Heterogeneous-Agent Trajectory Forecasting Incorporating Class Uncertainty

Boris Ivanovic[1] Kuan-Hui Lee[2] Pavel Tokmakov[2] Blake Wulfe[2] Rowan McAllister[2]
Adrien Gaidon[2] Marco Pavone[1]
[1]Stanford University [2]Toyota Research Institute
{borisi, pavone}@stanford.edu
{kuan.lee, pavel.tokmakov, blake.wulfe, rowan.mcallister, adrien.gaidon}@tri.global

Abstract

Reasoning about the future behavior of other agents is critical to safe robot navigation. The multiplicity of plausible futures is further amplified by the uncertainty inherent to agent state estimation from data, including positions, velocities, and semantic class. Forecasting methods, however, typically neglect class uncertainty, conditioning instead only on the agent's most likely class, even though perception models often return full class distributions. To exploit this information, we present HAICU*, a method for heterogeneous-agent trajectory forecasting that explicitly incorporates agents' class probabilities. We additionally present* PUP*, a new challenging real-world autonomous driving dataset, to investigate the impact of Perceptual Uncertainty in Prediction. It contains challenging crowded scenes with unfiltered agent class probabilities that reflect the long-tail of current state-of-the-art perception systems. We demonstrate that incorporating class probabilities in trajectory forecasting significantly improves performance in the face of uncertainty, and enables new forecasting capabilities such as counterfactual predictions.*

1. Introduction

Incorporating perceptual uncertainty into downstream components, such as forecasting and planning, is critical for the safe operation of autonomous vehicles. Failing to do so has already led to fatalities partially caused by perceptual errors propagated from vision [1] and LiDAR [7]. Currently, most trajectory forecasting methods do not explicitly incorporate or propagate perceptual uncertainties from their inputs to their outputs [49]. Instead, they classify an agent with its highest probability class from the upstream perception system. However, blindly trusting the most-likely class can have disastrous consequences, especially in the presence of class uncertainty, as shown in Figure 1. Misclassifications like these could cause an autonomous vehicle to make unnecessary evasive maneuvers and may occur frequently in challenging real-world conditions (Section 5). A safer way to deal with perceptual uncertainties is to propagate them through forecasting systems, so that planning components can make uncertainty-aware decisions [39, 5].

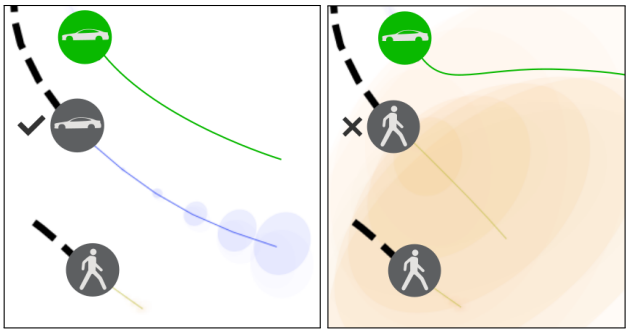

Figure 1. Classifying agents with their most-likely class can be disastrous in the presence of uncertainty. A state-of-the-art trajectory forecasting model, Trajectron++ [51], produces well-behaved predictions when agents are correctly classified (**left**; as a car). However, its uncertainty grows sharply when the agent is misclassified as a pedestrian (**right**). This can cause a sudden and unexpected change in the ego-vehicle's resulting motion plan, in green.

Contributions. Our key contributions are twofold. First, we present **HAICU**, a method for Heterogeneous-Agent trajectory forecasting Incorporating Class Uncertainty. We show how to effectively incorporate class probabilities from upstream perception systems into state-of-the-art trajectory forecasting methods, yielding better performance in the presence of uncertainty. We also demonstrate how this enables fine-grained introspection via counterfactual predictions by intervening on class probabilities directly to produce "what-if" predictions. Second, we present the Perceptual Uncertainty in Prediction (**PUP**) dataset[1]: a new, challenging, real-world autonomous driving dataset with complex scenes and significant agent class uncertainties. Our dataset better reflects the challenges of the long tail of current state-of-the-art perception systems, thus enabling others to more effectively study robustness to class uncertainty

[1]All of our code, models, and data will soon be available online.

in trajectory forecasting. Our experiments on the Lyft Level 5 [23] and PUP datasets show that our approach indeed significantly improvements upon the state-of-the-art, thanks to the inclusion of class probabilities.

2. Related Work

Modular Trajectory Forecasting. Modular methods decompose autonomous driving into distinct sub-tasks, usually perception, prediction, planning and control [54]. This "divide-and-conquer" strategy allows the complex problem of driving to be divided into more manageable subproblems that can each be solved in isolation once the interface between them is designed. A typical interface between perception and forecasting is to communicate only the most likely class and state estimate of each object detected by a perception system. Trajectory forecasting methods have thus traditionally assumed their inputs are known with certainty [35]. In reality, sensors are imperfect, and incorrect assumptions of certainty-equivalence in perception—where only the most likely class estimate is passed from perception to prediction but not its uncertainty—has been partially responsible for two separate fatalities [1, 7].

To our knowledge, prior forecasting work has not yet considered the propagation of class uncertainties through modular systems, but there have been many developments. For instance, as forecasting is an inherently multi-modal task (especially at intersections), several recent works have proposed multi-modal probabilistic models, trained using exact-likelihood [45, 13] or variational inference [52, 27, 26, 51]. Generative Adversarial Networks (GANs) [19] can generate empirical trajectory distributions from sampling multiple predictions [21, 48]. However, analytic distributions are often useful for gradient-based planning that minimize the likelihood of collisions [54]. Thus, we focus on methods that predict analytic trajectory distributions.

A straightforward, data-driven way to forecast multi-modal trajectories is by selecting a representative set sampled from human drivers, but specific to the current context [43], and possibly allowing for deviations too [13]. While a set of predefined lookup trajectories is useful for rapid single-agent prediction or planning, the number of trajectories required for multi-agent settings scales exponentially with the number of agents.

Attention-based models are better suited to multi-agent forecasting, especially when only a subset of agent-agent interactions are important [40]. They also naturally handle a variable number of agents as the road scene evolves [56]. To our knowledge, attention-based models have not yet been applied to heterogeneous forecasting. For more longer predictions (>3s), inferring plausible goal locations helps incorporate more historical data and yield more consistent predictions over time of pedestrians, [61, 31] ado-vehicles [3], or ego-vehicles [46, 58].

Some methods assume that heterogeneous traffic-agent prediction classes are known with certainty [37, 38]. While some prior work recognizes that errors in predicting classes can differ between pedestrians and vehicles at training time [44], to our knowledge, no work communicates these class uncertainties at test-time so the planning component can make safe decisions in real-time. We therefore propose to incorporate class-uncertainties into prediction, to enable safer planning downstream. By incorporating uncertainty into a *modular* framework, the effects of class uncertainty are transparent, and give an introspective ability w.r.t. counterfactual predictions, as seen in Figure 1.

End-to-End Prediction. End-to-end prediction methods perform detection, tracking, and prediction jointly, and operate directly on LiDAR sensor input. FaF [36] introduced the approach of projecting LiDAR points into a bird's eye view (BEV) grid, and generating predictions through inferred detections multiple timesteps in the future. This approach was extended by IntentNet [12], which incorporated HD map information as an input, and predicted agent intent as well. SpAGNN [10] modeled agent interactions using a graph network, and ILVM [11] extended this direction further by modeling the joint distribution over future trajectories using a latent variable model. These methods only consider homogeneous agents (vehicles); however, MultiXNet [17] recently extended the BEV approach to heterogeneous agents by adding separate outputs for each object type. While this approach accounts for perception uncertainty over the object type, the resulting number of predicted trajectories scales with the number of object types or requires a hard selection of the object type to be considered in the planner. Broadly, end-to-end methods only incorporate class probabilities implicitly, making it difficult to transparently analyze, probe (e.g., via counterfactual analysis), and understand internal state changes related to perceptual uncertainty. At a higher level, end-to-end methods are often tied to a specific perception configuration, making it difficult to evaluate the quality of their planning without simulating sensors directly.

Uncertainty Propagation. Recently, there has been much attention on uncertainty estimation in deep learning [6, 33, 18], especially so in the context of planning [16, 15]. However, works in this field mainly focus on uncertainty estimation rather than propagation. In particular, a core focus is estimating a model's parameter uncertainty due to finite training data (often called epistemic uncertainty [47]), rather than how to propagate uncertainty in input values through the model to its outputs.

Some works propagate uncertainty derived from imperfect sensors, often termed aleatoric uncertainty. Work by Brechtel et al. [9] for instance considers how perceptual uncertainty from occluded regions and imperfect estimates of other agents' positions and velocities are propagated downstream to forecasting and planning for a particular merging

scenario by treating autonomous driving as a partially observed Markov decision process (POMDP), but do not consider class uncertainties. In contrast, our method incorporates class uncertainty and uses learned behavior models to apply to multiple scenarios and intersections.

3. Problem Formulation

We aim to generate plausible future trajectory distributions for a time-varying number $N(t)$ of diverse interacting agents $A_1, \ldots, A_{N(t)}$. Each agent A_i has a class C_i taking one of K values (e.g., Car, Bicycle, Pedestrian). At each time t, an upstream perception model estimates the probability that agent A_i is of class $k = 1, ..., K$, producing a vector of class probabilities constrained to the $(K-1)$-simplex $\hat{\mathbf{c}}_i^{(t)} \in \Delta^{K-1}$ for all agents, where $\hat{c}_{i,k}^{(t)} = p(C_i = k; t)$ is the perception-estimated probability that agent A_i is of class k at time t, and

$$\Delta^{(K-1)} = \left\{ \mathbf{v} \in \mathbb{R}^K \mid \sum_{k=1}^{K} v_i = 1 \text{ and } v_i \geq 0 \; \forall \; i \right\} \quad (1)$$

At time t, given the state $\mathbf{s}_i^{(t)} \in \mathbb{R}^D$ of each agent (e.g., x, y positions, velocities, and accelerations), their estimated class probabilities $\hat{\mathbf{c}}_i^{(t)}$, and their histories for the previous H timesteps, which we denote as $\mathbf{x} = \mathbf{s}_{1,\ldots,N(t)}^{(t-H:t)} \in \mathbb{R}^{(H+1)\times N(t)\times D}$ and $\hat{\mathbf{c}} = \hat{\mathbf{c}}_{1,\ldots,N(t)}^{(t-H:t)} \in \mathbb{R}^{(H+1)\times N(t)\times K}$, our goal is to produce a distribution over all agents' future states for the next T timesteps, $\mathbf{y} = \mathbf{s}_{1,\ldots,N(t)}^{(t+1:t+T)} \in \mathbb{R}^{T\times N(t)\times D}$, which we denote as $p(\mathbf{y} \mid \mathbf{x}, \hat{\mathbf{c}})$.

4. Incorporating Class Uncertainty in Trajectory Forecasting

We build upon the Trajectron++ [51] framework to implement HAICU, due to its ability to perform multi-agent, multi-class trajectory forecasting and public codebase. In this section, we summarize the core components of the algorithm and highlight our key augmentations for incorporating class uncertainty. Figure 2 visualizes HAICU's probabilistic graphical model and overall network architecture.

Input Representation. We first abstract the scene as an undirected spatiotemporal graph $G = (V, E)$, where nodes represent agents and edges represent their interactions. We use the ℓ_2 distance as a proxy for agent interaction: an undirected edge connects A_i and A_j if $\|\mathbf{p}_i - \mathbf{p}_j\|_2 \leq d$ where $\mathbf{p}_i, \mathbf{p}_j \in \mathbb{R}^2$ are the 2D positions of agents A_i, A_j, respectively, and d is a chosen distance threshold. This differs from methods that use a directed graph structure (e.g., Trajectron++), the creation of which relies on hard agent classes to determine edge type and direction.

Encoding Agent History. With this graph representation in hand, our model focuses on encoding a node's state history and how it is influenced by its neighbors. To encode an agent's observed trajectory history, its current and previous states $\mathbf{s}_{1,\ldots,N(t)}^{(t-H:t)} \in \mathbb{R}^{(H+1)\times N(t)\times D}$ are fed into a Long Short-Term Memory (LSTM) network [22] with 32 hidden dimensions. Since we are interested in modeling trajectories, the states $\mathbf{s}_i^{(t)}$ are positions, velocities, and accelerations, which are easily estimated online.

Modeling Agent Interactions. To model neighboring agents' influence on the modeled agent, edge features from neighboring agents are aggregated via an element-wise sum. We choose to combine features in this way rather than with averaging or concatenation to handle a variable number of neighboring nodes with a fixed architecture while preserving count information [4, 27, 28, 26, 51]. These aggregated states are then fed into an LSTM with 8 hidden dimensions, yielding a single influence representation vector encoding the effect that all neighboring nodes have. The node history and edge influence encodings are then concatenated to produce a single node representation vector, $e_{\mathbf{x}}$.

Accounting for Multimodality. Our model leverages the Conditional Variational Autoencoder (CVAE) latent variable framework [55] to explicitly account for high-level multimodality in behavior. It produces the target $p(\mathbf{y} \mid \mathbf{x}, \hat{\mathbf{c}})$ distribution by introducing a discrete Categorical latent variable $z \in Z$ which encodes high-level latent behavior and allows for the prior distribution $p(\mathbf{y} \mid \mathbf{x}, \hat{\mathbf{c}})$ to be expressed as

$$p(\mathbf{y} \mid \mathbf{x}, \hat{\mathbf{c}}) = \sum_{z \in Z} p_\psi(\mathbf{y} \mid \mathbf{x}, z, \hat{\mathbf{c}}) p_\theta(z \mid \mathbf{x}, \hat{\mathbf{c}}), \quad (2)$$

where $|Z| = 25$ and ψ, θ are network weights. We chose $|Z|$ as such because it allows for the modeling of a wide variety of high-level latent behaviors and any unused latent classes will be ignored by the CVAE [24].

During training, a bi-directional LSTM with 32 hidden dimensions is used to encode a node's ground truth future trajectory, producing $q_\phi(z \mid \mathbf{x}, \mathbf{y})$ [55].

Generating Trajectories. The latent variable z and node representation vector $e_{\mathbf{x}}$ are then fed into the decoder, a 128-dimensional Gated Recurrent Unit (GRU) [14]. Each GRU cell outputs the parameters of a bivariate Gaussian distribution over control actions $\mathbf{u}^{(t)}$ (e.g., velocity). The agent's system dynamics are then integrated with $\mathbf{u}^{(t)}$ to obtain trajectories in position space [30, 57]. Since the only uncertainty at prediction time stems from HAICU's output, and we model agents with linear dynamics, i.e., single integrators, the resulting system dynamics are linear Gaussian. We model all agents as single integrators because we do not know their classes *a priori*. The single integrator model has no constraints, allowing for all possible agent movement. By comparison, e.g., the dynamically-extended unicycle [34] posits that agents are subject to non-holonomic

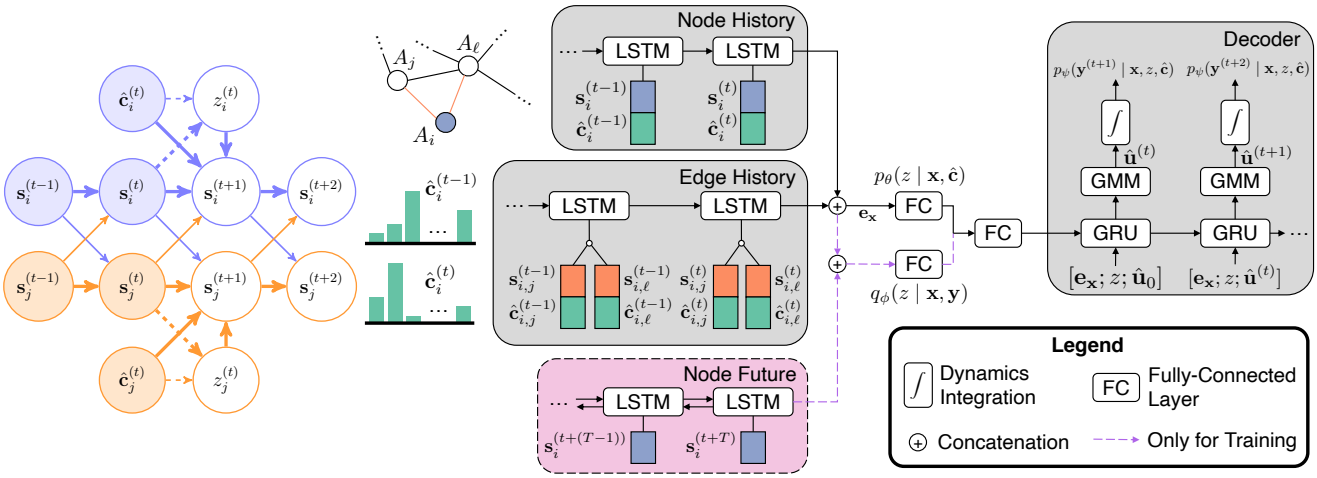

Figure 2. **Left**: Our inference-time probabilistic graphical model for forecasting taking into account agent interactions and class probabilities over time, illustrated for two agents (blue and orange). We use shading to denote known values and thick-arrow notation for carry-forward dependencies [46]. Perception provides class-probabilities for each agent $\hat{\mathbf{c}}$, and the known (shaded) past states of both agents help us infer (not generate—denoted by dashed lines) its intentions z at the current time t, which help predict the future states of agents. **Right**: Our approach represents a scene as an undirected spatiotemporal graph. Nodes and edges represent agents and their interactions, respectively. Our method incorporates agent class uncertainty by encoding class probability values alongside the agent's state.

constraints [41], over-constraining pedestrians.

Using one agent dynamics model simplifies HAICU's construction as all agents can use the same overall architecture. A different route is to include various agent dynamics, and one way of doing this in HAICU is to make the decoder multi-headed (i.e., using a different decoder per dynamics model). Such architectures are very popular in the literature, and we explore this avenue in Section 6.

Incorporating Class Uncertainty. To incorporate class probabilities in our model, we concatenate the input class probability vector $\hat{\mathbf{c}}_i^{(t)}$ with the state $\mathbf{s}_i^{(t)}$ and encode the resulting $(D + K)$-dimensional vector in the same way as the original state vector, with the node and edge history encoders. Neighboring agent class probability vectors are similarly aggregated in the edge encoder, after which the result is clamped to have maximum value 1.

Training the Model. We adopt the same discrete Info-VAE [59] objective function as in Trajectron++. Formally, we aim to solve

$$\begin{aligned}\max_{\phi,\theta,\psi} \sum_{i=1}^{N} \; & \mathbb{E}_{z\sim q_\phi(\cdot|\mathbf{x}_i,\mathbf{y}_i)}\big[\log p_\psi(\mathbf{y}_i \mid \mathbf{x}_i, z, \hat{\mathbf{c}}_i)\big] \\ & - \beta D_{KL}\big(q_\phi(z \mid \mathbf{x}_i, \mathbf{y}_i) \parallel p_\theta(z \mid \mathbf{x}_i, \hat{\mathbf{c}}_i)\big) \\ & + I_q(\mathbf{x}_i; z),\end{aligned} \tag{3}$$

where ϕ, θ, ψ are network weights and I_q is the mutual information between $\mathbf{x}_i$ and z under the distribution $q_\phi(\mathbf{x}_i, z)$. To compute I_q, we approximate $q_\phi(z \mid \mathbf{x}_i, \mathbf{y}_i)$ with $p_\theta(z \mid \mathbf{x}_i)$ and obtain the unconditioned latent distribution by summing out $\mathbf{x}_i$ over the batch [59]. Note that the Gumbel-Softmax reparameterization [29] is *not* used to backpropagate through the Categorical latent variable z because z is not sampled during training. Instead, the first term of Equation (3) is directly enumerated and summed since the latent space has only $|Z| = 25$ discrete elements.

Class	PUP (Ours)		Lyft Level 5 [23]	
	Num. (%)	S_{probs}	Num. (%)	S_{probs}
bicycle	1.2k (0.8)	1.60	0.1M (0.4)	0.09
car	117k (81.9)	1.10	5.0M (24.5)	0.00
largevehicle	18k (12.4)	1.30	–	–
motorcycle	0.5k (0.3)	1.57	–	–
pedestrian	6.3k (4.4)	1.44	0.7M (3.3)	0.01
unknown	0.2k (0.1)	0.05	14.6M (71.8)	0.00

Table 1. Our PUP dataset has both a wider variety of classes and more uncertainty (i.e., class probability entropy, S_{probs}) per agent class than the Lyft Level 5 dataset [23].

5. Datasets

We evaluate HAICU on two real-world autonomous driving datasets described in the following sections: Lyft Level 5 [23] and PUP, a new dataset that we are releasing with this work. Table 1 contains detailed statistics for both datasets and Figure 3 depicts a few scenes from PUP.

5.1. Lyft Level 5 Dataset

The Lyft Level 5 dataset is comprised of 1,118 hours of data collected in Palo Alto, USA. Each scene is annotated at 10 Hz ($\Delta t = 0.1s$) and is 25s long, containing 4 agent classes. Importantly, the Lyft dataset was the first to release class probabilities for each detected agent.

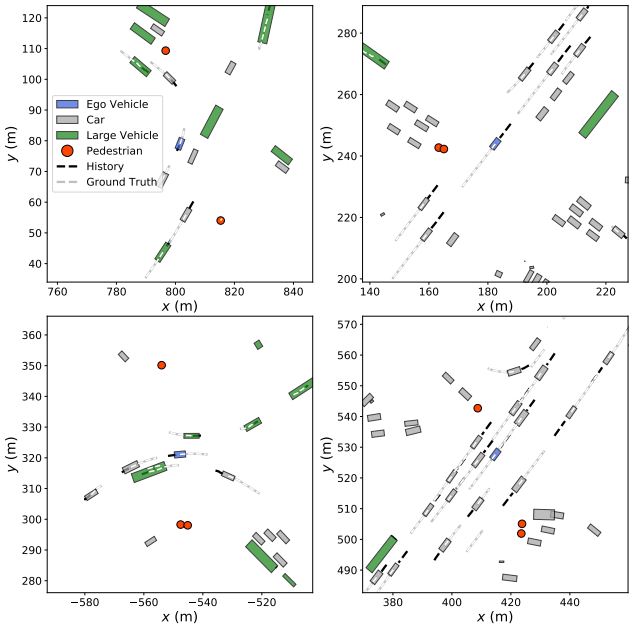

Figure 3. Example scenes from our new PUP dataset.

Prevalence of Class Switching. A key motivation of this work is building trajectory forecasting methods that are robust to perceptual classification errors and uncertainty. In the Lyft Level 5 dataset, we find that 2.1% of all agents experience class-switching, i.e., their highest probability class changes during observation. While the most common switches are between "unknown" and known classes, there are thousands of known class-to-class switches. For example, Appendix A.1 visualizes a scenario where a nearby car is misclassified as a pedestrian in the middle of an intersection. Another example in Appendix A.1 shows a pedestrian adjacent to the ego-vehicle being misclassified as a car while waiting to cross the street, demonstrating that class switching is not solely due to an agent being very far away from sensor view.

Class Switches are Long-lived. These misclassifications are not solely short-lived, temporary switches, either. We initially applied a temporal smoothing strategy, majority voting with a 5-timestep (0.5s) window, in an attempt to remove cases of high-frequency class switching, but very few switching instances were corrected (only 1–3%). Further, applying explicit temporal smoothing brings additional tradeoffs regarding accuracy and latency, both of which are especially important for an online task such as object detection and tracking in autonomous driving, but ultimately out of scope for this work.

Overconfidence. Even in the presence of class switches, we find the Lyft dataset's class probabilities to be overconfident, i.e., nearly always one-hot vectors, a common issue in deep learning [20]. Table 1 (right) shows the overall average entropy S_{probs} of each agent's class probabilities, $S_{\text{probs}} = -\sum_k P(C_i = k) \log P(C_i = k)$, where $S_{\text{probs}} = 0.00$ is a distribution with one class having probability 1.0. This certainty is also present over time, Appendix A.2 shows that the most-likely class has more than 90% probability on average per agent timestep.

Motivation for PUP. As discussed, the Lyft Level 5 dataset provides a vast amount of data in the regime where perception systems are very certain of their outputs. We argue, however, that perception systems will not always be so certain, and wish to investigate the benefits of incorporating such information in trajectory forecasting. Since we could not find any existing datasets that provide data in this uncertain regime, we present our own.

5.2. PUP Dataset

To provide more options for studying the effects of perceptual uncertainty in downstream tasks, one of our core contributions is the release of a novel real-world autonomous driving dataset comprised of 1,637 distinct scenes collected with a state-of-the-art self-driving fleet in a major urban city center. Each scene is 10s long and annotated at 10 Hz with ground-truth trajectories and classes. There are 11 unique agent classes (6 mobile, 4 static, and "unknown"). Each agent has time-varying class probabilities produced by a state-of-the-art in-house perception stack. For reference, our data was collected with the following AP@0.5 values per class: 0.75 for cars, 0.49 for largevehicles, 0.80 for pedestrians, and 0.50 for both motorcycles and bicycles.

Our goal is to quantitatively evaluate the performance of trajectory forecasting in the presence of class uncertainty. To accomplish this, we select a diverse set of scenes by oversampling the long tail of perceptual challenges, whether due to epistemic (model) or aleatoric (scene) uncertainty. Furthermore, we do not perform any post-hoc filtering or smoothing of the agent class probabilities, intentionally releasing the raw frame-by-frame outputs from an in-house camera- and LiDAR-based perception module to enable a wide variety of future work (e.g., from low-latency strategies for temporal smoothing to trajectory forecasting to planning with such information).

Table 1 shows a side-by-side comparison of the PUP dataset's class composition and average class uncertainty with those of the Lyft dataset, as measured by mean class probability entropy. At a high level, our class probabilities are more uncertain, with far fewer unknown agents.

6. Experimental Protocol

Baselines. We compare HAICU against the following state-of-the-art approaches that also produce multimodal predictions for varying numbers of diverse agents:

(1) MATS [25]: each scene is modeled with a mixture of

affine dynamical systems, forward-integrated to produce predictions.

(2) Trajectron++ [51]: a state-of-the-art LSTM-CVAE encoder-decoder whose architecture is based on the spatiotemporal structure of the scene.

Note that both MATS and Trajectron++ assume perfect agent classification, and rely on such information in their network components (e.g., sharing weights among same-class components, assuming prior dynamics models for each class). We also evaluate ablations of HAICU:

(3) One-Hot: HAICU, but with one-hot class probabilities passed in. This corresponds to the hard class-conditioning of Trajectron++ (no uncertainty modeling) while using our class-agnostic weight-sharing scheme.

(4) Multi-Head: HAICU, but rather than encoding class probabilities in $e_{\mathbf{x}}$, its decoder becomes multi-headed and produces an output for each possible agent dynamics model (i.e., dynamically-extended unicycle [34] for vehicles and single integrator for others, as in [51]); these outputs are then combined in a mixture model where the class probabilities $\hat{\mathbf{c}}_i^{(t)}$ are the mixing probabilities. Concretely, this ablation evaluates the merit of considering various agent dynamics (as in [51]) while incorporating class probabilities.

Metrics. As in prior work [2, 21, 26, 50, 32, 60, 51], we evaluate trajectory forecasting methods with the following four error metrics:

1. *Average Displacement Error (ADE)*: Mean ℓ_2 distance between the ground truth and predicted trajectories.
2. *Final Displacement Error (FDE)*: ℓ_2 distance between the predicted final position and the ground truth final position at the prediction horizon T.
3. *Average Negative Log-Likelihood (ANLL)*: Mean NLL of the ground truth trajectory under the output distribution.
4. *Final Negative Log-Likelihood (FNLL)*: NLL of the final ground truth position under the output distribution at the prediction horizon T.

For ADE and FDE we compare methods' single most-likely trajectory prediction (establishing accuracy for the deterministic use case), while for ANLL and FNLL we compute likelihoods using their full output distributions (determining performance for probabilistic use cases).

Evaluation Methodology. For both datasets, we use $70\%, 15\%, 15\%$ data (scene) splits for training, validation, and testing, respectively. Models are trained to predict forward 20 timesteps (2s) from at most 20 timesteps (2s) of observed data.

When collecting statistics or evaluating methods that require agent classes (e.g., Trajectron++ and MATS) on the PUP dataset, we use an agent's most often most-likely class as their fixed classification.

We trained each model until their validation performance stopped improving. Further training details, hyperparameters, and data augmentation information can be found in Appendix D. All methods were implemented in PyTorch [42] on a computer running Ubuntu 18.04 containing an AMD Ryzen 1800X CPU and two NVIDIA GTX 1080 Ti GPUs.

7. Experimental Results

7.1. Lyft Dataset Results

Table 2 summarizes our evaluation on the Lyft dataset, and shows that HAICU outperforms state-of-the-art trajectory forecasting methods on the probabilistic ANLL and FNLL metrics, and is competitive on the deterministic ADE and FDE metrics. Notably, even though the Lyft dataset does not have much class uncertainty (Table 1), our method and its ablations still outperform MATS and Trajectron++, which architecturally incorporate agent class information, indicating that a class-agnostic modeling scheme yields improvements. This is also the reason why our method with one-hot class probabilities performs similarly to our method with full probability input, as the Lyft dataset is already mostly comprised of one-hot class probability vectors.

Further, encoding class probabilities with the state input outperforms a multi-headed output mixture. This is likely due to the multi-headed version of our model being an interpolation between Trajectron++ (with separate encoder and decoder components per class) and HAICU (same encoder and decoder components for all classes), using the same encoder for all classes but different decoder per class.

Finally, while all models were trained with a prediction horizon of $2s$, we also evaluate their performance on a $3s$ prediction horizon as an additional test of temporal generalization. As can be seen in Table 2, HAICU maintains strong performance at longer time horizons. Additional results per agent class can be found in Appendix C.1.

7.2. PUP Dataset Results

Quantitative Results. Table 3 summarizes the evaluation on the PUP dataset, and shows that HAICU significantly outperforms state-of-the-art trajectory forecasting methods on both deterministic (two-tailed t-test; $P < 0.025$) and probabilistic (two-tailed t-test; $P < 10^{-10}$) metrics across all prediction horizons.

Notably, in the presence of increased class uncertainty our method now outperforms the one-hot ablated version,

Metric	ADE (m)			FDE (m)			ANLL (nats)			FNLL (nats)		
Pred. Horizon	1s	2s	3s	1s	2s	3s	1s	2s	3s	1s	2s	3s
MATS [25]	0.24	0.60	1.21	0.34	1.22	2.90	−0.52	0.74	3.78	0.25	2.47	12.06
Trajectron++ [51]	0.14	0.29	0.47	0.26	0.60	1.05	−2.88	−1.75	−1.06	−1.34	−0.13	0.68
HAICU (Multi-Head)	0.14	0.28	0.45	0.26	0.58	0.99	−2.92	−1.82	−1.15	−1.44	−0.25	0.54
HAICU (One-Hot)	0.14	0.27	0.44	0.25	0.56	0.95	−3.12	−2.01	**−1.36**	−1.56	**−0.47**	**0.27**
HAICU	**0.13**	**0.26**	**0.43**	**0.24**	**0.54**	**0.94**	**−3.14**	**−2.02**	−1.35	**−1.57**	−0.45	0.31

Table 2. Our model outperforms state-of-the-art heterogeneous-agent methods on the deterministic ADE and FDE metrics as well as the probabilistic ANLL and FNLL metrics on the Lyft Level 5 dataset [23]. It is expected that our method performs similarly with an ablation that only takes in one-hot class probabilities because the Lyft data mostly contains one-hot class probabilities. Lower is better, bold is best.

Metric	ADE (m)			FDE (m)			ANLL (nats)			FNLL (nats)		
Pred. Horizon	1s	2s	3s	1s	2s	3s	1s	2s	3s	1s	2s	3s
MATS [25]	0.48	0.78	1.23	0.48	1.08	2.12	2.23	2.91	5.66	2.23	3.60	11.15
Trajectron++ [51]	0.30	0.51	0.75	0.48	0.93	1.52	−1.21	−0.44	0.04	−0.30	0.69	1.32
HAICU (Multi-Head)	0.35	0.59	0.84	0.55	1.05	1.64	−1.17	−0.38	0.18	−0.28	0.86	1.57
HAICU (One-Hot)	0.27	0.46	0.69	0.42	0.85	1.41	−1.54	−0.76	−0.23	−0.59	0.44	1.08
HAICU	**0.26**	**0.43**	**0.65**	**0.40**	**0.79**	**1.35**	**−1.60**	**−0.86**	**−0.35**	**−0.77**	**0.32**	**0.96**

Table 3. Our model outperforms state-of-the-art heterogeneous-agent methods on the deterministic ADE and FDE metrics as well as the probabilistic ANLL and FNLL metrics on our new PUP dataset. Lower is better, bold is best.

significantly so on ANLL (two-tailed t-test; $P < 0.02$), verifying that our model is able to effectively use the full input probability information. Unlike the Lyft dataset, the multi-headed version of our model does not perform as well. We believe this is a direct result of the increased class uncertainty as the decoder heads are trained together in an overall mixture model. As a result, class uncertainty directly competes with class-based dynamics (the output heads are simultaneously trying to model the same target ground truth trajectory), leading to a reduction in overall output diversity. This motivates future work in determining an effective strategy for incorporating agent dynamics in the presence of class uncertainty.

Further, as in Section 7.1, all models used a prediction horizon of $2s$ during training and we also evaluate their performance on a longer, $3s$ prediction horizon. As can be seen in Table 3, our approach maintains its strong performance over longer time horizons. Additional detailed results per agent class can be found in Appendix C.2.

Qualitative Results. Incorporating class uncertainty in trajectory forecasting yields qualitative differences in output behavior. Figure 4 visualizes a scene from the PUP dataset with two vehicles moving parallel to each other, whose future trajectories are forecasted by Trajectron++, our one-hot ablated model, and our model. In this example, we see that Trajectron++ makes overconfident predictions which overshoot the marked agent's ground truth future trajectory. Our one-hot ablated model version similarly overshoots the ground truth, although with a bit more uncertainty on the left side of the ground truth. In comparison, our method not only significantly more accurately predicts the ground truth, it also produces equal amounts of uncertainty on either side of the ground truth. Specifically, incorporating full probabilities with our approach in this example yielded an ADE and FDE that are 1.9 m and 3.9 m less, respectively, than both Trajectron++ and our one-hot ablated model. The predicted distribution is also more accurate, with ANLL and FNLL values that are 0.24 nats and 0.83 nats less, respectively, than both Trajectron++ and our one-hot ablated model.

Counterfactual Predictions. By explicitly conditioning on class probabilities at the input, our method is additionally able to make *counterfactual* predictions, i.e., predictions where the input class probabilities are manually specified ahead of time in order to produce "what-if" predictions. In the example visualized in Figure 5, a pedestrian is walking towards the bottom-right, a stopped car is starting to move on the left, and two vehicles are driving straight towards the bottom-right. Our method's predictions with the original probabilities in the data are shown in Figure 5 (left). In particular, our model predicts that each agent will mostly keep moving along the same heading with some uncertainty. In Figure 5 (middle), we manually change the class probabilities for each agent to be a uniform distribution over all classes (representing total class uncertainty). Immediately, we see that our model produces more uncertainty, accounting for the possibility of each agent having a wide range of dynamic capabilities (e.g., turning radii, acceleration rates). Conversely, in Figure 5 (right) we made the class probabilities totally certain (one-hot for "pedestrian") for each agent. With this information, our model predicts forward motion at normal human walking speeds for all agents, as desired. This leaves the original bottom-left pedestrian prediction virtually unchanged, but greatly alters the car predictions

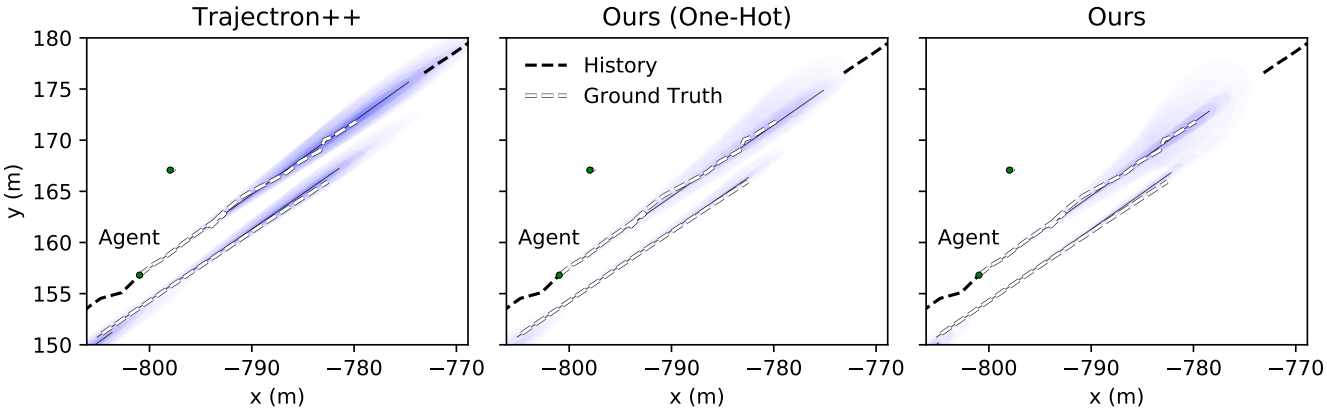

Figure 4. Our method effectively propagates agent class probability uncertainty through to its outputs. In this example, the marked agent is a vehicle with high class uncertainty (class probability entropy $S_{\text{probs}} = 2.06$, maximum possible entropy is $\ln(11) \approx 2.40$). Unlike Trajectron++ [51] (**left**) and an ablated version of our model that only takes in one-hot class probabilities (**middle**), our method is able to incorporate such information and produces much more accurate predictions (**right**).

on the right to match how a pedestrian would behave.

Prediction Smoothness. Our model's predictions interpolate smoothly in the space of input probabilities. In Figure 6, we manually vary the class probabilities of the visualized agent from its original values (high probability of being a car) to fully uncertain values (uniform probability for all classes). We can see that as class uncertainty increases, so does our model's output uncertainty. Prediction smoothness is desirable as it means that our model can smoothly propagate input uncertainty through to its outputs across a wide range of class probability values.

8. Conclusion

We investigate the problem of robustness to perceptual uncertainty in multi-agent trajectory forecasting. In particular, we highlight the importance of leveraging the full distribution over the semantic classes of agents which is typically provided by perception models, but often quantized to the (potentially-overconfident) mode. We introduce a new method (*HAICU*) for heterogeneous-agent trajectory forecasting that explicitly incorporates class probabilities, as well as a new autonomous driving dataset (*PUP*) to study the impact of Perceptual Uncertainty in Prediction, focusing on the challenging long-tail of current state-of-the-art perception systems. Thanks to its explicit incorporation and propagation of semantic uncertainty, our method outperforms the state-of-the-art in multi-class multi-agent trajectory forecasting on both the Lyft Level 5 dataset [23] and our PUP benchmark. In addition to a more informed representation of uncertainty, our approach also enables new forecasting capabilities such as counterfactual predictions, opening up interesting future research in causal reasoning and interpretability for prediction and planning.

Acknowledgment. We thank Jie Li for her input throughout the project. Toyota Research Institute ("TRI") provided funds to assist the authors with their research but this article solely reflects the opinions and conclusions of its authors and not TRI or any other Toyota entity. We also acknowledge the support of the Natural Sciences and Engineering Research Council of Canada (NSERC), funding reference number 545934-2020.

References

[1] National Highway Traffic Safety Administration. PE 16-007. https://static.nhtsa.gov/odi/inv/2016/INCLA-PE16007-7876.PDF, Jan 2017. Tesla Crash Preliminary Evaluation Report. 1, 2

[2] Alexandre Alahi, Kratarth Goel, Vignesh Ramanathan, Alexandre Robicquet, Li Fei-Fei, and Silvio Savarese. Social LSTM: Human trajectory prediction in crowded spaces. In *IEEE Conf. on Computer Vision and Pattern Recognition*, 2016. 6

[3] Stefano V. Albrecht, Cillian Brewitt, John Wilhelm, Balint Gyevnar, Francisco Eiras, Mihai Dobre, and Subramanian Ramamoorthy. Interpretable goal-based prediction and planning for autonomous driving. *arXiv preprint arXiv:2002.02277*, 2020. 2

[4] Peter W. Battaglia, Razvan Pascanu, Matthew Lai, Danilo Rezende, and Koray Kavukcuoglu. Interaction networks for learning about objects, relations and physics. In *Conf. on Neural Information Processing Systems*, 2016. 3

[5] Dhaivat Bhatt, Dishank Bansal, Gunshi Gupta, Hanju Lee, Krishna Murthy Jatavallabhula, and Liam Paull. Probabilistic object detection: Strengths, weaknesses, opportunities. In *Workshop on AI for Autonomous Driving at the International Conference on Machine Learning (ICML)*, 2020. 1

[6] Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight uncertainty in neural networks. In *International Conference on Machine Learning (ICML)*, volume 37, pages 1613–1622, 2015. 2

[7] National Transportation Safety Board. Preliminary report highway: HWY18MH010. https://www.ntsb.gov/

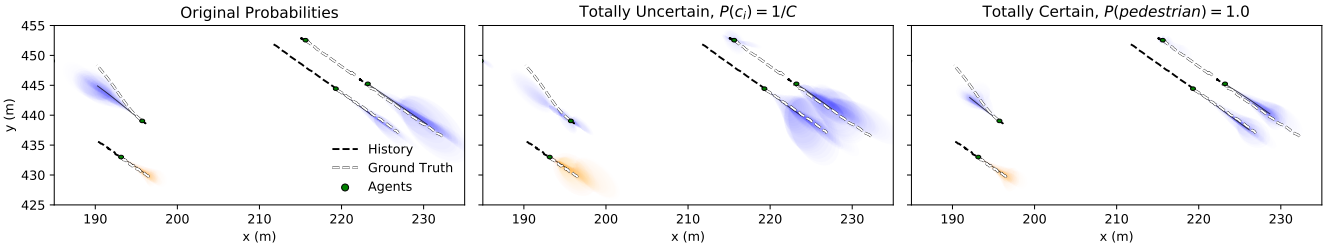

Figure 5. Our method is able to make counterfactual predictions, i.e., predictions where the input probability distribution is manually specified ahead of time in order to produce "what-if" prediction outputs. Prediction color denotes the original class of the agent (orange for pedestrians, blue for vehicles). **Left**: Our model's predictions with the original class probabilities for all agents. **Middle**: All agents have fully-uncertain class probabilities. **Right**: All agents have fully-certain pedestrian class probabilities.

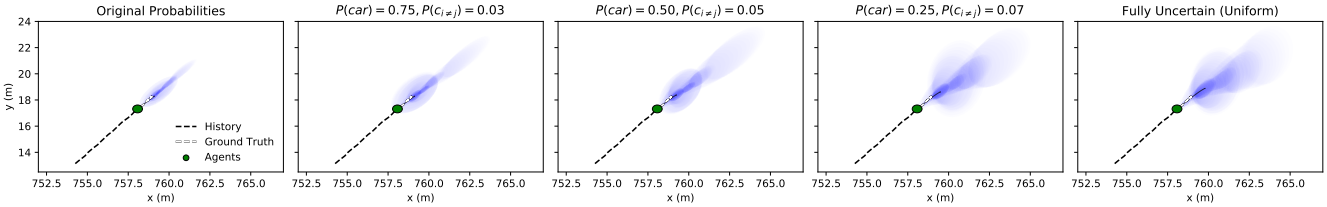

Figure 6. Our method's counterfactual predictions interpolate smoothly in the space of input probabilities.

investigations/AccidentReports/Reports/HAR1903.pdf, Mar 2018. Highway Accident Report Collision Between Vehicle Controlled by Developmental Automated Driving System and Pedestrian Tempe, Arizona. 1, 2

[8] Samuel R. Bowman, Luke Vilnis, Oriol Vinyals, Andrew M. Dai, Rafal Jozefowicz, and Samy Bengio. Generating sentences from a continuous space. In *Proc. Annual Meeting of the Association for Computational Linguistics*, 2015. 12

[9] Sebastian Brechtel, Tobias Gindele, and Rüdiger Dillmann. Probabilistic decision-making under uncertainty for autonomous driving using continuous POMDPs. In *17th international IEEE conference on intelligent transportation systems (ITSC)*, pages 392–399. IEEE, 2014. 2

[10] Sergio Casas, Cole Gulino, Renjie Liao, and Raquel Urtasun. SpAGNN: Spatially-aware graph neural networks for relational behavior forecasting from sensor data. In *International Conference on Robotics and Automation (ICRA)*, pages 9491–9497. IEEE, 2020. 2

[11] Sergio Casas, Cole Gulino, Simon Suo, Katie Luo, Renjie Liao, and Raquel Urtasun. Implicit latent variable model for scene-consistent motion forecasting. In *European Conference on Computer Vision (ECCV)*, pages 624–641, 11 2020. 2

[12] Sergio Casas, Wenjie Luo, and Raquel Urtasun. Intentnet: Learning to predict intention from raw sensor data. In *Conference on Robot Learning (CoRL)*, pages 947–956. PMLR, 2018. 2

[13] Yuning Chai, Benjamin Sapp, Mayank Bansal, and Dragomir Anguelov. Multipath: Multiple probabilistic anchor trajectory hypotheses for behavior prediction. *arXiv preprint arXiv:1910.05449*, 2019. 2

[14] Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. In *Proc. of Conf. on Empirical Methods in Natural Language Processing*, pages 1724–1734, 2014. 3

[15] Kurtland Chua, Roberto Calandra, Rowan McAllister, and Sergey Levine. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. In *Neural Information Processing Systems (NeurIPS)*, pages 4754–4765, 2018. 2

[16] Stefan Depeweg, José Miguel Hernández-Lobato, Finale Doshi-Velez, and Steffen Udluft. Learning and policy search in stochastic dynamical systems with Bayesian neural networks. *arXiv preprint arXiv:1605.07127*, 2016. 2

[17] Nemanja Djuric, Henggang Cui, Zhaoen Su, Shangxuan Wu, Huahua Wang, Fang-Chieh Chou, Luisa San Martin, Song Feng, Rui Hu, Yang Xu, Alyssa Dayan, Sidney Zhang, Brian C. Becker, Gregory P. Meyer, Carlos Vallespi-Gonzalez, and Carl K. Wellington. MultiXNet: Multiclass multistage multimodal motion prediction. *CoRR*, abs/2006.02000, 2020. 2

[18] Yarin Gal, Jiri Hron, and Alex Kendall. Concrete dropout. In *Neural Information Processing Systems (NeurIPS)*, volume 30, 2017. 2

[19] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Conf. on Neural Information Processing Systems*, 2014. 2

[20] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. On calibration of modern neural networks. In *International Conference on Machine Learning*. PMLR, 2017. 5

[21] Agrim Gupta, Justin Johnson, Li Fei-Fei, Silvio Savarese, and Alexandre Alahi. Social GAN: Socially acceptable trajectories with generative adversarial networks. In *IEEE Conf. on Computer Vision and Pattern Recognition*, 2018. 2, 6

[22] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 1997. 3

[23] John Houston, Guido Zuidhof, Luca Bergamini, Yawei Ye, Ashesh Jain, Sammy Omari, Vladimir Iglovikov, and Peter Ondruska. One thousand and one hours: Self-driving motion prediction dataset. In *Conf. on Robot Learning*, 2020. 2, 4, 7, 8, 12, 13, 14, 16, 17

[24] Masha Itkina, Boris Ivanovic, Ransalu Senanayake, Mykel J. Kochenderfer, and Marco Pavone. Evidential sparsification of multimodal latent spaces in conditional variational autoencoders. In *Conf. on Neural Information Processing Systems*, 2020. 3

[25] Boris Ivanovic, Amine Elhafsi, Guy Rosman, Adrien Gaidon, and Marco Pavone. MATS: An interpretable trajectory forecasting representation for planning and control. In *Conf. on Robot Learning*, 2020. 5, 7, 14, 15

[26] Boris Ivanovic and Marco Pavone. The Trajectron: Probabilistic multi-agent trajectory modeling with dynamic spatiotemporal graphs. In *IEEE Int. Conf. on Computer Vision*, 2019. 2, 3, 6

[27] Boris Ivanovic, Edward Schmerling, Karen Leung, and Marco Pavone. Generative modeling of multimodal multi-human behavior. In *IEEE/RSJ Int. Conf. on Intelligent Robots & Systems*, 2018. 2, 3

[28] Ashesh Jain, Amir R. Zamir, Silvio Savarese, and Ashutosh Saxena. Structural-RNN: Deep learning on spatio-temporal graphs. In *IEEE Conf. on Computer Vision and Pattern Recognition*, 2016. 3

[29] Eric Jang, Shixiang Gu, and Ben Poole. Categorial reparameterization with gumbel-softmax. In *Int. Conf. on Learning Representations*, 2017. 4

[30] Rudolf E. Kalman. A new approach to linear filtering and prediction problems. *ASME Journal of Basic Engineering*, 82:35–45, 1960. 3

[31] Kris M. Kitani, Brian D. Ziebart, James Andrew Bagnell, and Martial Hebert. Activity forecasting. In *European Conference on Computer Vision*, pages 201–214. Springer, 2012. 2

[32] Vineet Kosaraju, Amir Sadeghian, Roberto Martín-Martín, Ian Reid, S. Hamid Rezatofighi, and Silvio Savarese. Social-BiGAT: Multimodal trajectory forecasting using bicycle-GAN and graph attention networks. In *Conf. on Neural Information Processing Systems*, 2019. 6

[33] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. *arXiv preprint arXiv:1612.01474*, 2016. 2

[34] Steven M. LaValle. Better unicycle models. In *Planning Algorithms*, pages 743–743. Cambridge Univ. Press, 2006. 3, 6

[35] Stéphanie Lefèvre, Dizan Vasquez, and Christian Laugier. A survey on motion prediction and risk assessment for intelligent vehicles. *ROBOMECH journal*, 1(1):1–14, 2014. 2

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

[37] Yuexin Ma, Xinge Zhu, Sibo Zhang, Ruigang Yang, Wenping Wang, and Dinesh Manocha. Trafficpredict: Trajectory prediction for heterogeneous traffic-agents. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 6120–6127, 2019. 2

[38] Osama Makansi, Eddy Ilg, Ozgun Cicek, and Thomas Brox. Overcoming limitations of mixture density networks: A sampling and fitting framework for multimodal future prediction. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019. 2

[39] Rowan McAllister, Yarin Gal, Alex Kendall, Mark Van Der Wilk, Amar Shah, Roberto Cipolla, and Adrian Vivian Weller. Concrete problems for autonomous vehicle safety: advantages of Bayesian deep learning. In *International Joint Conferences on Artificial Intelligence*, 2017. 1

[40] Jean Mercat, Thomas Gilles, Nicole El Zoghby, Guillaume Sandou, Dominique Beauvois, and Guillermo Pita Gil. Multi-head attention for multi-modal joint vehicle motion forecasting. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 9638–9644. IEEE, 2020. 2

[41] Brian Paden, Michal Čáp, Sze Zheng Yong, Dmitry Yershov, and Emilio Frazzoli. A survey of motion planning and control techniques for self-driving urban vehicles. *IEEE Transactions on Intelligent Vehicles*, 1(1):33–55, 2016. 4

[42] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in PyTorch. In *Conf. on Neural Information Processing Systems - Autodiff Workshop*, 2017. 6

[43] Tung Phan-Minh, Elena Corina Grigore, Freddy A Boulton, Oscar Beijbom, and Eric M Wolff. Covernet: Multimodal behavior prediction using trajectory sets. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14074–14083, 2020. 2

[44] Andrea Piazzoni, Jim Cherian, Martin Slavik, and Justin Dauwels. Modeling sensing and perception errors towards robust decision making in autonomous vehicles. *arXiv preprint arXiv:2001.11695*, 2020. 2

[45] Nicholas Rhinehart, Kris M. Kitani, and Paul Vernaza. R2p2: A reparameterized pushforward policy for diverse, precise generative path forecasting. In *The European Conference on Computer Vision (ECCV)*, September 2018. 2

[46] Nicholas Rhinehart, Rowan McAllister, Kris Kitani, and Sergey Levine. PRECOG: Prediction conditioned on goals in visual multi-agent settings. In *International Conference on Computer Vision (ICCV)*, October 2019. 2, 4

[47] Timothy J. Ross. Engineering decisions involving aleatoric and epistemic uncertainty. 2006. 2

[48] Debaditya Roy, Tetsuhiro Ishizaka, C. Krishna Mohan, and Atsushi Fukuda. Vehicle trajectory prediction at intersections using interaction based generative adversarial networks. In *IEEE Intelligent Transportation Systems Conference (ITSC)*, pages 2318–2323, 2019. 2

[49] Andrey Rudenko, Luigi Palmieri, Michael Herman, Kris M. Kitani, Dariu M. Gavrila, and Kai O. Arras. Human motion trajectory prediction: A survey. *Int. Journal of Robotics Research*, 39(8):895–935, 2020. 1

[50] Amir Sadeghian, Vineet Kosaraju, Ali Sadeghian, Noriaki Hirose, S. Hamid Rezatofighi, and Silvio Savarese. SoPhie: An attentive GAN for predicting paths compliant to social

and physical constraints. In *IEEE Conf. on Computer Vision and Pattern Recognition*, 2019. 6

[51] Tim Salzmann, Boris Ivanovic, Punarjay Chakravarty, and Marco Pavone. Trajectron++: Dynamically-feasible trajectory forecasting with heterogeneous data. In *European Conf. on Computer Vision*, 2020. 1, 2, 3, 6, 7, 8, 12, 14, 15

[52] Edward Schmerling, Karen Leung, Wolf Vollprecht, and Marco Pavone. Multimodal probabilistic model-based planning for human-robot interaction. In *Proc. IEEE Conf. on Robotics and Automation*, 2018. 2

[53] Christoph Schöller, Vincent Aravantinos, Florian Lay, and Alois Knoll. What the constant velocity model can teach us about pedestrian motion prediction. *IEEE Robotics and Automation Letters*, 2020. 12

[54] Wilko Schwarting, Javier Alonso-Mora, and Daniela Rus. Planning and decision-making for autonomous vehicles. *Annual Review of Control, Robotics, and Autonomous Systems*, 2018. 2

[55] Kihyuk Sohn, Honglak Lee, and Xinchen Yan. Learning structured output representation using deep conditional generative models. In *Conf. on Neural Information Processing Systems*, 2015. 3

[56] Yichuan Charlie Tang and Ruslan Salakhutdinov. Multiple futures prediction. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. 2

[57] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. The extended Kalman filter. In *Probabilistic Robotics*, pages 54–64. MIT Press, 2005. 3

[58] Hang Zhao, Jiyang Gao, Tian Lan, Chen Sun, Benjamin Sapp, Balakrishnan Varadarajan, Yue Shen, Yi Shen, Yuning Chai, Cordelia Schmid, Congcong Li, and Dragomir Anguelov. TNT: Target-driven trajectory prediction. In *Conference on Robot Learning (CoRL)*. PMLR, 2020. 2

[59] Shengjia Zhao, Jiaming Song, and Stefano Ermon. InfoVAE: Balancing learning and inference in variational autoencoders. In *Proc. AAAI Conf. on Artificial Intelligence*, 2019. 4

[60] Tianyang Zhao, Yifei Xu, Mathew Monfort, Wongun Choi, Chris Baker, Yibiao Zhao, Yizhou Wang, and Ying N. Wu. Multi-agent tensor fusion for contextual trajectory prediction. In *IEEE Conf. on Computer Vision and Pattern Recognition*, 2019. 6

[61] Brian D. Ziebart, Nathan Ratliff, Garratt Gallagher, Christoph Mertz, Kevin Peterson, J. Andrew Bagnell, Martial Hebert, Anind K. Dey, and Siddhartha Srinivasa. Planning-based prediction for pedestrians. In *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3931–3936. IEEE, 2009. 2

A. Detailed Lyft Dataset Statistics

A.1. Class Switching

Class Switching Histogram. As mentioned in the main text, we find that 2.1% of all agents in the Lyft Level 5 dataset [23] experience class-switching, i.e., their highest probability class changes at least once during observation. Figure 7 dives deeper and shows the number of most-likely classes an agent has. For example, if an agent has 1 class then it experiences no class switches and has a consistent most-likely class throughout observation. Accordingly, if an agent has 2 classes then it experiences at least one class switch between two distinct classes during observation.

Examples of Class Switching. Figure 8 visualizes a scenario where a nearby car is misclassified as a pedestrian in the middle of an intersection. Another example is visualized in Figure 9 where a pedestrian adjacent to the ego-vehicle is misclassified as a car while waiting to cross the street, demonstrating that class switching is not solely due to an agent being very far away from sensor view.

Types of Class Switching. Figure 10 visualizes the 15 most common class switches (out of 45 total) in the Lyft dataset, as well as the mean agent class probabilities for each case. As can be seen, there are thousands of agents which experience known-class to known-class switches (e.g., pedestrian to car).

A.2. Probability of the Most-Likely Class

Figure 11 shows the average probability of the agent's most-likely class (indicated by the subfigure title). As can be seen, the most-likely class has more than 92% probability on average per agent timestep.

B. Detailed PUP Dataset Statistics

B.1. Scene-Agent Histograms

Figure 12 shows the distribution of agents within the PUP dataset's scenes. In particular, it visualizes a histogram over the number of agents in a scene of a specific type. For example, there are more than 200 scenes with ~ 30 cars in them and seldom any scenes with more than 20 bicycles.

C. Additional Results

C.1. Lyft Dataset

Table 4 shows the per-class performance of our method, its ablations, and baselines on the Lyft dataset. As in Table 2, our method generally performs the best across all classes and is similar to the one-hot ablation due to the abundance of one-hot class probabilities in the Lyft dataset.

C.2. PUP Dataset

Table 5 shows the per-class performance of our method, its ablations, and baselines on our PUP dataset. HAICU's strong performance across agent classes is evident, and while other baselines or ablations yield strong performance on specific classes, they are not able to maintain performance across agent classes in general.

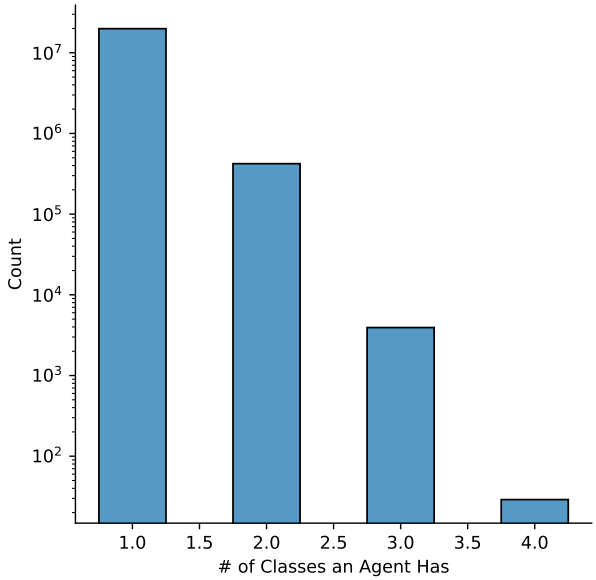

Figure 7. A histogram of the number of most-likely classes an agent has. For example, if an agent has 4 classes then it experiences at least three class switches between 4 distinct classes during observation

D. Additional Training Information

We anneal the β hyperparameter in Equation (3) following an increasing sigmoid [8]. Specifically, β takes a low value at early training iterations so that the model is encouraged to encode information in z. At later training iterations, a higher β value shifts the role of information encoding from $q_\phi(z \mid \mathbf{x}, \mathbf{y})$ to $p_\theta(z \mid \mathbf{x}, \hat{\mathbf{c}})$.

To avoid overfitting to environment-specific characteristics, such as the general directions that agents move, we augment the data from each scene with rotation [53]. In particular, we rotate all trajectories in a scene around the scene's origin by γ, where γ varies from 0° to 360° (exclusive) in 15° intervals, as in [51]. We apply this augmentation to autonomous driving datasets because most of them are recorded in cities whose streets are roughly orthogonal and separated by blocks. While this augmentation is equivalent to rotating all trajectories to a canonical agent-centric frame, we chose to rotate all trajectories at train-time and train the model to be rotation-invariant since it avoids the need to perform canonical (and possibly noisy) trajectory rotation online at test time.

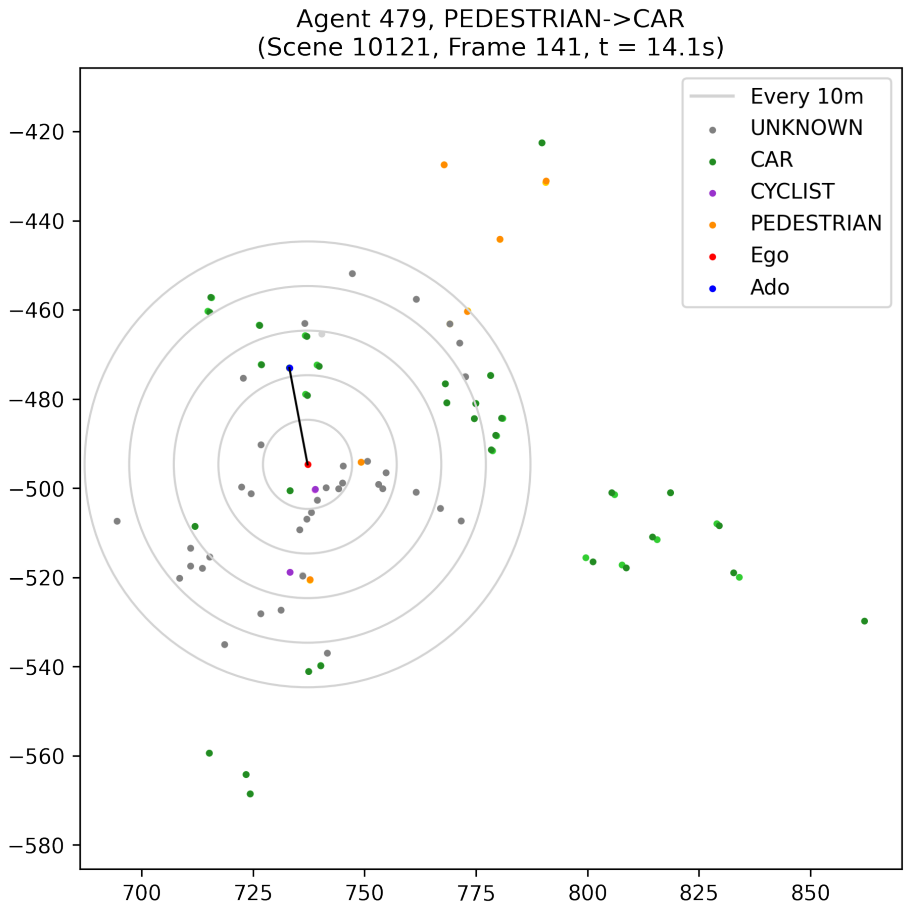

Figure 8. A scenario in the Lyft Level 5 dataset [23] where a nearby car is misclassified as a pedestrian in the middle of an intersection. In this example, the misclassified agent (in blue) is only around 20m away from the ego-vehicle (in red). The solid black line indicates the distance between the two agents and the light gray lines mark 10m radius increments from the ego-vehicle. The light/dark versions of the other agent colors show the location of the associated agent in the previous frame (light color) and the current frame (dark color). The title indicates the class switch that occurs from the previous to the current frame for the misclassified agent.

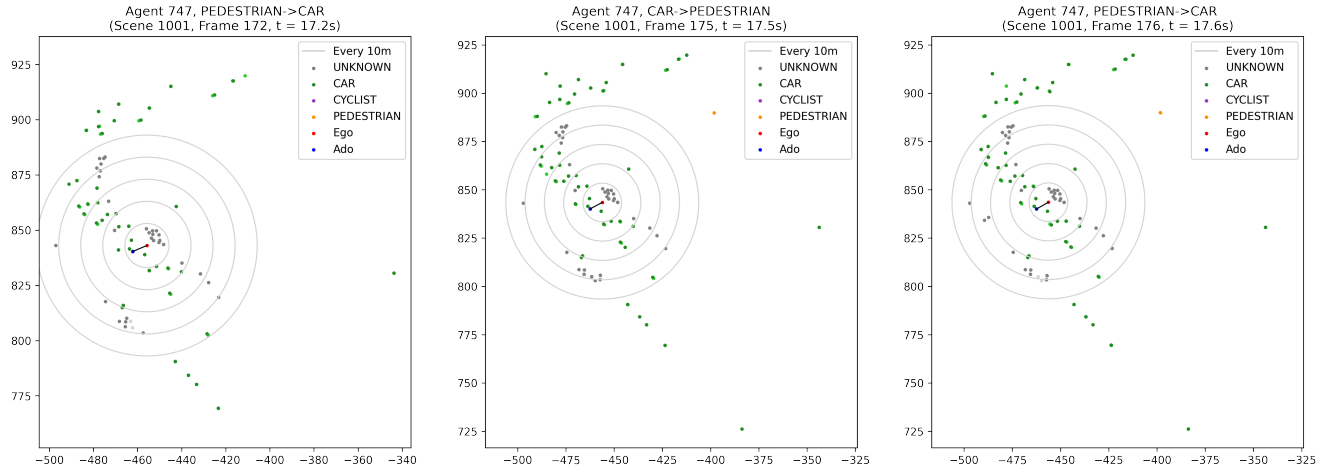

Figure 9. A scenario in the Lyft Level 5 dataset [23] where a pedestrian next to the ego-vehicle is misclassified as a car while waiting to cross the street. In this example, the misclassified agent (in blue) is less than 10m away from the ego-vehicle (in red). The solid black line indicates the distance between the two agents and the light gray lines mark 10m radius increments from the ego-vehicle. The light/dark versions of the other agent colors show the location of the associated agent in the previous frame (light color) and the current frame (dark color). The title indicates the class switch that occurs from the previous to the current frame for the misclassified agent.

Lyft – Car	**ADE (m)**			**FDE (m)**			**ANLL (nats)**			**FNLL (nats)**		
Pred. Horizon	1s	2s	3s	1s	2s	3s	1s	2s	3s	1s	2s	3s
MATS [25]	0.31	0.99	2.33	0.46	2.21	6.17	−0.21	1.62	4.75	0.85	4.10	13.67
Trajectron++ [51]	0.23	0.47	0.77	0.43	0.97	1.74	−2.07	−0.94	−0.19	−0.63	0.75	1.71
HAICU (Multi-Head)	0.24	0.48	0.80	0.43	1.00	1.81	−2.07	−0.94	−0.20	−0.63	0.73	1.71
HAICU (One-Hot)	0.22	0.46	0.76	0.42	0.95	**1.71**	−2.47	**−1.34**	**−0.66**	**−0.91**	**0.24**	**1.09**
HAICU	**0.21**	**0.45**	**0.75**	**0.40**	**0.94**	**1.71**	**−2.48**	**−1.34**	−0.64	**−0.91**	0.27	1.17
Lyft – Cyclist	**ADE (m)**			**FDE (m)**			**ANLL (nats)**			**FNLL (nats)**		
Pred. Horizon	1s	2s	3s	1s	2s	3s	1s	2s	3s	1s	2s	3s
MATS [25]	0.22	0.61	1.14	0.33	1.25	2.45	−0.80	0.68	3.20	0.16	2.59	11.14
Trajectron++ [51]	0.14	0.35	0.64	0.29	0.81	1.58	−2.64	−1.31	−0.45	−0.93	0.66	1.70
HAICU (Multi-Head)	0.14	0.32	0.55	0.28	0.71	1.26	−2.72	−1.47	−0.67	−1.08	0.38	1.33
HAICU (One-Hot)	0.13	0.30	0.51	0.27	0.66	1.17	−3.08	−1.82	**−1.07**	−1.34	**−0.04**	**0.78**
HAICU	**0.12**	**0.28**	**0.48**	**0.25**	**0.62**	**1.12**	**−3.10**	**−1.84**	**−1.07**	**−1.37**	**−0.04**	**0.78**
Lyft – Pedestrian	**ADE (m)**			**FDE (m)**			**ANLL (nats)**			**FNLL (nats)**		
Pred. Horizon	1s	2s	3s	1s	2s	3s	1s	2s	3s	1s	2s	3s
MATS [25]	0.21	0.40	0.68	0.28	0.70	1.42	−0.53	0.35	3.29	0.06	1.61	10.69
Trajectron++ [51]	0.09	0.19	**0.30**	0.18	0.39	**0.62**	−3.20	−1.90	−1.14	−1.40	−0.09	0.74
HAICU (Multi-Head)	0.09	0.19	**0.30**	0.18	0.39	**0.62**	−3.40	**−2.12**	**−1.36**	**−1.64**	**−0.30**	**0.53**
HAICU (One-Hot)	0.09	0.19	**0.30**	**0.17**	**0.38**	0.63	−3.45	**−2.12**	−1.33	−1.60	−0.24	0.64
HAICU	**0.08**	**0.18**	**0.30**	**0.17**	**0.38**	0.65	**−3.46**	**−2.12**	−1.31	**−1.64**	**−0.30**	**0.53**
Lyft – Unknown	**ADE (m)**			**FDE (m)**			**ANLL (nats)**			**FNLL (nats)**		
Pred. Horizon	1s	2s	3s	1s	2s	3s	1s	2s	3s	1s	2s	3s
MATS [25]	0.22	0.40	0.71	0.28	0.71	1.57	−0.55	0.30	3.87	−0.09	1.57	12.76
Trajectron++ [51]	**0.09**	**0.14**	0.18	**0.15**	0.22	0.28	**−3.61**	**−2.84**	**−2.43**	**−2.43**	**−1.85**	**−1.42**
HAICU (Multi-Head)	**0.09**	**0.14**	**0.17**	**0.15**	**0.21**	**0.27**	−3.50	−2.78	−2.38	−2.40	−1.83	−1.39
HAICU (One-Hot)	0.10	0.15	0.18	0.16	0.22	0.28	−3.49	−2.77	−2.38	−2.38	−1.83	−1.40
HAICU	**0.09**	**0.14**	0.18	**0.15**	0.22	0.28	−3.53	−2.78	−2.38	−2.38	−1.81	−1.38

Table 4. Per-class performance of our method, its ablations, and baselines on the Lyft Level 5 dataset [23]. Lower is better, bold is best.

PUP – Bicycle	**ADE (m)**			**FDE (m)**			**ANLL (nats)**			**FNLL (nats)**		
Pred. Horizon	1s	2s	3s	1s	2s	3s	1s	2s	3s	1s	2s	3s
MATS [25]	0.76	1.39	2.33	0.76	2.03	4.20	3.54	4.67	7.02	3.54	5.81	11.70
Trajectron++ [51]	0.23	0.38	0.61	0.35	0.74	1.37	−1.71	−1.10	−0.63	−1.01	−0.08	0.60
HAICU (Multi-Head)	0.29	0.48	0.71	0.45	0.88	1.42	−1.70	−1.03	−0.53	−0.95	0.06	0.81
HAICU (One-Hot)	**0.21**	**0.36**	0.56	**0.32**	0.69	1.23	**−2.12**	**−1.48**	**−0.99**	−1.40	**−0.41**	**0.31**
HAICU	**0.21**	**0.36**	**0.54**	0.33	**0.66**	**1.13**	**−2.12**	−1.47	−0.98	**−1.41**	−0.38	0.32
PUP – Car	**ADE (m)**			**FDE (m)**			**ANLL (nats)**			**FNLL (nats)**		
Pred. Horizon	1s	2s	3s	1s	2s	3s	1s	2s	3s	1s	2s	3s
MATS [25]	0.42	0.65	1.01	0.42	0.89	1.73	1.57	2.10	5.89	1.57	2.63	13.47
Trajectron++ [51]	0.25	0.40	0.57	0.37	0.68	1.12	−1.88	−1.44	−1.10	−1.39	−0.73	−0.14
HAICU (Multi-Head)	0.32	0.51	0.70	0.50	0.86	1.31	−1.88	−1.38	−0.98	−1.34	−0.55	0.15
HAICU (One-Hot)	0.25	**0.38**	0.55	**0.36**	**0.65**	1.10	−2.12	−1.65	−1.29	−1.58	−0.90	−0.27
HAICU	**0.24**	**0.38**	**0.54**	**0.36**	**0.65**	**1.09**	**−2.15**	**−1.67**	**−1.31**	**−1.60**	**−0.92**	**−0.29**
PUP – Largevehicle	**ADE (m)**			**FDE (m)**			**ANLL (nats)**			**FNLL (nats)**		
Pred. Horizon	1s	2s	3s	1s	2s	3s	1s	2s	3s	1s	2s	3s
MATS [25]	0.62	0.94	1.42	**0.62**	1.27	2.39	5.02	5.42	8.77	5.02	5.83	15.46
Trajectron++ [51]	0.48	0.68	0.88	0.66	1.04	1.54	−0.42	−0.03	0.31	−0.03	0.65	1.27
HAICU (Multi-Head)	0.58	0.83	1.07	0.83	1.28	1.80	−0.44	0.03	0.42	0.03	0.82	1.51
HAICU (One-Hot)	**0.46**	**0.64**	**0.85**	**0.62**	**1.00**	**1.53**	−0.80	−0.33	0.03	−0.27	0.43	**1.07**
HAICU	**0.46**	0.65	0.88	0.63	1.03	1.60	**−0.81**	**−0.35**	**0.02**	**−0.29**	**0.42**	**1.07**
PUP – Motorcycle	**ADE (m)**			**FDE (m)**			**ANLL (nats)**			**FNLL (nats)**		
Pred. Horizon	1s	2s	3s	1s	2s	3s	1s	2s	3s	1s	2s	3s
MATS [25]	0.28	0.39	0.59	0.28	**0.51**	0.98	0.74	1.04	4.14	0.74	1.34	10.34
Trajectron++ [51]	0.24	0.38	0.52	0.36	0.64	0.96	−1.51	−0.99	−0.57	−0.98	−0.10	0.49
HAICU (Multi-Head)	0.25	0.37	0.53	0.34	0.63	1.05	−1.51	−0.92	−0.43	−0.92	0.13	0.92
HAICU (One-Hot)	**0.20**	**0.30**	**0.43**	**0.27**	**0.51**	**0.86**	**−1.96**	**−1.43**	**−1.02**	**−1.41**	**−0.52**	**0.07**
HAICU	0.21	0.32	0.47	0.29	0.56	0.98	−1.91	−1.37	−0.94	−1.36	−0.43	0.21
PUP – Pedestrian	**ADE (m)**			**FDE (m)**			**ANLL (nats)**			**FNLL (nats)**		
Pred. Horizon	1s	2s	3s	1s	2s	3s	1s	2s	3s	1s	2s	3s
MATS [25]	0.32	0.52	0.79	**0.32**	0.72	1.32	0.27	1.31	2.47	0.27	2.35	4.79
Trajectron++ [51]	**0.21**	**0.34**	**0.50**	**0.32**	**0.62**	**0.97**	**−1.18**	**−0.28**	**0.36**	**−0.17**	**1.13**	**2.05**
HAICU (Multi-Head)	0.26	0.44	0.65	0.41	0.82	1.29	−0.89	0.02	0.66	0.14	1.45	2.31
HAICU (One-Hot)	0.23	0.39	0.58	0.36	0.72	1.18	−1.10	−0.22	0.41	−0.11	1.18	2.06
HAICU	0.24	0.42	0.62	0.38	0.77	1.24	−1.04	−0.16	0.48	−0.04	1.25	2.15
PUP – Unknown	**ADE (m)**			**FDE (m)**			**ANLL (nats)**			**FNLL (nats)**		
Pred. Horizon	1s	2s	3s	1s	2s	3s	1s	2s	3s	1s	2s	3s
Trajectron++ [51]	0.40	0.88	1.44	0.79	1.86	3.14	−0.56	1.17	1.89	1.80	3.28	3.65
HAICU (Multi-Head)	0.41	0.88	1.41	0.79	1.83	2.97	−0.59	1.01	1.96	1.32	3.28	3.74
HAICU (One-Hot)	0.29	0.68	1.16	0.59	1.51	2.58	−1.13	0.55	1.46	1.25	2.86	3.26
HAICU	**0.18**	**0.47**	**0.86**	**0.40**	**1.09**	**2.10**	**−1.57**	**−0.14**	**0.64**	**0.07**	**2.00**	**2.32**

Table 5. Per-class performance of our method, its ablations, and baselines on our PUP dataset. Lower is better, bold is best. We could not evaluate MATS [25] on unknown agents (0.1% of the data) due to MATS' training-time computational requirements. Specifically, constructing dense square matrices (and backpropagating through them) to model many batched scenes exhausted our computational resources. To remedy this, we temporally subsampled our PUP dataset for MATS, which removed (short-lived) unknown agents.

Figure 10. The 15 most common class switches (out of 45 total) in the Lyft Level 5 dataset [23], as well as the mean agent class probabilities over time for each case. Each subfigure title indicates the type of class switch as well as the number of affected agents in brackets.

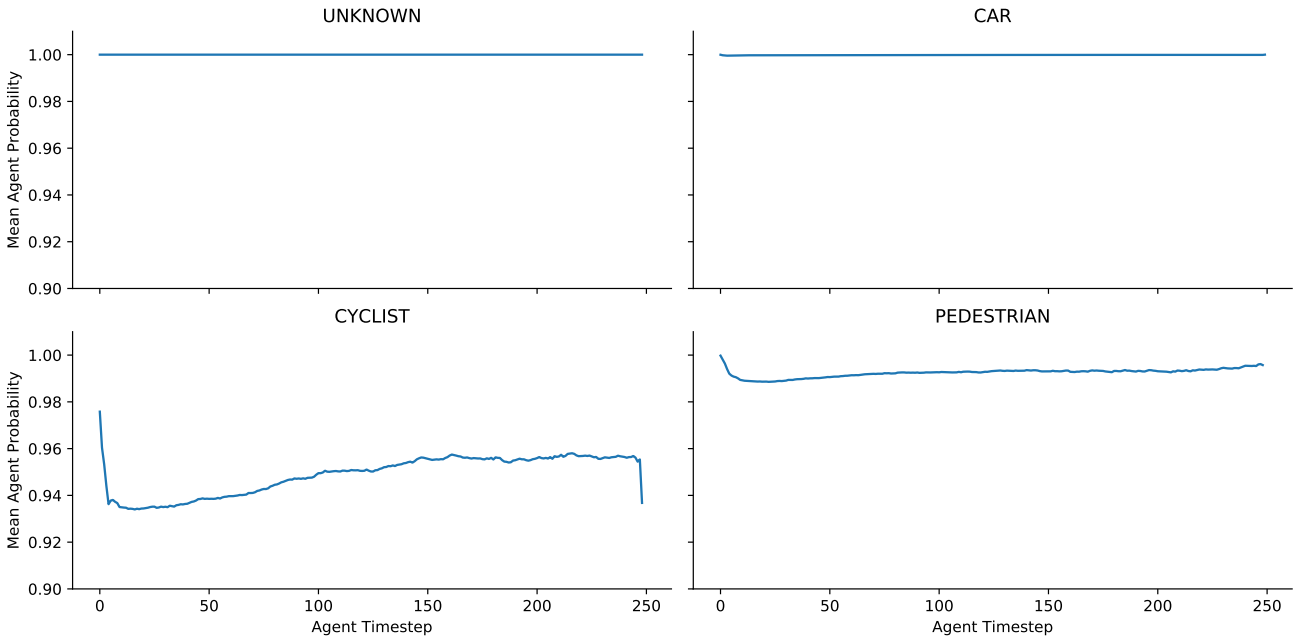

Figure 11. The average probability of the agent's most-likely class over time for each class in the Lyft Level 5 dataset [23].

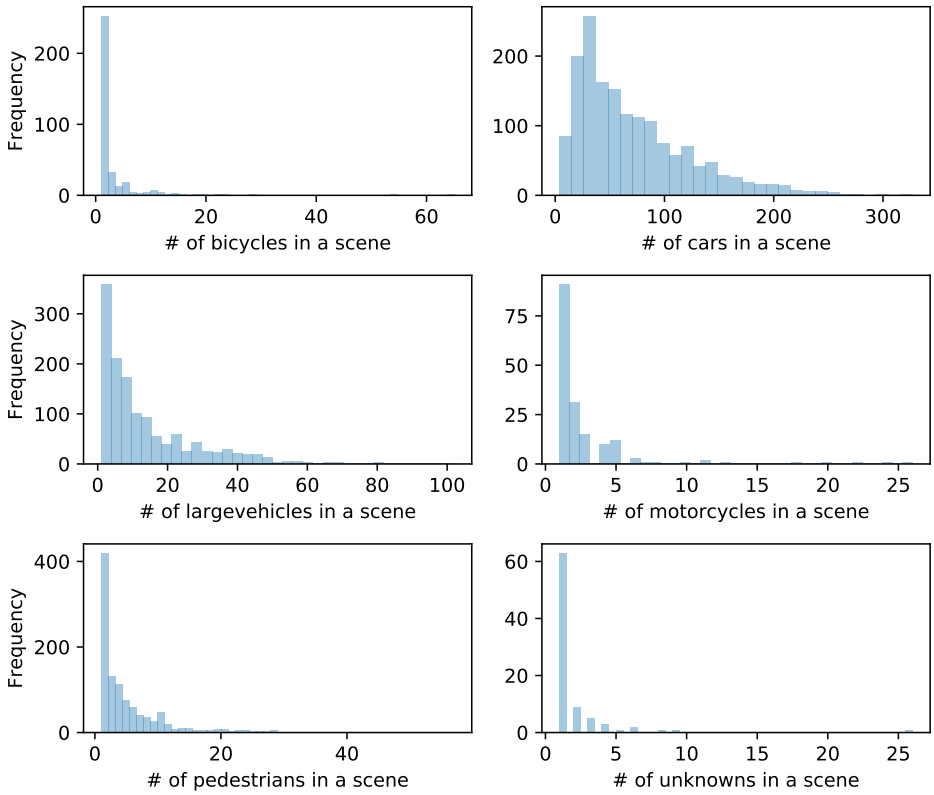

Figure 12. The distribution of the number of agents in a scene of a specific type. For example, there are more than 200 scenes with $\sim$ 30 cars in them.