DeepHybrid: Learned Hybrid Trajectory Prediction via Factored Inference and Adaptive Sampling

Anonymous Author(s)

Abstract: Understanding high-level intent is important for trajectory prediction, a complex multi-modal problem due to uncertainty in human intent. Existing approaches explore the discrete nature of human intent before predicting continuous trajectories, to improve accuracy and support explainability, yet they often assume a fixed intent over time. In this work, we introduce DeepHybrid, a general and expressive hybrid prediction framework. By modeling traffic agents as a hybrid discrete-continuous system, our approach is capable of predicting discrete intent changes over time. We learn the probabilistic hybrid model as a maximum likelihood estimation problem and leverage neural proposal distributions to sequentially sample from the exponentially growing discrete space. The overall approach affords a better trade-off between accuracy and coverage. We train and validate our model on the Argoverse dataset, and demonstrate its effectiveness through comprehensive ablation studies and comparisons with state-of-the-art models.

Keywords: Trajectory Prediction, Factored Inference, Adaptive Sampling

1 Introduction

Predicting future trajectories of traffic agents is a key task for autonomous vehicles. This task is challenging due to multi-modal human intent. There is an inherent trade-off between accurately representing the distribution of trajectories and covering the diversity of potential intents [1, 2, 3]. Several recent works address the trade-off explicitly using a multi-stage approach [4, 5, 6, 7, 8]. First, they infer high-level human intent, such as driving maneuvers and goal locations, to provide task-specific coverage, such as maximizing the space covered by the sampled goals [8]. Next, trajectories are generated conditioned on the intent. The models are trained to maximize the data likelihood to support prediction accuracy. They demonstrate great success in terms of prediction accuracy and coverage, and provide explainability in predicted trajectories. However, the existing approaches often use a simplified intent model that assumes the intent is fixed over time, to keep the prediction space reasonable. In practice, a traffic agent can change its intent (i.e. follow the lane, perform a lane change, and turn), especially over long horizons.

When accounting for the evolving discrete intent, the number of discrete modes grows exponentially in the prediction horizon [9]. This has been studied in the context of factored inference, e.g. by merging and pruning mode hypotheses [10, 11] or sampling from the prediction space [12, 13]. In the domain of trajectory prediction, the exponentially growing discrete space can be eased by expanding the discrete predictions at a few selected points [14] or accounting for the most probable intent [15], but they may not provide sufficient accuracy and coverage in a multi-modal problem.

We propose an approach that better captures both accuracy and coverage by explicitly modeling discrete intent sequences. Our approach DeepHybrid uses a learned probabilistic hybrid automata model, as illustrated in Fig. 1(a), to infer a sequence of high-level discrete modes when generating low-level trajectory predictions. We use neural proposal distributions in the hybrid model and the farthest point sampling algorithm to obtain good coverage of trajectories with only a few samples, while preserving model accuracy. Our contributions are as follows: i) We formulate trajectory prediction as a general and expressive hybrid prediction problem allowing an evolving discrete intent, and learn a probabilistic hybrid automaton model as a deep neural network. ii) We predict a small set of hybrid prediction samples that support both accuracy and coverage via task-specific learned proposal distributions and sample selection. iii) We train and validate our model using a naturalistic driving dataset and perform detailed experiments to demonstrate the effectiveness of our approach.

Submitted to the 5th Conference on Robot Learning (CoRL 2021). Do not distribute.
Figure 1: (a) Graphical model of a hybrid system representing traffic agents, where $Z$, $X$, $C$ represent discrete mode variables, continuous state variables, and context variables, respectively. Arrows indicate variables dependencies over time. (b) Overview of DeepHybrid: Given a learned hybrid model, it leverages the learned proposal distribution to generate hybrid sequence samples (red arrows) from an exponentially growing space, which capture intent changes (red dots) over time and support coverage, and further selects a small set of accurate and diverse trajectories.

2 Related Work

Multi-Modal Trajectory Prediction Trajectory prediction has been studied extensively in the past few years. To account for uncertainty and multi-modality in prediction space, generative adversarial networks (GAN) [16] and variational autoencoders (VAE) [17, 18] are used to generate multiple trajectory predictions by sampling a latent space. Several works have attempted to improve coverage of the possible outcomes [1, 3, 19, 20], yet there is an inherent trade-off between accurately representing the trajectory distribution and covering a diverse set of intents [1, 2, 3]. To account for this trade-off between accuracy and coverage explicitly, hybrid models are proposed to classify discrete intent and generate trajectories conditioned on the intent. The intent is defined by a variety of choices, including driving maneuvers [4, 21, 22], goal locations or waypoints [6, 8, 23, 24, 25, 26], and target lanes [7, 27, 28, 29], etc. In these hybrid approaches, the intent is assumed to be fixed over time. In practice, however, the agent may change its intent over time, especially over a long horizon, or follow different intents to get to the same target location or lane. When accounting for evolving discrete intent, [14] leverages a support vector machine to infer discrete intent over specific decision points, and [15] proposes a discrete choice modelling approach to infer discrete anchors over time, but they either expand discrete predictions at a few selected steps or predict the most-probable intent, to avoid dealing with the exponentially growing discrete space. In this work, we propose a general and expressive hybrid prediction framework that accounts for evolving intent by inferring a sequence of discrete modes over time, and predicting trajectories consistent with the mode sequence.

Factored Inference The discrete prediction space suffers from exponential growth as a function of the prediction horizon. This problem has been addressed in the context of factored inference, by approximating the intractable state space through pruning and sampling techniques. For instance, multiple model estimation algorithms estimate the possible operational modes for a system, and filter states from an exponential number of hypotheses, by merging and pruning hypotheses [10, 11]. Furthermore, [13] models the hybrid system through a hybrid Bayesian network, and proposes a sampling-based approximation algorithm to track hybrid states. In parallel, [30, 12, 31] model hybrid systems through a probabilistic hybrid automaton [32], and apply efficient pruning, search, and sampling methods to maintain reasonable estimation performance. We extend the sampling-based approaches to sample discrete states through a learned proposal function for coverage, and apply pruning to find representative samples when the number of samples allowed is limited.

Efficient Sequential Sampling Our approach combines ideas from sequential trajectory prediction sampling [33], sequential Monte Carlo [34, 35], and adaptive sampling [36, 37, 38], by sequentially sampling from the hybrid model through a proposal distribution, conditioned on previously generated samples. In autonomous driving, only a small set of prediction samples can be afforded, as evaluating each sample for downstream tasks such as risk assessment is expensive [39]. To select a limited number of samples, [3] leverages farthest point sampling (FPS) to choose semantically meaningful samples from a latent space; [6] uses non-maximum suppression (NMS) to prune trajectories that are close to each other to improve coverage; [8] proposes task-specific sub-sampling techniques towards optimizing the evaluation metrics.
3 Problem Formulation

In this section, we introduce the hybrid system model used for DeepHybrid, followed by a formal problem statement on learning this model.

3.1 Hybrid System Modeling

We model a traffic agent as a probabilistic hybrid automaton (PHA) [32]. The PHA is a tuple $\mathcal{H} = (S, W, F, T, S_0, Z)$, where $S = X \cup \{Z\}$ denotes the hybrid state variables – $Z$ denotes discrete mode variables with a finite domain $Z$, and $X$ denotes continuous state variables; $W$ specifies the input/output variables, which consists of context variables $C$, continuous observation variables $O_X$, and discrete observation variables $O_Z$; $F$ specifies the dynamics evolution over continuous variables for each mode, conditioned on the current mode variable, the previous continuous variable, and the context variable; $T$ specifies the discrete transition distribution over discrete modes, conditioned on the previous mode variable, the previous continuous variable, and the context variable; $S_0$ denotes the initial hybrid state.

The dependencies of hybrid state variables in a PHA are depicted as a graphical model in Fig. 1(a), where we omit the observation variables for simplicity. The state evolution is governed by the transition function $T$ and the dynamics function $F$, indicated by the three arrows going to $Z_t$ and to $X_t$, respectively. Compared to existing PHA estimation methods [30, 31] that define the discrete variable to be hidden and unobservable, we assume that the discrete variables are observable for training purposes in the problem of trajectory prediction, by instantiating the discrete intent as driving maneuvers or goal locations, and defer learning with hidden intent for future work.

3.2 Hybrid Model Learning

Given a set of observed discrete-continuous future agent states $O = (O_X, O_Z)$ and context states $C$, we want to find a model parameterized by $\theta$ that maximizes the following data log likelihood [30], as a maximum likelihood estimation (MLE) problem:

$$L_{\text{MLE}}(O, C) = \sum_{o \in O} \log p(o; \theta) = \sum_{o \in O} \sum_{t=1}^{H} \log p_F(o_t^X|o_{t-1}^X, o_t^Z, c; \theta) + \log p_T(o_t^Z|o_{t-1}^Z, o_{t-1}^X, c; \theta),$$  \hspace{1cm} (1)

where $o = \{o_t^X, o_t^Z\}_{t=1}^{H}$ is an observed future hybrid trajectory sequence with horizon $H$, $p_F$ is a Gaussian distribution over continuous states, $p_T$ is a categorical distribution over discrete modes, and the prior for the observations is omitted as we assume that the observation noise is negligible, as common in most trajectory prediction literature.

**Hybrid Model Learning Problem [30]:** Given a set of future observation data, the goal is to learn the model $p_F$ and $p_T$ such that the data log likelihood defined in Eq. (1) is maximized.

4 Approach

In this section, we introduce our approach to learn a PHA through an encoder-decoder deep neural network model, as depicted in Fig. 2. The encoder encodes the belief about the context information $c$ as a hidden state vector $h_0$, and the decoder samples a sequence of hybrid states $\{s_t^H\}_{t=1}^{H}$ up to a finite horizon through a hybrid model, conditioned on $h_0$. Given the learned model, we use the decoder to sample multiple predictions and apply farthest point sampling in the continuous trajectory space to generate a small set of predictions with good coverage, as visualized in Fig. 1(b).

4.1 Encoder

In the encoder, we first encode the context information $c$, including the observed path of the target agent $c_F$ and the observed map data $c_M$, per time step in the past. The observed path at each step is encoded through a multi-layer perceptron (MLP), and the map is encoded through a model based on [40], which takes the map input as a set of lane centerlines, and performs self-attention to pool the encoded states from all centerlines. Next, we run the encoded context state $c_c$ through an LSTM network to get the hidden context state vector $h_0$ at the most recent observed step $t = 0$. 

3
Figure 2: Overview of the proposed deep neural network. The encoder encodes the belief about the context information, such as observed path \( c_P \) and map \( c_M \), and passes the combined encoded state \( c_h \) through an LSTM network to obtain the hidden state vector \( h_0 \). The decoder is another LSTM that generates a sequence of hybrid states through a learned hybrid model, including a transition function \( T \), a dynamics function \( F \), and a proposal distribution \( Q \) for improving task-specific performance.

### 4.2 Decoder

In the decoder, we generate a sequence of hybrid states, including discrete modes and continuous positions, using another LSTM model. At each time step, the LSTM takes as inputs the hidden state \( h_{t-1} \) and the hybrid state sample \( s'_{t-1} \) from the previous step, and outputs a new hidden state \( h_t \) and an output state \( y_t \). The output LSTM state \( y_t \) is passed through a transition function \( T \), modeled as an MLP layer that produces a categorical distribution \( P_T(Z_t) \) over all possible discrete modes. We leverage a Gumbel-softmax sampler to sample the next mode \( z_t' \) from the distribution, which is then concatenated with \( y_t \) and fed into a dynamics function \( F \) as an MLP layer that outputs the distribution of the continuous state \( P_F(X_t) \), parameterized as a Gaussian distribution with mean \( X_t' \) and unit variance, which is chosen arbitrary for stable training and is assumed in models such as [6]. We then sample a continuous state \( x_t' \) from the distribution. The predicted hybrid sample \( s_t' = (z_t', x_t') \) at time step \( t \) and the hidden state \( h_t \) are used to generate the sample in the next step, until the prediction horizon is reached.

The decoder is useful in two places. First, given an observation of future hybrid state sequence, we can compute its likelihood by plugging it into the LSTM model, and compute the log likelihood as the summation of the discrete log-likelihood from \( P_T \) and continuous log-likelihood from \( P_F \), using Eq. (1). This allows us to faithfully optimize the hybrid model, by maximizing the log-likelihood given the observations. Second, we can sample from the decoder to obtain a set of hybrid prediction trajectories when it is impractical to infer all future options. Each sample is predicted with a closed-form likelihood, which is advantageous to existing sampling-based methods such as GANs [16, 3] and VAEs [18] that only approximate the likelihoods.

### 4.3 Learned Proposal Distributions

Given a learned hybrid model, it is impractical to infer all discrete intent options, as the number of options increases exponentially at each time step. Many factored inference algorithms [13, 12] therefore aim to sample from the hybrid model as an approximation; however, it may take a large number of samples to sufficiently cover the prediction space. We propose to learn a proposal function \( Q \) on the top of the transition function, to sample the mode for the task of achieving better prediction accuracy and coverage. The proposal function is used to sample per time step, as opposed to sampling a full trajectory at once, to avoid a large proposal state. Inspired by adaptive sampling, the proposal function takes the same input as the transition function, in addition to the previously generated trajectory sequence samples \( s^{(P)} \), and outputs a categorical distribution over the next discrete modes. More specifically, each previous sample is encoded through an MLP layer, and the proposal function takes the max pooling of all the previous sample encodings and passes the inputs through an MLP layer to generate logits over each mode.

To train the proposal function for accuracy and coverage, we generate \( K \) trajectory sequences \( \{ s^{(k)} = (z^{(k)}, x^{(k)}) \}_{k=1}^K \) sequentially from the decoder, and compute the min-of-\( K \) L2 loss com-
pared to the ground truth continuous observations $o_X$:
\[
\mathcal{L}_Q = \min_{k \in K} \| \mu^{(k)} - o_X \|_2.
\]

There exist a few other options to learn the proposal function for coverage, such as maximizing entropy [41]. In this paper, we focus on the task of improving the diversity of the continuous trajectories when guaranteeing prediction accuracy, and choose the min-of-$K$ L2 loss (or variety loss [16]) that is widely used in the multi-modal trajectory prediction literature. While it is possible to train the model with only the min-of-$K$ L2 loss to favor towards prediction coverage, as in [16, 3], it leads to a diluted probability density function compared to the ground truth [2]. Therefore, we choose to improve prediction coverage while ensuring accuracy, by introducing the data likelihood loss in Eq. (1). As a result, we can leverage the proposal distribution to generate representative samples, while obtaining the real probability of these samples from the transition function. To encourage the proposal distribution to be close to the transition distribution, we add a regularization loss on the L2 differences between the two distribution logits $\mathcal{L}_{\text{reg}}$.

### 4.4 Trajectory Sample Selection

In many autonomous vehicle applications, we can only afford a small set of prediction samples, due to the non-trivial computational complexity of evaluating these samples for risk assessment [39]. To further improve coverage and boost prediction performance with a limited budget on samples, we propose to use the farthest point sampling (FPS) algorithm [42, 43] to select trajectories that are far away from each other from samples generated from the proposal distribution, while maintaining their probabilities through the learned hybrid model. The algorithm works by selecting the next sample that is farthest away from the previously selected samples, in terms of the distance between end locations, with the first sample selected with the highest likelihood. FPS is able to capture the majority of distinct options thanks to its 2-optimal coverage property [42, 43].

### 4.5 Model Training and Inference

In training time, we jointly train the hybrid model and the proposal distribution with the loss
\[
\mathcal{L} = -\mathcal{L}_{\text{MLE}} + \alpha \mathcal{L}_Q + \beta \mathcal{L}_{\text{reg}},
\]
where the MLE term is negated as a loss to minimize, and $\alpha$ and $\beta$ are the loss coefficients.

At inference time, we i) sequentially call the hybrid model $M$ times with the proposal function to generate $M$ hybrid trajectory sequences, ii) compute their likelihoods based on the probabilities from the transition function and the dynamics function, and iii) perform FPS to select the final $N$ trajectory samples, and normalize the probabilities of each sample so that they sum up to 1.

### 5 Experimental Results

In this section, we introduce the dataset and the model details, followed by a series of experiments demonstrating the effectiveness of our approach and probing its components.

#### 5.1 Dataset and Model Details

We train and validate DeepHybrid on Argoverse v1.1 [44], a widely used benchmark for single agent trajectory prediction. The data contains 324,557 segments of agent trajectories, including two seconds of observed trajectories and three seconds of trajectories to predict, sampled at 10Hz, as well as map information such as lane centerlines. We augment the dataset with discrete mode labels, defined as stop, fast forward, slow forward, left turn, right turn, depending on the velocity and angular changes differentiated from the trajectories. We run a Gaussian process filter [45] to smooth the noisy trajectories prior to auto-labeling the maneuvers over time.

In the encoder, DynamicsNet is an MLP with 32 neurons; MapNet utilizes a similar structure as VectorNet [40]; the encoder LSTM has a hidden size of 32 and an output dimension of 32. In the decoder, the transition function and the proposal function use a two-layer MLP with (32, 5) neurons followed by a softmax layer; the dynamics function is a two-layer MLP with (32, 2) neurons; the
Table 1: Min-of-6 ADE/FDE using different discrete function choices. Our proposed adaptive proposal function, which is learned to optimize min-of-$K$ losses, achieves the lowest errors.

<table>
<thead>
<tr>
<th>Discrete Function</th>
<th>1 Second</th>
<th>3 Seconds</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transition Function</td>
<td>0.45</td>
<td>1.19</td>
</tr>
<tr>
<td>Proposal Function (non-Adaptive)</td>
<td>0.44</td>
<td>1.00</td>
</tr>
<tr>
<td>Proposal Function (Adaptive)</td>
<td><strong>0.33</strong></td>
<td><strong>0.86</strong></td>
</tr>
</tbody>
</table>

235 
In the study, we first generate samples using the proposal distribution (or the discrete distribution for the bottom row), and select $N = 6$ samples. The results are summarized in Table 2. When $M = N = 6$, no subsampling occurs. When $M > N$, a random sampler and a best sampler do not improve the errors, as selecting only the most likely samples leads to worse errors, since trajectory prediction is a multi-modal problem. As $M$ grows, the best sampler acts similar to a maximum likelihood estimator, and exhibits inferior results as the problem is multi-modal. NMS improves results but is limited by a fixed distance threshold: when the threshold is small (i.e. 2 meters), it fails to provide enough coverage in cases where the predicted samples are very far away; when the threshold is large (i.e. 4 meters), the number of valid samples can be smaller than $N$. On the other hand, FPS reduces the errors the most, by finding the 6 samples that provide both accuracy and coverage. When $M$ is larger than 50, the error reduction is small for both NMS and FPS. We note that without the learned proposal distribution, FPS does not achieve the same results, as the discrete structure is not explored efficiently in the samples generated from the transition function.

240 

245 

250 

255 

260 

265 

270 

275 

280 

285
Table 2: Min-of-6 ADE/FDE over 3 seconds using different sample selection methods. FPS achieves the best performance by selecting 6 samples generated from the proposal distributions.

<table>
<thead>
<tr>
<th>Selection Method</th>
<th>6 / 6 samples</th>
<th>6 / 30 samples</th>
<th>6 / 50 samples</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>minADE</td>
<td>minFDE</td>
<td>minADE</td>
</tr>
<tr>
<td>Proposal + Random</td>
<td>0.86</td>
<td>1.68</td>
<td>1.02</td>
</tr>
<tr>
<td>Proposal + Best</td>
<td>0.86</td>
<td>1.68</td>
<td>1.10</td>
</tr>
<tr>
<td>Proposal + NMS (2m)</td>
<td>0.86</td>
<td>1.68</td>
<td>0.76</td>
</tr>
<tr>
<td>Proposal + NMS (4m)</td>
<td>0.86</td>
<td>1.68</td>
<td>0.80</td>
</tr>
<tr>
<td>Proposal + FPS</td>
<td>0.86</td>
<td>1.68</td>
<td>0.74</td>
</tr>
<tr>
<td>Transition + FPS</td>
<td>1.19</td>
<td>2.43</td>
<td>1.06</td>
</tr>
</tbody>
</table>

Table 3: NLL and min-of-6 ADE/FDE over 3 seconds compared to hybrid prediction baselines. Our model balances between accuracy and coverage, with a variant (DeepHybrid-C) trained on the coverage task achieving the lowest minADE.

<table>
<thead>
<tr>
<th>Model</th>
<th>NLL</th>
<th>minADE</th>
<th>minFDE</th>
</tr>
</thead>
<tbody>
<tr>
<td>DESIRE [17]</td>
<td>-</td>
<td>0.92</td>
<td>1.77</td>
</tr>
<tr>
<td>MultiPath [48]</td>
<td>-</td>
<td>0.80</td>
<td>1.68</td>
</tr>
<tr>
<td>TNT [6]</td>
<td>-</td>
<td>0.73</td>
<td>1.29</td>
</tr>
<tr>
<td>DeepHybrid-C</td>
<td>34.54</td>
<td>0.66</td>
<td>1.27</td>
</tr>
<tr>
<td>DeepHybrid</td>
<td>30.87</td>
<td>0.72</td>
<td>1.26</td>
</tr>
</tbody>
</table>

5.3 Quantitative Results

We compare our full model with a number of representative baselines, including i) DESIRE [17] that utilizes a conditional VAE model to generate trajectory prediction samples from a latent space; ii) MultiPath [48] that learns the trajectory modalities as a set of anchors and predicts trajectories through anchor classification and offset regression; iii) TNT [6] that first infers discrete target locations and second predicts target-conditioned trajectories to support multi-modality; iv) DeepHybrid-C that is a variant of our model but is trained with only the task-specific coverage loss, defined in Eq. (2), which shares the same spirit as [15] that optimizes for the minimum loss, but leverages our discrete adaptive sampling and FPS approaches. We use the metrics reported in [6], and present the comparison in Table 3. DeepHybrid outperforms all baselines that assume a fixed intent over time. We further improve the minADE metric with a variant, DeepHybrid-C, that is solely trained towards optimizing this metric, as indicated by the fourth row, but sacrifices accuracy measured by the negative log-likelihood metric (NLL) [49]. Our method, on the other hand, allows for the trade-off between accuracy and coverage.

The work that is closest to ours in spirit is ManeuverLSTM [4], which models driving modes explicitly as maneuvers labeled from trajectory data, and assumes the maneuver is fixed over time. For a fair comparison, we use the same context encoders and the same definition of maneuvers used in our model. We use five samples for comparison given the number of maneuvers defined for ManeuverLSTM. In addition to the standard metrics, we introduce min-of-K discrete error rate (minDER) that measures the percentage of wrong predictions in discrete states for the best predicted sample, to quantify the discrete prediction accuracy and coverage. Table 4 demonstrates that our model outperforms this baseline by a large margin, in both continuous and discrete metrics, by supporting evolving maneuver intent and utilizing a proposal function to explore the intent space.

5.4 Qualitative Results

In Fig. 3, we present a qualitative example to demonstrate the effectiveness of FPS. Fig. 3(a) shows the most likely examples selected based on the predicted likelihood, which favor the option to follow the middle lane or merge to the right lane. On the other hand, NMS selects more diverse samples, but

<table>
<thead>
<tr>
<th>Model</th>
<th>minADE</th>
<th>1 Second minFDE</th>
<th>3 Seconds minFDE</th>
<th>minDER</th>
<th>3 Seconds minDER</th>
</tr>
</thead>
<tbody>
<tr>
<td>ManeuverLSTM [4]</td>
<td>0.41</td>
<td>0.52</td>
<td>1.06</td>
<td>1.94</td>
<td>11.01%</td>
</tr>
<tr>
<td>DeepHybrid</td>
<td>0.32</td>
<td>0.40</td>
<td>0.80</td>
<td>1.47</td>
<td>7.65%</td>
</tr>
</tbody>
</table>

Table 4: Comparison to ManeuverLSTM. DeepHybrid achieves better performance in both discrete and continuous error metrics.
Figure 3: Sample selection using different methods: (a) best (purple), (b) NMS (2m) (green), (c) NMS (4m) (orange), (d) FPS (red). Ground truth past and future trajectory are in blue and cyan. Predicted samples in grey. Numbers indicate the sampling order. FPS achieves the best coverage.

Figure 4: Comparison between DeepHybrid (red) and ManeuverLSTM (olive) predictions, where ground truth future trajectories are in cyan, and red dots depict where the mode changes in DeepHybrid predictions. (a) In a lane change scenario, DeepHybrid identifies multiple time slots, highlighted in black circles, to change the intent. (b) In a turning scenario, DeepHybrid predicts a fast turn and a slow turn, highlighted in black circles, that is transitioned from a forward maneuver. ManeuverLSTM has worse accuracy due to the assumption of fixed maneuvers over time.

As a more robust and threshold-free alternative, FPS finds diverse options more effectively.

In conclusion, we present a general and expressive hybrid prediction model that accounts for evolving discrete modes in the future trajectory, and leverages learned proposal functions and farthest point sampling technique to select a small number of accurate and diverse samples from an exponential space. The effectiveness of our model is validated in the Argoverse dataset, through both quantitative and qualitative experiments.
References


