 2080Ti GPU for 100 epochs. As for our experiments, we set the number of candidate trajectories N to 200 unless specified otherwise.

4. Experiments

4.1. Benchmarks, Metrics & Base Models

Benchmarks We conduct experiments on three widely-used benchmarks. *ETH [28]/UCY [16]*: It is one of the most commonly used benchmarks in the field of trajectory prediction and is composed of 5 sub-datasets. We follow previous work [1] on the leave-one-out evaluation. *Grand Central Station Dataset (GCS) [47]*: It contains a large amount of trajectories recorded from the Grand Central Station, which is a complex and crowded scene suitable for evaluation from both context and social perspective. As for train-test split settings, we took the first 80% (about 24 minutes) of the video as train set, and the rest 20% (about 6 minutes) as test set. *NBA Sports VU Dataset (NBA) [20]*: The NBA Sports VU Dataset contains the trajectory information of all ten players in real NBA games. The motion of agents in this scenario is strongly influenced by other players and the court. We randomly select 50k samples in total from the 2015-2016 season with a split of 65%, 10%, 25% as training, validation and testing data following [18]. For ETH/UCY and GCS, We take 8 steps (3.2 seconds) for observation and predict the upcoming 12 steps (4.8 seconds) following [8]. For NBA, we follow the traditional setting of taking 5 steps (2.0 seconds) for observation and predicting the upcoming 10 steps (4.0 seconds).

Average & Final Displacement Error (ADE & FDE) The ADE and FDE are two of the most commonly used evaluation metrics for trajectory prediction. ADE refers to the average L2 distance between the predicted trajectory and the ground truth over all timesteps, and FDE is the L2 distance at the last timestep. Under a best-of-k prediction setting, the performance is measured by the minimum ADE/FDE. Following the common setting [8], we adopt $k = 20$ in the experiments.

Base Models We introduce five representative base prediction models to comprehensively evaluate the effectiveness of stimulus verification, covering three typical generative models (*i.e.* GAN, VAE, Normalizing Flow) and the classification framework. *Autoregressive Flow* [13]: A generative model that is capable of learning a bijective transformation from the data space to a latent distribution. It is also adopted in LDS [22] as the base model. *Social GAN* [8]: A GAN-based framework that learns social interactions with a global pooling operation. *Sophie* [33]: A GAN-based framework that takes into account both the social interaction and scene context information by attention mechanisms. *PCCSNet* [39]: A prediction framework to solve the trajectory prediction in a classification and regression manner. *GroupNet* [43]: A recently proposed encoding framework that specializes at comprehensively modeling interactions using multi-scale hypergraphs. In our experiments, GroupNet is combined with CVAE to get the final predictions.

4.2. Main Results

In this section, we evaluate and discuss the effectiveness of proposed stimulus verification quantitatively. Results in Tab. 1 clearly shows that significant improvements can be achieved with stimulus verification for all prediction models on all benchmarks. By verifying from both context and social aspects, the performance can be further elevated to a higher level. Particularly, it is worth noting that even on prediction models that already have impressive results such as PCCSNet and GroupNet, our proposed stimulus verification can still offer at least 4.8% and up to 15.9% accuracy boost on ETH/UCY dataset. Moreover, although some of the base models have already introduced social and context information, stimulus verification still brings great improvements. This indicates that introducing stimulus information in sampling process is essential and helpful to ensure that selected samples comply with the constrains of involved stimuli.

Further, Fig. 3 shows the detailed improvements for different ks after verification. As the value of k decreases, the improvement grows larger, which shows that our approach exhibits higher superiority on more stringent criteria. For instance, with both stimulus, we are able to achieve a 17.8%/19.4% boost for top-1 performance on the GCS dataset with PCCSNet [39]. These results demonstrate the effectiveness of stimulus coherence on finding the more accurate predictions, since trajectories with higher coherence scores are sampled with higher priority according to our design, resulting in greater lifts with smaller ks.

4.3. Analysis

Visualization of Stimulus Coherence Visualizations in Fig. 4 illustrate that candidate predictions which fit better with the corresponding stimulus, as well as the ground truth, are assigned with high coherence scores. For example, in Fig. 4-b and c, trajectories following social rules that avoid collision with other agents are with better coherence. On the other hand, those impossible trajectories intersecting unwalkable regions, *e.g.* the darkest blue ones in Fig. 4-a, tend to have much lower coherence scores. Such phenomenon demonstrates that our approach successfully learns the correlation between a kind of stimulus and a certain trajectory, and is capable of distinguishing eligible trajectories in terms of the stimulus. By sampling candidates with high coherence scores, we can get accurate final predictions.

Dataset	Method	Original	+ Context	+ Social	+ Both	*Impr.*
ETH/UCY	Autoregressive Flow [13]	0.28 / 0.41	0.27 / 0.36	0.25 / 0.36	**0.24 / 0.34**	14.3% / 17.1%
	Social GAN [8]	0.61 / 1.21	0.51 / 0.96	0.50 / 0.95	**0.49 / 0.94**	19.7% / 22.3%
	Sophie [33]	0.39 / 0.65	0.35 / 0.54	0.33 / 0.53	**0.33 / 0.50**	15.4% / 23.1%
	PCCSNet [39]	0.21 / 0.42	0.20 / 0.40	0.20 / 0.39	**0.20 / 0.39**	4.8% / 7.1%
	GroupNet [43]	0.25 / 0.44	0.24 / 0.39	0.23 / 0.38	**0.23 / 0.37**	8.0% / 15.9%
GCS	Autoregressive Flow [13]	5.24 / 6.66	5.07 / 6.21	4.88 / 5.98	**4.73 / 5.54**	9.7% / 16.8%
	Social GAN [8]	6.06 / 8.98	5.81 / 8.04	5.34 / 7.11	**5.29 / 6.95**	12.7% / 22.6%
	Sophie [33]	5.53 / 9.48	5.22 / 8.72	5.15 / 8.51	**5.11 / 8.39**	7.6% / 11.5%
	PCCSNet [39]	4.17 / 6.02	4.10 / 5.74	4.03 / 5.43	**3.98 / 5.29**	4.6% / 12.1%
	GroupNet [43]	3.76 / 5.20	3.62 / 4.83	3.54 / 4.66	**3.50 / 4.58**	6.9% / 11.9%
NBA	Autoregressive Flow [13]	1.91 / 2.32	1.77 / 1.91	1.56 / 1.67	**1.57 / 1.64**	17.8% / 29.3%
	Social GAN [8]	1.78 / 2.48	1.61 / 1.97	1.60 / 1.95	**1.59 / 1.94**	10.7% / 21.8%
	Sophie [33]	1.55 / 2.14	1.45 / 1.84	1.44 / 1.84	**1.44 / 1.83**	7.1% / 14.5%
	PCCSNet [39]	1.47 / 1.70	1.42 / 1.49	1.42 / 1.44	**1.40 / 1.42**	4.8% / 16.5%
	GroupNet [43]	1.13 / 1.26	1.08 / 1.13	1.08 / 1.14	**1.08 / 1.12**	4.4% / 11.1%

Table 1. Performance (ADE/FDE) of base prediction models before and after stimulus verification on three widely-used benchmarks. For ETH/UCY, the values are reported by the average of all 5 sub-datasets. Since the official implementation of Sophie [33] is not publicly available, we report the results of our own implementation. The result of GroupNet [43] on NBA dataset is evaluated by the official pre-trained model at [36].

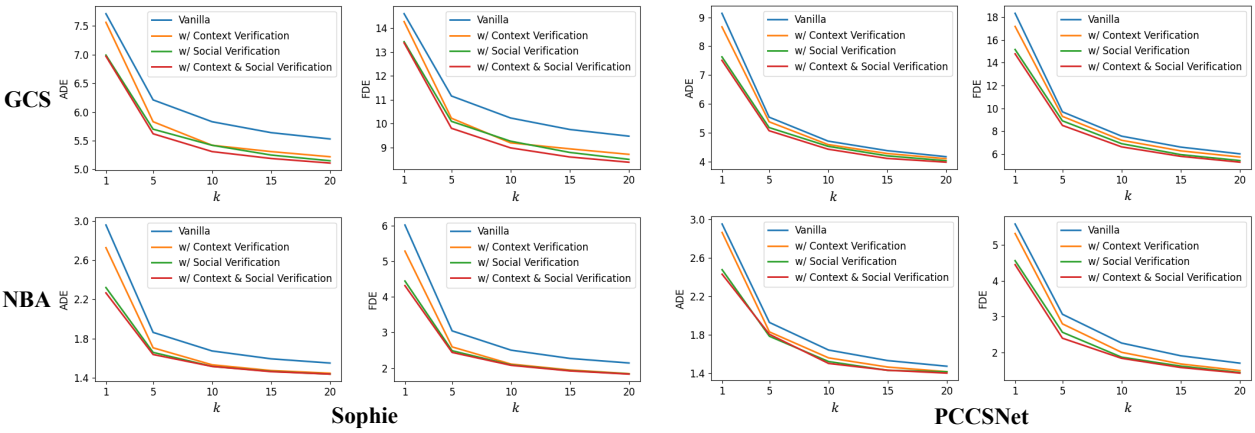

Figure 3. Improvements on ADE/FDE for different ks with stimulus verification. A smaller k refers to a more stringent criteria.

Contribution of Stimulus Coherence In Tab. 2, we demonstrate the contribution of stimulus coherence by comparing against vanilla NMS, which directly samples candidates according to Eq. 14 without first sorting them by the coherence scores. The results clearly show that the coherence scores will guide to select more accurate predictions with higher priority and thereby significant performance improvement can be achieved.

Number of Candidates Since the stimulus verification samples final trajectories from several candidates, we study how the number of candidates will influence the performance. The results are shown in Tab. 3. When N is in a small level, the performance of stimulus verification elevates fast as the number grows. After N grows large enough, the speed of performance improvement gradually reduces.

Comparisons with Prior Works We compare the performance of stimulus verification against three previous sampling approaches, namely DLow [45], LDS [22], and NPSN [2]. Following their settings of base model, we compare DLow and LDS on Autoregressive Flow, and NPSN on Social GAN. The results in Tab. 4 demonstrate that our approach achieves better results with a clear advantage. Apart from boosting the accuracy, our stimulus verification enjoys the advantage of a much wider range of application as it can be easily adopted to any prediction model that is capable of

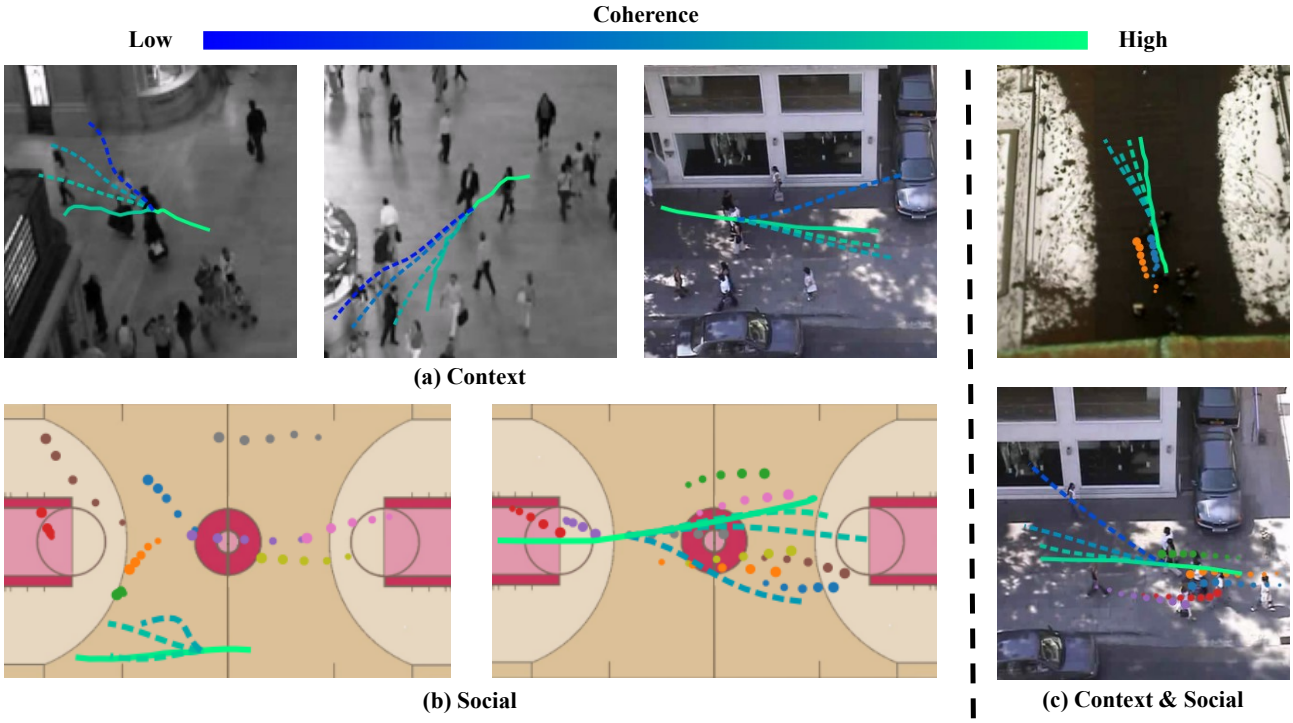

Figure 4. Illustration of predicted trajectories along with their corresponding coherence scores indicated with color. Solid lines refer to the observation and ground truth future trajectory, and dashed lines refer to different predictions. Colored dots in (b,c) refer to the trajectories of social agents, where the direction is from smaller ones to larger ones.

Method	ADE	FDE
GroupNet [43]	0.25	0.44
+*Vanilla NMS*	0.24	0.41
+*Stimulus Verification*	**0.23**	**0.37**

Table 2. Comparison of stimulus verification against vanilla NMS on ETH/UCY benchmark.

N	20	50	100	200
ADE	0.25	0.24	0.23	**0.23**
FDE	0.44	0.39	0.38	**0.37**

Table 3. Comparison between different number of candidates N with GroupNet on the ETH/UCY benchmark.

generating multi-modal predictions, rather than being confined to generative models only.

5. Conclusion

In this paper, we propose stimulus verification as a universal sampling process regarding the trajectory prediction problem. It uses a conditionally parameterized probabilistic model to find the coherence between a trajectory and a certain stimulus. The coherence is further treated as the guidance for stimulus-coherent trajectories, ensuring the selected ones to comply with the constrains of the involved sitimuli. As an external module beyond the base prediction framework, stimulus verification can be easily adopted to any multi-modal approach in a plug-and-play manner. Extensive experiments confirm that significant improvements can be achieved with our approach across various base prediction models and benchmarks.

Method	ADE	FDE
Autoregressive Flow [13]	0.28	0.41
+*DLow* [46]	0.24	0.39
+*LDS* [22]	0.24	0.36
+*Stimulus Verification*	**0.24**	**0.34**
Social GAN [8]	0.61	1.21
+*NPSN* [2]	0.50	0.96
+*Stimulus Verification*	**0.49**	**0.94**

Table 4. Comparison between the stimulus verification and previous sampling approaches on ETH/UCY benchmark.

Acknowledgment This work was supported by the National Key Research and Development Project of China (No. 2021ZD0110704), Shanghai Municipal Science and Technology Major Project (2021SHZDZX0102), Shanghai Qi Zhi Institute, and Shanghai Science and Technology Commission (21511101200).

References

[1] Alexandre Alahi, Kratarth Goel, Vignesh Ramanathan, Alexandre Robicquet, Li Fei-Fei, and Silvio Savarese. Social lstm: Human trajectory prediction in crowded spaces. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 961–971, 2016. 2, 4, 6

[2] Inhwan Bae, Jin-Hwi Park, and Hae-Gon Jeon. Non-probability sampling network for stochastic human trajectory prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6477–6487, 2022. 1, 2, 3, 7, 8

[3] Yuning Chai, Benjamin Sapp, Mayank Bansal, and Dragomir Anguelov. Multipath: Multiple probabilistic anchor trajectory hypotheses for behavior prediction. *arXiv preprint arXiv:1910.05449*, 2019. 1

[4] Changan Chen, Yuejiang Liu, Sven Kreiss, and Alexandre Alahi. Crowd-robot interaction: Crowd-aware robot navigation with attention-based deep reinforcement learning. In *2019 international conference on robotics and automation (ICRA)*, pages 6015–6022. IEEE, 2019. 1

[5] David K Duvenaud, Dougal Maclaurin, Jorge Iparraguirre, Rafael Bombarell, Timothy Hirzel, Alan Aspuru-Guzik, and Ryan P Adams. Convolutional networks on graphs for learning molecular fingerprints. In *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015. 5

[6] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014. 5

[7] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014. 1

[8] Agrim Gupta, Justin Johnson, Li Fei-Fei, Silvio Savarese, and Alexandre Alahi. Social gan: Socially acceptable trajectories with generative adversarial networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2255–2264, 2018. 1, 2, 4, 6, 7, 8

[9] Tsubasa Hirakawa, Takayoshi Yamashita, Toru Tamaki, and Hironobu Fujiyoshi. Survey on vision-based path prediction. In *International Conference on Distributed, Ambient, and Pervasive Interactions*, pages 48–64. Springer, 2018. 1, 2

[10] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997. 5

[11] Boris Ivanovic and Marco Pavone. The trajectron: Probabilistic multi-agent trajectory modeling with dynamic spatiotemporal graphs. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2375–2384, 2019. 2

[12] Xu Jia, Bert De Brabandere, Tinne Tuytelaars, and Luc V Gool. Dynamic filter networks. *Advances in neural information processing systems*, 29, 2016. 2, 3

[13] Durk P Kingma, Tim Salimans, Rafal Jozefowicz, Xi Chen, Ilya Sutskever, and Max Welling. Improved variational inference with inverse autoregressive flow. *Advances in neural information processing systems*, 29, 2016. 6, 7, 8

[14] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013. 1

[15] Vineet Kosaraju, Amir Sadeghian, Roberto Martín-Martín, Ian Reid, Hamid Rezatofighi, and Silvio Savarese. Social-bigat: Multimodal trajectory forecasting using bicycle-gan and graph attention networks. *Advances in Neural Information Processing Systems*, 32:137–146, 2019. 2

[16] Laura Leal-Taixé, Michele Fenzi, Alina Kuznetsova, Bodo Rosenhahn, and Silvio Savarese. Learning an image-based motion context for multiple people tracking. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3542–3549, 2014. 2, 6

[17] Namhoon Lee, Wongun Choi, Paul Vernaza, Christopher B. Choy, Philip H. S. Torr, and Manmohan Chandraker. Desire: Distant future prediction in dynamic scenes with interacting agents. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017. 1, 2

[18] Jiachen Li, Fan Yang, Masayoshi Tomizuka, and Chiho Choi. Evolvegraph: Multi-agent trajectory prediction with dynamic relational reasoning. *Advances in neural information processing systems*, 33:19783–19794, 2020. 6

[19] Junwei Liang, Lu Jiang, Juan Carlos Niebles, Alexander G Hauptmann, and Li Fei-Fei. Peeking into the future: Predicting future person activities and locations in videos. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5725–5734, 2019. 2, 3

[20] linouk23. Nba player movements. https://github.com/linouk23/NBA-Player-Movements, 2016. 2, 6

[21] Wei-Chiu Ma, De-An Huang, Namhoon Lee, and Kris M. Kitani. Forecasting interactive dynamics of pedestrians with fictitious play. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017. 2

[22] Yecheng Jason Ma, Jeevana Priya Inala, Dinesh Jayaraman, and Osbert Bastani. Likelihood-based diverse sampling for trajectory forecasting. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 13279–13288, 2021. 1, 2, 3, 6, 7, 8

[23] Karttikeya Mangalam, Harshayu Girase, Shreyas Agarwal, Kuan-Hui Lee, Ehsan Adeli, Jitendra Malik, and Adrien Gaidon. It is not the journey but the destination: Endpoint conditioned trajectory prediction. In *European Conference on Computer Vision*, pages 759–776. Springer, 2020. 1, 3, 4

[24] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014. 2

[25] Abduallah Mohamed, Kun Qian, Mohamed Elhoseiny, and Christian Claudel. Social-stgcnn: A social spatio-temporal graph convolutional neural network for human trajectory prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14424–14432, 2020. 2

[26] Bo Pang, Yizhuo Li, Yifan Zhang, Muchen Li, and Cewu Lu. Tubetk: Adopting tubes to track multi-object in a one-step training model. In *CVPR*, pages 6308–6318, 2020. 1

[27] Bo Pang, Kaiwen Zha, Hanwen Cao, Jiajun Tang, Minghui Yu, and Cewu Lu. Complex sequential understanding

through the awareness of spatial and temporal concepts. *Nature Machine Intelligence*, 2(5):245–253, 2020. 1

[28] Stefano Pellegrini, Andreas Ess, Konrad Schindler, and Luc Van Gool. You'll never walk alone: Modeling social behavior for multi-target tracking. In *2009 IEEE 12th International Conference on Computer Vision*, pages 261–268. IEEE, 2009. 2, 6

[29] Danilo Rezende and Shakir Mohamed. Variational inference with normalizing flows. In *International conference on machine learning*, pages 1530–1538. PMLR, 2015. 1, 2

[30] Nicholas Rhinehart, Kris M Kitani, and Paul Vernaza. R2p2: A reparameterized pushforward policy for diverse, precise generative path forecasting. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 772–788, 2018. 2

[31] Alexandre Robicquet, Amir Sadeghian, Alexandre Alahi, and Silvio Savarese. Learning social etiquette: Human trajectory understanding in crowded scenes. In *European conference on computer vision*, pages 549–565. Springer, 2016. 2

[32] Andrey Rudenko, Luigi Palmieri, Michael Herman, Kris M. Kitani, Dariu M. Gavrila, and Kai O. Arras. Human motion trajectory prediction: A survey. 2019. 2

[33] Amir Sadeghian, Vineet Kosaraju, Ali Sadeghian, Noriaki Hirose, Hamid Rezatofighi, and Silvio Savarese. Sophie: An attentive gan for predicting paths compliant to social and physical constraints. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1349–1358, 2019. 1, 2, 3, 4, 6, 7

[34] Tim Salzmann, Boris Ivanovic, Punarjay Chakravarty, and Marco Pavone. Trajectron++: Dynamically-feasible trajectory forecasting with heterogeneous data. In *European Conference on Computer Vision*, pages 683–700. Springer, 2020. 1, 2

[35] Liushuai Shi, Le Wang, Chengjiang Long, Sanping Zhou, Mo Zhou, Zhenxing Niu, and Gang Hua. Sgcn: Sparse graph convolution network for pedestrian trajectory prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8994–9003, 2021. 1

[36] sjtuxcx. Groupnet. https://github.com/MediaBrain-SJTU/GroupNet/blob/main/saved_models/nba/pretrain.p, 2022. 7

[37] Kihyuk Sohn, Honglak Lee, and Xinchen Yan. Learning structured output representation using deep conditional generative models. *Advances in neural information processing systems*, 28, 2015. 2

[38] Jianhua Sun, Yuxuan Li, Liang Chai, Hao-Shu Fang, Yong-Lu Li, and Cewu Lu. Human trajectory prediction with momentary observation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6467–6476, 2022. 1

[39] Jianhua Sun, Yuxuan Li, Hao-Shu Fang, and Cewu Lu. Three steps to multimodal trajectory prediction: Modality clustering, classification and synthesis. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 13250–13259, 2021. 1, 2, 6, 7

[40] Jianhua Sun, Zehao Wang, Jiefeng Li, and Cewu Lu. Unified and fast human trajectory prediction via conditionally parameterized normalizing flow. *IEEE Robotics and Automation Letters*, 7(2):842–849, 2021. 1

[41] Zhi Tian, Chunhua Shen, and Hao Chen. Conditional convolutions for instance segmentation. In *European Conference on Computer Vision*, pages 282–298. Springer, 2020. 2, 3

[42] Allan Wang, Christoforos Mavrogiannis, and Aaron Steinfeld. Group-based motion prediction for navigation in crowded environments. In *Conference on Robot Learning*, pages 871–882. PMLR, 2022. 1

[43] Chenxin Xu, Maosen Li, Zhenyang Ni, Ya Zhang, and Siheng Chen. Groupnet: Multiscale hypergraph neural networks for trajectory prediction with relational reasoning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6498–6507, 2022. 1, 2, 6, 7, 8

[44] Cunjun Yu, Xiao Ma, Jiawei Ren, Haiyu Zhao, and Shuai Yi. Spatio-temporal graph transformer networks for pedestrian trajectory prediction. In *European Conference on Computer Vision*, pages 507–523. Springer, 2020. 2

[45] Ye Yuan and Kris Kitani. Dlow: Diversifying latent flows for diverse human motion prediction. In *European Conference on Computer Vision*, pages 346–364. Springer, 2020. 7

[46] Ye Yuan and Kris M Kitani. Diverse trajectory forecasting with determinantal point processes. In *ICLR*, 2020. 1, 2, 3, 8

[47] Bolei Zhou, Xiaogang Wang, and Xiaoou Tang. Understanding collective crowd behaviors: Learning a mixture model of dynamic pedestrian-agents. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 2871–2878. IEEE, 2012. 2, 6

[48] Deyao Zhu, Mohamed Zahran, Li Erran Li, and Mohamed Elhoseiny. Motion forecasting with unlikelihood training in continuous space. In *Conference on Robot Learning*, pages 1003–1012. PMLR, 2022. 2