

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

Xin Huang¹, Guy Rosman², Igor Gilitschenski², Ashkan Jasour¹,
 Stephen G. McGill², John J. Leonard^{1,2}, Brian C. Williams¹

Abstract—Modeling multi-modal high-level intent is important for ensuring diversity in trajectory prediction. Existing approaches explore the discrete nature of human intent before predicting continuous trajectories, to improve accuracy and support explainability. However, these approaches often assume the intent to remain fixed over the prediction horizon, which is problematic in practice, especially over longer horizons. To overcome this limitation, we introduce HYPER, a general and expressive hybrid prediction framework that models evolving human intent. 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 via a maximum likelihood estimation problem and leverage neural proposal distributions to sample adaptively 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.

I. 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 model high-level human intent as a discrete state and infer such intent to provide sufficient coverage over multi-modal human intent. 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 is 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

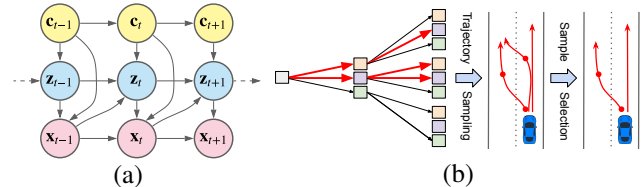


Fig. 1: (a) Graphical model of a hybrid system representing traffic agents, where z , x , c represent discrete mode variables, continuous state variables, and context variables, respectively. Arrows indicate variable dependencies over time. (b) Overview of HYPER: Given a learned hybrid model, it leverages 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 chooses a small set of diverse trajectories using a sample selection algorithm.

discrete space can be mitigated by expanding the discrete predictions at a few selected points [14] or accounting for the most probable intent [15]; however, 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, HYbrid trajectory PrEdictor (HYPER), uses a learned probabilistic hybrid automata model, as illustrated in Fig. 1(a), to jointly infer a sequence of high-level discrete modes when generating low-level trajectory predictions. We use neural proposal distributions [16] 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 leverage a learned proposal function to sample *adaptively* from an exponentially growing discrete space in the hybrid model to support both *accuracy and coverage*, and utilize a sample selection technique to further improve prediction performance given limited samples. iii) We train and validate our model on a naturalistic driving dataset and perform detailed experiments to test our hypothesis and demonstrate the effectiveness of our approach.

II. 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,

¹Computer Science and Artificial Intelligence Laboratory, MIT, Cambridge, MA 01239, USA xhuang@csail.mit.edu

²Toyota Research Institute, Cambridge, MA 02139, USA. This article solely reflects the opinions and conclusions of its authors and not TRI or any other Toyota entity.

generative adversarial networks (GAN) [17] and variational autoencoders (VAE) [18], [19] 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], [20], [21], 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 continuous trajectories conditioned on the intent. The intent is defined by a variety of choices, including driving maneuvers [4], [22], [23], goal locations or waypoints [6], [8], [24], [25], and target lanes [7], [26], [27], 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 modeling approach to infer discrete anchors over time. Such models 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, [28], [12], [29], [30] model hybrid systems through a probabilistic hybrid automaton (PHA) [31] or a switching linear dynamical systems (SLDS) [32], and apply efficient pruning, search, and sampling methods to maintain reasonable estimation performance. Existing factored inference methods often assume a linear system to obtain closed-form or trackable solutions, and do not fully utilize the relevant context information. In this work, we model a hybrid system as deep neural networks, which excel at modeling nonlinear agent dynamics and learning complex environment contexts such as map information.

Trajectory Sampling Many trajectory prediction methods [33], [17], [19] sample multiple predictions in parallel from a learned distribution. Our approach utilizes a sequential adaptive sampling technique to generate samples through a learned proposal distribution, conditioned on previously generated samples. This allows the model to provide more coverage with limited samples. The learned adaptive proposal

function is inspired by a few ideas from sequential trajectory prediction sampling [34], sequential Monte Carlo [35], [16], [36], and adaptive sampling [37], [38], [39]. Compared to existing sequential Monte Carlo methods, our proposal function is used to sample per time step, as opposed to sampling a full trajectory at once, to avoid a large proposal state.

In many applications such as 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 [40]. To select a limited number of candidates from all predicted samples, [3], [21] leverage diverse sampling techniques 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. Similar to [6], our approach selects samples directly over the predicted trajectories, offering better interpretability.

III. PROBLEM FORMULATION

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

A. Hybrid System Modeling

We model a traffic agent as a *probabilistic hybrid automaton* (PHA) [31]. Compared to a hidden Markov model [41], the transitions in PHA have an autonomous property [12], i.e. the discrete mode evolution depends on the continuous state. This property provides better model capacity in mimicking the behavior of traffic agents. The PHA is a tuple $\mathcal{H} = \langle \mathbf{s}, \mathbf{w}, F, T, \mathbf{s}_0, \mathcal{Z} \rangle^1$, where $\mathbf{s} = \mathbf{x} \cup \mathbf{z}$ denotes the hybrid state variables – \mathbf{z} denotes the discrete mode with a finite domain \mathcal{Z} , and $\mathbf{x} \in \mathbb{R}^{n_x}$ denotes continuous state variables; \mathbf{w} specifies the input/output variables, which consists of context variables \mathbf{c} , continuous observation variables \mathbf{o}_x , and discrete observation variables \mathbf{o}_z ; $F : \mathcal{Z} \rightarrow \mathcal{F}$ specifies the continuous evolution of the automaton for each discrete mode, in terms of a set of discrete-time difference equations \mathcal{F} over the variables \mathbf{x} and \mathbf{c} ; $T : \mathcal{Z} \rightarrow \mathcal{T}$ specifies the discrete evolution of the automaton for each discrete mode, as a finite set of transition probabilities drawn from \mathcal{T} ; \mathbf{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 \mathbf{z}_t and to \mathbf{x}_t , respectively.

Similar to existing trajectory prediction approaches, we assume that the discrete labels are observable at training time, defined as maneuvers [4], [22], [23] or goal locations [6], [8]. These labels can be obtained through auto-labelling or unsupervised clustering over continuous trajectories. While it is challenging to perfectly label driver intent, we show in the experiments that our model is robust to imperfect labels, and defer learning with hidden intent for future work.

¹We use the lowercase bold symbols to denote both the set of variables and the vector, and lowercase standard symbols for variable instantiations.

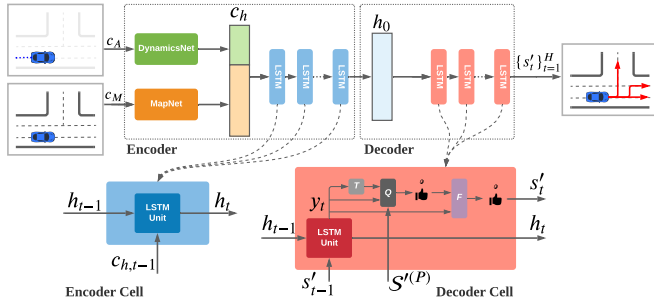


Fig. 2: Overview of HYPER. The encoder encodes the context information, such as observed path c_A and map c_M , and passes the combined encoded state c_h through an LSTM network to obtain the hidden state vector h_0 . The decoder is another LSTM that generates a sequence of hybrid states through a learned hybrid model, including a transition function T , a dynamics function F , and a proposal distribution Q for improving coverage performance. Thumb and coin sign in the decoder cell indicates random sampling.

B. Hybrid Model Learning

Given a set of observed discrete-continuous future agent states $O = (O_x, O_z)^2$ and context states C , we want to learn a hybrid model parameterized by θ that maximizes the following data log likelihood [28], as a maximum likelihood estimation (MLE) problem:

$$\begin{aligned} \mathcal{L}_{\text{MLE}}(O, C) &= \sum_{o, c \in (O, C)} \log p(o|c; \theta) \\ &= \sum_{o, c} \sum_{t=1}^H \log p_F(o_x^t | o_x^{t-1}, o_z^t, c; \theta) + \log p_T(o_z^t | o_x^{t-1}, o_z^{t-1}, c; \theta), \end{aligned} \quad (1)$$

where $o = \{(o_x, o_z)^t\}_{t=1}^H$ 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. The prior for the observations is omitted as we assume that the observation noise is negligible, which is common in most trajectory prediction literature.

IV. APPROACH

In this section, we introduce our approach³ to learn a PHA-based encoder-decoder deep neural network model, as depicted in Fig. 2. The encoder embeds the context information c into a hidden state vector h_0 , and the decoder samples a sequence of hybrid states $\{s'_t\}_{t=1}^H$ 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).

A. Encoder

In the encoder, we first encode the context information c , including the observed path of the target agent c_A and the observed map data c_M , per time step in the past. All features

are normalized to an agent-centric frame with respect to the last observed position of the target agent, as in [42]. The observed path at each step is encoded through a multi-layer perceptron (MLP). The map is a static feature that is encoded through a model based on [42], which takes the map input as lane centerlines within 80 meters of the last observed agent position, and performs self-attention to pool the encoded states from all centerlines. Next, we run the encoded context state c_h through an LSTM network, which is commonly used in handling sequential data in trajectory prediction [34], [4], [3], to get the hidden context state vector h_0 at the most recent observed step $t = 0$.

B. Decoder

In the decoder, we generate a sequence of hybrid states, including discrete modes and continuous positions, using another LSTM network. At each time step, the LSTM unit 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^4 , modeled as an MLP layer that produces a categorical distribution $P_T(z_t)$ over discrete modes at time t as logits.

Existing factored inference algorithms [13], [12] sample from T to obtain discrete samples; however, naively sampling from T may take a large number of samples to sufficiently cover the prediction space. Therefore, we propose learning an additional *proposal function* Q on the top of the transition function, to sample the discrete mode for the *task* of achieving better prediction accuracy and coverage. The proposal function takes input from i) the distribution from the transition function T , ii) the output LSTM state y_t , and iii) the set of previously generated trajectory sequence samples $\mathcal{S}'^{(P)}$, and outputs a categorical distribution over the next discrete modes as logits. The last input allows us to sample adaptively conditioned on previously generated samples. Each previous full trajectory sample is encoded through an MLP layer. The max pooling of all sample encodings is passed through an MLP layer before being fed into the proposal function. Compared to existing sequential Monte Carlo methods [35], [16], [36], our proposal function is used to sample per time step, as opposed to sampling a full trajectory at once, to avoid a large proposal state.

Given the output of Q , we leverage a Gumbel-softmax sampler to sample the next mode z'_t , whose real probability is obtained from T . The mode is concatenated with y_t and fed into a dynamics function F as an MLP layer that outputs the distribution of the continuous state $P_F(x_t)$. The distribution is parameterized as a Gaussian distribution with mean μ_{x_t} and unit variance, which is chosen arbitrarily 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

²While the continuous state can be directly observed from perception systems, the discrete observation can be estimated from continuous observations, as discussed in Sec. III-A.

³Code at github.com/ToyotaResearchInstitute/tristan.

⁴By definition, the transition function takes the previous state and the context information directly as inputs, as in Fig. 1(a). We follow existing sequential prediction models [43], [33], [17] to read its inputs through an LSTM, and abuse T to represent the auxiliary transition function that takes the LSTM output y_t .

hidden state h_t are used to generate the sample in the next step, until the prediction horizon is reached.

Transition Function vs. Proposal Function: Although both T and Q output a categorical distribution over modes, they serve different purposes. The transition function T is part of the hybrid model defined in Sec. III-A, and it is used to compute the *real* probability of a sample. For instance, given an observation of a future hybrid state sequence, we can compute its likelihood by plugging it into the LSTM model (i.e. through T and F). The likelihood is the summation of the discrete log-likelihood from P_T and continuous log-likelihood from P_F , as in Eq. (1). It allows us to faithfully optimize the hybrid model, by maximizing the log-likelihood given the ground truth future observations, as defined in Sec. III-B.

On the other hand, the proposal function Q determines which samples to generate, in order to improve prediction coverage in an exponentially growing space. It does not represent the true sample weight, which is determined by the transition function T . The weighted sample set allows us to cover the prediction distribution efficiently with only a few samples, which is advantageous to existing sampling-based methods that require a large number of samples to approximate the probability distribution. In Sec. V, we show a use case of sample weights to quantify prediction accuracy as negative log-likelihood.

C. Learned Proposal Distributions

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 $\mathcal{L}_Q = \min_{k \in K} \|x^{(k)} - o_x\|_2^2$.

There exist a few other options to learn the proposal function for coverage, such as maximizing entropy [44]. 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 [17]) 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 [17], [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 and the smoothness of the trajectory, we add a regularization loss \mathcal{L}_{reg} penalizing i) the L2 difference between the two distribution logits and ii) the L2 loss over the second derivative of the predicted trajectory.

D. 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 [40]. To further improve coverage and boost prediction performance with a limited budget on samples, we propose to use the farthest point sampling (FPS) algorithm [45]. The algorithm selects 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 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 [45], as we show in Sec. V-B.2 on how it captures diverse samples with the proposal distribution.

E. Model Training and Inference

In training time, we train the entire model end-to-end,

$$\mathcal{L} = -\mathcal{L}_{\text{MLE}} + \alpha\mathcal{L}_Q + \beta\mathcal{L}_{\text{reg}}, \quad (2)$$

where the MLE term (c.f. Eq. (1)) is negated as a loss to minimize, and α and β 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.

V. 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 compared to baselines.

A. Dataset and Model Details

We train and validate HYPER on Argoverse v1.1 [46], 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 offline with discrete mode labels over time, defined as *stop*, *fast forward*, *slow forward*, *left turn*, *right turn*, depending on the velocity and angular changes differentiated from the trajectories. Despite having a short prediction horizon, almost 40% of Argoverse data exhibits evolving intents with more than one label in three seconds, as we demonstrate in the experiments.

In the encoder, DynamicsNet is an MLP with 32 neurons; MapNet utilizes a similar structure as VectorNet [42]; the encoder LSTM has one layer with 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 sampler is a Gumbel-Softmax sampler [47] that produces

Discrete Function	1 Second		3 Seconds	
	minADE	minFDE	minADE	minFDE
Transition	0.45	0.62	1.19	2.43
Proposal (non-Adapt.)	0.44	0.48	1.00	1.92
<i>Proposal (Adaptive)</i>	0.33	0.44	0.86	1.68

TABLE I: Min-of-6 errors using different discrete function choices. Our proposed adaptive proposal function achieves the lowest errors.

differentiable samples; the decoder LSTM has the same structure as the encoder LSTM. All MLPs are followed by ReLU and dropout layers with a rate of 0.1.

The loss coefficients α and β in Eq. (2) are 1. The sizes of samples K, M, N are 6, 50, and 6, respectively. The model is optimized using Adam [48] and trained on a single NVIDIA Tesla V100 GPU, with a batch size of 16 and a learning rate of $1e-3$. It has 164K parameters and takes approximately 17 milliseconds to generate 50 predicted trajectories and 2 milliseconds to select 6 samples using FPS, making our model suitable in real-time systems.

We adopt the metrics used by the Argoverse benchmark to evaluate the prediction performance, including minimum average displacement error (minADE), minimum final displacement error (minFDE), a variant of minFDE that adds a probability penalization term (brier-minFDE), and miss rate (MR). All statistics are collected in the Argoverse validation dataset. For sampling-based methods (i.e. Gumbel-softmax), we run them five times and take the average. We annotate the method used in our model with *italics* font in the tables.

B. Ablation Study

1) *Learned Adaptive Proposal Distribution*: We demonstrate the contribution of the learned proposal distribution by comparing different options of discrete intent proposal functions, including i) the learned transition function T (i.e. setting α and β to 0), ii) the non-adaptive proposal function that with no access to the other samples, and iii) our proposed proposal function Q that samples adaptively by considering previously generated samples. The results are summarized in Table. I, in which we obtain 6 samples *without* further sample selection, and compute the errors of the best sample. We observe that the proposal functions, learned to optimize the minimum errors, result in better metrics compared to the discrete function, especially through adaptive sampling.

2) *Trajectory Sample Selection*: We validate the effectiveness of our sample selection method, FPS, by comparing it with a few standard options, including i) a random sampler picking samples based on their weights, ii) a most-likely sampler that selects the top likely samples, similar to selecting the most-probable intent in [15] and best- k enumeration in [29], iii) a sampler based on non-maximum suppression (NMS), as used in [6], which selects samples greedily by finding the next sample that is distant enough from existing samples given a threshold. For a fair comparison, the distance measure in NMS is the same as FPS based on final locations, and we empirically choose 2 different distance thresholds (2 meters and 4 meters) to select the next sample. If the number of valid NMS samples is smaller than N , we select

the remaining samples randomly.

In the study, we first generate $M \in \{6, 30, 50\}$ samples using the proposal distribution, and select $N = 6$ samples. The results are summarized in Table II. When $M = N = 6$, no subsampling occurs. When $M > N$, a random sampler and a most-likely 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 most-likely 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.

C. Quantitative Results

We compare our full model with a number of representative baselines, including i) **DESIRE** [18] that utilizes a conditional VAE model to generate trajectory prediction samples from a latent space; ii) **DiversityGAN** [3] that predicts diverse samples by learning a latent space in a GAN model such that the prediction samples with different semantic meanings are far away in that latent space. This baseline is similar to other diverse sampling works that improve coverage through a learned latent space [21]; iii) **MultiPath** [49] that learns the trajectory modalities as a set of anchors and predicts trajectories through anchor classification and offset regression; iv) **TNT** [6] that first infers discrete target locations and second predicts target-conditioned trajectories to support multi-modality. In addition, we introduce a few variants of our models to validate our hypothesis, including v) **SingleMode** that assumes a single mode and only samples from the continuous distribution; vi) **HYPER-Linear** that uses only linear layers in the decoder to simulate a linear dynamic system as in existing factored inference literature that assumes linear dynamics; vii) **HYPER-NoMap** that does not use the map as the input feature; viii) **HYPER-Coverage** that is trained with only the task-specific *coverage* loss. It shares the same spirit as [15] that optimizes for prediction coverage; ix) **HYPER-Perturb** that is trained on the same dataset, in which we randomly perturb 5% of discrete labels with a different mode chosen uniformly.

We use the metrics reported in [6], and present the comparison in Table III. HYPER outperforms all baselines that assume a fixed intent over time by using a fixed high-level intent state in the trajectory decoder (i-iv), ignore the discrete structure in the model (v), presume linear dynamics (vi), or ignore the map features (vii). In order to demonstrate its robustness with noisy discrete labels, we randomly perturbed 5% of discrete labels and observe that our model trained on the perturbed data achieves similar results (ix). We further improve the minADE metric with a variant,

Selection Method	6 / 6 samples		6 / 30 samples		6 / 50 samples	
	minADE	minFDE	minADE	minFDE	minADE	minFDE
Proposal + Random	0.86	1.68	1.02	2.27	0.98	2.12
Proposal + Most-likely	0.86	1.68	1.10	2.45	1.13	2.51
Proposal + NMS (2m)	0.86	1.68	0.76	1.38	0.73	1.30
Proposal + NMS (4m)	0.86	1.68	0.80	1.57	0.78	1.49
<i>Proposal + FPS</i>	0.86	1.68	0.74	1.30	0.72	1.26

Table II: Accuracy over 3s for various sample selection methods. FPS achieves the best performance by selecting 6 samples from the proposal distributions.

Model	NLL	minADE	minFDE	MR
DESIRE [18]	-	0.92	1.77	0.18
DiversityGAN [3]	-	1.13	2.20	0.17
MultiPath [49]	-	0.80	1.68	0.14
TNT [6]	-	0.73	1.29	0.09
<i>SingleMode</i>	78.46	0.87	2.00	0.22
<i>HYPER-Linear</i>	45.86	0.79	1.38	0.12
<i>HYPER-NoMap</i>	40.55	0.77	1.35	0.11
<i>HYPER-Coverage</i>	34.54	0.66	1.27	0.08
<i>HYPER-Perturb</i>	31.02	0.71	1.27	0.09
<i>HYPER</i>	30.87	0.72	1.26	0.09

TABLE III: Prediction performance over 3 seconds compared to baseline models. Our model balances between accuracy and coverage, with a variant (HYPER-Coverage) trained on the coverage task achieving the lowest minADE and MR.

Model	1 Second / 3 Seconds (5 samples)			
	minADE	minFDE	br-minFDE	minDER(%)
M-LSTM[4]	0.41/1.06	0.52/1.94	1.23/2.65	5.70/11.01
<i>HYPER</i>	0.32/0.80	0.40/1.47	1.05/2.12	5.18/7.65

TABLE IV: Compared to ManeuverLSTM, HYPER achieves better results in both discrete and continuous error metrics.

HYPER-Coverage, that is solely trained towards optimizing this metric, but sacrifices accuracy measured by the negative log-likelihood metric (NLL) [50]. Our method, on the other hand, balances between accuracy and coverage.

The work that is closest to ours in spirit is **ManeuverLSTM** (M-LSTM) [4]. It 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 model, training process, and definition of maneuvers as used in our model, except forcing each sample to have a fixed mode over the prediction horizons for ManeuverLSTM. We use five samples for a fair 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 IV 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.

D. 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 fail to provide sufficient coverage. On the other hand, NMS (b) selects more diverse samples, but suffers from a fixed distant

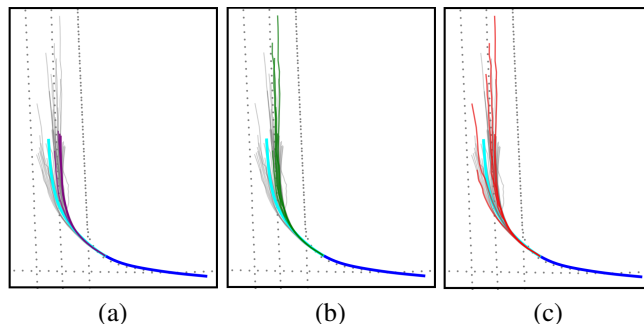


Fig. 3: Sample selection using different methods: (a) most-likely (purple), (b) NMS (2m) (green), (c) FPS (red). Ground truth past and future trajectory are in blue and cyan. Predicted samples in grey. FPS achieves the best coverage.

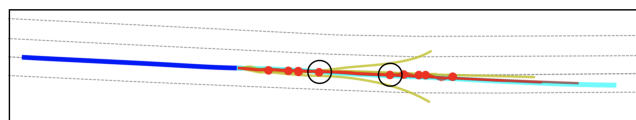


Fig. 4: Comparison between HYPER (red) and ManeuverLSTM (olive) predictions in a lane change scenario, where ground truth future trajectories are in cyan, and red dots depict where the mode changes in HYPER predictions. HYPER identifies multiple time slots, highlighted in black circles, to change the intent.

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

In Fig. 4, we show a representative example that demonstrates the advantage of supporting evolving driving modes. In this lane change scenario, the ground truth future trajectory follows the lane for a few seconds and then performs a lane change. Our model (predictions in red) infers the mode change successfully and predicts a few options on when to change, as highlighted by the red dots circled in black. On the other hand, ManeuverLSTM (predictions in olive) only predicts a single option for each maneuver, ignoring the fact that modes may change in the future sequence. This leads to worse accuracy and coverage.

VI. CONCLUSION

In conclusion, we present a general and expressive hybrid prediction model that accounts for evolving discrete modes in the future trajectory. The model leverages learned proposal functions and the farthest point sampling algorithm 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

- [1] O. Makansi, E. Ilg, O. Cicek, and T. Brox, "Overcoming limitations of mixture density networks: A sampling and fitting framework for multimodal future prediction," in *CVPR*, 2019, pp. 7144–7153.
- [2] L. A. Thiede and P. P. Brahma, "Analyzing the Variety Loss in the Context of Probabilistic Trajectory Prediction," in *ICCV*, 2019.
- [3] X. Huang, S. G. McGill, J. A. DeCastro, L. Fletcher, J. J. Leonard, B. C. Williams, and G. Rosman, "DiversityGAN: Diversity-aware vehicle motion prediction via latent semantic sampling," *IEEE Robotics and Automation Letters*, vol. 5, no. 4, pp. 5089–5096, 2020.
- [4] N. Deo and M. M. Trivedi, "Multi-modal trajectory prediction of surrounding vehicles with maneuver based LSTMs," in *IVS*, 2018, pp. 1179–1184.
- [5] J. Guan, Y. Yuan, K. M. Kitani, and N. Rhinehart, "Generative hybrid representations for activity forecasting with no-regret learning," in *CVPR*, 2020, pp. 173–182.
- [6] H. Zhao, J. Gao, T. Lan, C. Sun, B. Sapp, B. Varadarajan, Y. Shen, Y. Shen, Y. Chai, C. Schmid *et al.*, "TNT: Target-driven trajectory prediction," in *CoRL*, 2020.
- [7] L. Zhang, P.-H. Su, J. Hoang, G. C. Haynes, and M. Marchetti-Bowick, "Map-adaptive goal-based trajectory prediction," in *CoRL*, 2020.
- [8] T. Gilles, S. Sabatini, D. Tsishkou, B. Stanciulescu, and F. Moutarde, "HOME: Heatmap output for future motion estimation," *arXiv preprint arXiv:2105.10968*, 2021.
- [9] M. W. Hofbaur and B. C. Williams, "Hybrid estimation of complex systems," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 34, no. 5, pp. 2178–2191, 2004.
- [10] H. A. Blom and Y. Bar-Shalom, "The interacting multiple model algorithm for systems with Markovian switching coefficients," *TACON*, vol. 33, no. 8, pp. 780–783, 1988.
- [11] P. Andersson, "Adaptive forgetting in recursive identification through multiple models," *International Journal of Control*, vol. 42, no. 5, pp. 1175–1193, 1985.
- [12] L. Blackmore, S. Funiak, and B. C. Williams, "A combined stochastic and greedy hybrid estimation capability for concurrent hybrid models with autonomous mode transitions," *Robotics and Autonomous Systems*, vol. 56, no. 2, pp. 105–129, 2008.
- [13] D. Koller, U. Lerner, and D. Anguelov, "A general algorithm for approximate inference and its application to hybrid Bayes nets," in *UAI*, 1999.
- [14] S. K. Jayaraman, L. P. Robert, X. J. Yang, and D. M. Tilbury, "Multimodal hybrid pedestrian: A hybrid automaton model of urban pedestrian behavior for automated driving applications," *IEEE Access*, vol. 9, pp. 27 708–27 722, 2021.
- [15] P. Kothari, B. Siffringer, and A. Alahi, "Interpretable social anchors for human trajectory forecasting in crowds," in *CVPR*, 2021.
- [16] S. Gu, Z. Ghahramani, and R. E. Turner, "Neural adaptive sequential Monte Carlo," in *NeurIPS*, 2015, pp. 2629–2637.
- [17] A. Gupta, J. Johnson, L. Fei-Fei, S. Savarese, and A. Alahi, "Social GAN: Socially acceptable trajectories with generative adversarial networks," in *CVPR*, 2018, pp. 2255–2264.
- [18] N. Lee, W. Choi, P. Vernaza, C. B. Choy, P. H. Torr, and M. Chandraker, "DESIRE: Distant future prediction in dynamic scenes with interacting agents," in *CVPR*, 2017, pp. 336–345.
- [19] T. Salzmann, B. Ivanovic, P. Chakravarty, and M. Pavone, "Trajectron++: Multi-agent generative trajectory forecasting with heterogeneous data for control," *ECCV*, 2020.
- [20] T. Phan-Minh, E. C. Grigore, F. A. Boulton, O. Beijbom, and E. M. Wolff, "CovertNet: Multimodal behavior prediction using trajectory sets," in *CVPR*, 2020, pp. 14 074–14 083.
- [21] Y. Yuan and K. M. Kitani, "Diverse trajectory forecasting with determinantal point processes," in *ICLR*, 2020.
- [22] D. Richardos, B. Anastasia, D. Georgios, and A. Angelos, "Vehicle maneuver-based long-term trajectory prediction at intersection crossings," in *CAVS*. IEEE, 2020, pp. 1–6.
- [23] M. Hasan, E. Paschalidis, A. Solernou, H. Wang, G. Markkula, and R. Romano, "Maneuver-based anchor trajectory hypotheses at roundabouts," *arXiv preprint arXiv:2104.11180*, 2021.
- [24] K. Mangalam, H. Girase, S. Agarwal, K.-H. Lee, E. Adeli, J. Malik, and A. Gaidon, "It is not the journey but the destination: Endpoint conditioned trajectory prediction," in *ECCV*. Springer, 2020, pp. 759–776.
- [25] Y. Liu, J. Zhang, L. Fang, Q. Jiang, and B. Zhou, "Multimodal motion prediction with stacked transformers," in *CVPR*, 2021.
- [26] B. Kim, S. H. Park, S. Lee, E. Khoshimjonov, D. Kum, J. Kim, J. S. Kim, and J. W. Choi, "LaPred: Lane-aware prediction of multi-modal future trajectories of dynamic agents," in *CVPR*, 2021.
- [27] H. Song, D. Luan, W. Ding, M. Y. Wang, and Q. Chen, "Learning to predict vehicle trajectories with model-based planning," *arXiv preprint arXiv:2103.04027*, 2021.
- [28] L. Blackmore, S. Gil, S. Chung, and B. Williams, "Model learning for switching linear systems with autonomous mode transitions," in *CDC*. IEEE, 2007, pp. 4648–4655.
- [29] E. M. Timmons and B. C. Williams, "Best-first enumeration based on bounding conflicts, and its application to large-scale hybrid estimation," *JAIR*, vol. 67, pp. 1–34, 2020.
- [30] P. Becker-Ehmck, J. Peters, and P. Van Der Smagt, "Switching linear dynamics for variational bayes filtering," in *ICML*. PMLR, 2019, pp. 553–562.
- [31] M. W. Hofbaur and B. C. Williams, "Mode estimation of probabilistic hybrid systems," in *International Workshop on Hybrid Systems: Computation and Control*. Springer, 2002, pp. 253–266.
- [32] S. Linderman, M. Johnson, A. Miller, R. Adams, D. Blei, and L. Paninski, "Bayesian learning and inference in recurrent switching linear dynamical systems," in *Artificial Intelligence and Statistics*. PMLR, 2017, pp. 914–922.
- [33] A. Alahi, K. Goel, V. Ramanathan, A. Robicquet, L. Fei-Fei, and S. Savarese, "Social LSTM: Human trajectory prediction in crowded spaces," in *CVPR*, 2016, pp. 961–971.
- [34] S. H. Park, B. Kim, C. M. Kang, C. C. Chung, and J. W. Choi, "Sequence-to-sequence prediction of vehicle trajectory via LSTM encoder-decoder architecture," in *2018 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2018, pp. 1672–1678.
- [35] P. Del Moral, A. Doucet, and A. Jasra, "An adaptive sequential Monte Carlo method for approximate Bayesian computation," *Statistics and computing*, vol. 22, no. 5, pp. 1009–1020, 2012.
- [36] C. Naesseth, S. Linderman, R. Ranganath, and D. Blei, "Variational sequential monte carlo," in *AISTATS*. PMLR, 2018, pp. 968–977.
- [37] Y. Xu, J. Choi, S. Dass, and T. Maiti, "Sequential bayesian prediction and adaptive sampling algorithms for mobile sensor networks," *TACON*, vol. 57, no. 8, pp. 2078–2084, 2011.
- [38] R. Bardenet, A. Doucet, and C. Holmes, "Towards scaling up Markov chain Monte Carlo: an adaptive subsampling approach," in *ICML*. PMLR, 2014, pp. 405–413.
- [39] N.-C. Xiao, M. J. Zuo, and C. Zhou, "A new adaptive sequential sampling method to construct surrogate models for efficient reliability analysis," *Reliability Engineering & System Safety*, vol. 169, pp. 330–338, 2018.
- [40] A. Wang, X. Huang, A. Jasour, and B. Williams, "Fast risk assessment for autonomous vehicles using learned models of agent futures," in *RSS*, 2020.
- [41] R. Krishnan, U. Shalit, and D. Sontag, "Structured inference networks for nonlinear state space models," in *AAAI*, vol. 31, no. 1, 2017.
- [42] J. Gao, C. Sun, H. Zhao, Y. Shen, D. Anguelov, C. Li, and C. Schmid, "VectorNet: Encoding HD maps and agent dynamics from vectorized representation," in *CVPR*, 2020, pp. 11 525–11 533.
- [43] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Advances in neural information processing systems*, 2014, pp. 3104–3112.
- [44] R. Zhao, X. Sun, and V. Tresp, "Maximum entropy-regularized multi-goal reinforcement learning," in *ICML*. PMLR, 2019, pp. 7553–7562.
- [45] T. F. Gonzalez, "Clustering to minimize the maximum intercluster distance," *Theoretical computer science*, vol. 38, pp. 293–306, 1985.
- [46] M.-F. Chang, J. Lambert, P. Sangkloy, J. Singh, S. Bak, A. Hartnett, D. Wang, P. Carr, S. Lucey, D. Ramanan *et al.*, "Argoverse: 3D tracking and forecasting with rich maps," in *CVPR*, 2019, pp. 8748–8757.
- [47] E. Jang, S. Gu, and B. Poole, "Categorical reparameterization with Gumbel-Softmax," *arXiv preprint arXiv:1611.01144*, 2016.
- [48] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *ICLR*, 2015.
- [49] Y. Chai, B. Sapp, M. Bansal, and D. Anguelov, "Multipath: Multiple probabilistic anchor trajectory hypotheses for behavior prediction," in *CoRL*, 2019.
- [50] B. Ivanovic and M. Pavone, "The trajectron: Probabilistic multi-agent trajectory modeling with dynamic spatiotemporal graphs," in *ICCV*, 2019, pp. 2375–2384.