

Heterogeneous-Agent Trajectory Forecasting Incorporating Class Uncertainty

Boris Ivanovic^{1†} Kuan-Hui Lee² Pavel Tokmakov² Blake Wulfe²
Rowan McAllister² Adrien Gaidon² Marco Pavone^{1,3}

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.

I. INTRODUCTION

Incorporating perceptual uncertainty into downstream components, such as forecasting and planning, is critical for the safe operation of autonomous vehicles. However, most trajectory forecasting methods do not explicitly incorporate or propagate perceptual uncertainties from their inputs to their outputs [1]. Instead, they classify an agent with its highest probability class from the upstream perception system. Although, blindly trusting the most-likely class can have disastrous consequences, especially in the presence of class uncertainty, as shown in Fig. 1. Misclassifications like these could cause an autonomous vehicle to make unnecessary evasive maneuvers and may occur frequently in challenging real-world conditions (Section V). A safer way to deal with perceptual uncertainties is to propagate them through forecasting systems, so that planning components can make uncertainty-aware decisions [2], [3].

Contributions. Our key contributions are threefold. First, we present *HAICU*, a method for Heterogeneous-Agent

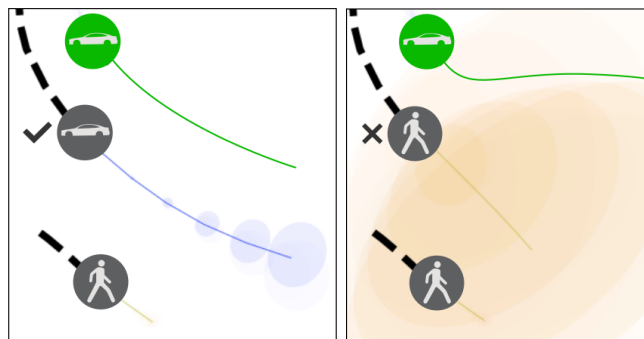


Fig. 1. Classifying agents with their most-likely class can be disastrous in the presence of uncertainty. A state-of-the-art trajectory forecasting model, Trajectron++ [4], produces well-behaved predictions when agents are correctly classified (left; as a car). However, its uncertainty grows sharply when the agent is misclassified (right; as a pedestrian). This can cause a sudden and unexpected change in the ego-vehicle’s resulting motion plan, in green.

trajectory forecasting Incorporating Class Uncertainty (Section IV). We show that directly incorporating class probabilities from upstream perception systems into a state-of-the-art trajectory forecasting method effectively improves performance in the presence of uncertainty (i.e., object classification error) without any reduction in overall accuracy or added computational complexity (Section VI). We also demonstrate how this enables fine-grained introspection via counterfactual predictions by modifying class probabilities directly to produce “what-if” predictions (Section VI-B).

Second, we analyze the Lyft Level 5 dataset [5] and show that, while it is currently the only public dataset that contains class probabilities, they are overconfident and thus unsuitable for the study of perceptual uncertainty (Section V-A).

Finally, we present the Perceptual Uncertainty in Prediction (*PUP*) dataset: a new challenging, real-world autonomous driving dataset with complex scenes and unfiltered 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 in trajectory forecasting (Section V-B). Our experiments on the Lyft [5] and *PUP* datasets show that *HAICU* significantly improves upon existing approaches, thanks to the inclusion of class probabilities.

II. RELATED WORK

Modular Trajectory Forecasting. Modular methods decompose autonomous driving into distinct sub-tasks, usually perception, prediction, and control [6]. A typical interface between perception and forecasting communicates only the most likely class and state estimate of each object

*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.

[†]This work was completed while the author was at Stanford University.

¹Boris Ivanovic is with NVIDIA Research {bivanovic@nvidia.com}

²Kuan-Hui Lee, Pavel Tokmakov, Blake Wulfe, Rowan McAllister, and Adrien Gaidon are with the Toyota Research Institute {first.last}@tri.global

³Marco Pavone is with the Department of Aeronautics and Astronautics, Stanford University, and with NVIDIA Research {pavone@stanford.edu, mpavone@nvidia.com}

detected by a perception system. As a result, trajectory forecasting methods usually assume their inputs are known with certainty [7], [8], [9]. In reality, sensors are imperfect and incorrect assumptions of certainty-equivalence in perception—where only the most likely class estimate is passed to prediction (but not its uncertainty)—can lead to disastrous outcomes, as in Fig. 1.

To the best of 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 [10], [11] or variational inference [12], [13], [14], [4]. Generative Adversarial Networks (GANs) [15] can generate empirical trajectory distributions from sampling multiple predictions [16], [17]. However, analytic distributions are often more useful for gradient-based planning that minimizes collision likelihood [6]. Thus, we focus on methods that predict analytic trajectory distributions.

End-to-End Prediction. End-to-end prediction methods perform detection, tracking, and prediction jointly, operating directly on raw sensor data. FaF [18] introduced the approach of projecting LiDAR points into a bird’s eye view (BEV) grid, and generating predictions through inferred future detections. This approach was extended by IntentNet [19], which incorporated HD map information and predicted agent intent. SpAGNN [20] modeled agent interactions using a graph network, and ILVM [21] extended it by modeling the joint distribution over future trajectories with a latent variable model. These methods only consider homogeneous agents (vehicles); however, MultiXNet [22] recently extended the BEV approach to heterogeneous agents using separate outputs per agent class. While this approach accounts for class uncertainty, the number of predicted trajectories scales with the number of classes or requires a hard selection of the class 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 the effects of perceptual uncertainty.

Uncertainty Propagation. Methods for propagating uncertainty through neural networks broadly view input data as noisy samples of a true underlying data distribution, and focus on both estimating the true distribution as well as propagating its uncertainty to the output. Towards this end, Bayesian neural networks [23], [24] and Markov models [25] are commonly applied. Our work differs as it does not need to perform estimation; object classifiers can fully characterize their output confidence, e.g., as a Categorical distribution over classes, and provide it to downstream modules.

III. PROBLEM FORMULATION

We aim to generate plausible future trajectory distributions for a time-varying number $N(t)$ of diverse interacting agents $A_1, \dots, A_{N(t)}$. 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 $k = 1, \dots, K$, producing a vector of class probabilities constrained to the $(K - 1)$ -simplex $\hat{c}_i^{(t)} \in \Delta^{K-1}$ 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 $\Delta^{(K-1)} = \{\mathbf{v} \in \mathbb{R}^K \mid \sum_{k=1}^K v_k = 1 \text{ and } v_k \geq 0 \forall i\}$. 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 $\hat{c}_i^{(t)}$, and their histories for the previous H timesteps, which we denote as $\mathbf{x} = \mathbf{s}_{1, \dots, N(t)}^{(t-H:t)} \in \mathbb{R}^{(H+1) \times N(t) \times D}$ and $\hat{\mathbf{c}} = \hat{\mathbf{c}}_{1, \dots, N(t)}^{(t-H:t)} \in \mathbb{R}^{(H+1) \times N(t) \times K}$, our goal is to produce a distribution over all agents’ future states for the next T timesteps, $\mathbf{y} = \mathbf{s}_{1, \dots, N(t)}^{(t+1:t+T)} \in \mathbb{R}^{T \times N(t) \times D}$, which we denote as $p(\mathbf{y} \mid \mathbf{x}, \hat{\mathbf{c}})$.

IV. INCORPORATING CLASS UNCERTAINTY IN TRAJECTORY FORECASTING

We build upon the Trajectron++ [4] framework to implement HAICU¹, 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. Fig. 2 visualizes HAICU’s probabilistic graphical model and 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 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 $\mathbf{s}_{1, \dots, N(t)}^{(t-H:t)} \in \mathbb{R}^{(H+1) \times N(t) \times D}$ are fed into a Long Short-Term Memory (LSTM) network [27] 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 [28], [13], [29], [14], [4]. 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 representation vector, $e_{\mathbf{x}}$.

Accounting for Multimodality. Our model leverages the Conditional Variational Autoencoder (CVAE) latent variable framework [30] 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

¹All code, models, dataset samples, and dataset links will be made available at <https://github.com/TRI-ML/HAICU>.

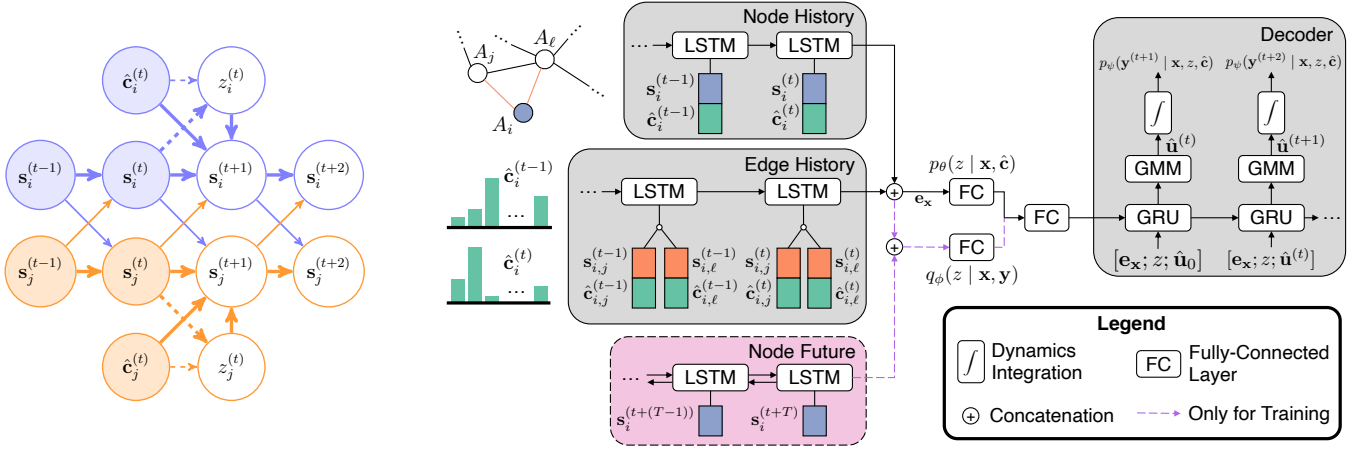


Fig. 2. **Left:** Our inference-time probabilistic graphical model for forecasting taking into account agent interactions and class probabilities over time, illustrated for two agents (blue and orange). Known values are shaded and we use thick-arrow notation for carry-forward dependencies [26]. Perception provides class-probabilities for each agent \hat{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’s network architecture, incorporating agent class uncertainty by encoding class probability values alongside the agent’s state.

$z \in Z$ which encodes high-level latent behavior and allows for the desired distribution $p(y | x, \hat{c})$ to be expressed as $p(y | x, \hat{c}) = \sum_{z \in Z} p_{\psi}(y | x, z, \hat{c}) p_{\theta}(z | x, \hat{c})$, 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 [31].

Generating Trajectories. The latent variable z and node representation vector e_x are then fed into the decoder, a 128-dimensional Gated Recurrent Unit (GRU) [32]. 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 [33], [34]. 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 [35] posits that agents are subject to non-holonomic constraints [36], 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 VI.

Incorporating Class Uncertainty. To incorporate class probabilities in our model, we concatenate the input class probability vector $\hat{c}_i^{(t)}$ with the state $s_i^{(t)}$ 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. Concretely, the reason why a CVAE can associate input uncertainty patterns to output trajectories is because the decoder $p_{\psi}(y | x, z, \hat{c})$ directly conditions on the input probabilities \hat{c} .

Training the Model. We adopt the same discrete InfoVAE [37] objective function as in Trajectron++. Formally, for each training example $(\{\mathbf{x}_i, \hat{c}_i\}, \mathbf{y}_i)$, we aim to maximize

$$\mathbb{E}_{z \sim q_{\phi}(\cdot | \mathbf{x}_i, \mathbf{y}_i)} [\log p_{\psi}(\mathbf{y}_i | \mathbf{x}_i, z, \hat{c}_i)] - \beta D_{KL}(q_{\phi}(z | \mathbf{x}_i, \mathbf{y}_i) \| p_{\theta}(z | \mathbf{x}_i, \hat{c}_i)) + I_q(\mathbf{x}_i, z), \quad (1)$$

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 $q_{\phi}(z | \mathbf{x}_i, \mathbf{y}_i)$ with $p_{\theta}(z | \mathbf{x}_i)$ and obtain the unconditioned latent distribution by summing out \mathbf{x}_i over the batch [37]. During training, a bi-directional LSTM with 32 hidden dimensions is used to encode a node’s ground truth future trajectory, producing $q_{\phi}(z | x, y)$ [30].

V. DATASETS

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

A. 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, and so far only, to release class probabilities for each detected agent.

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 14.5k agents with known class-to-class switches. For example, Appendix B.2 visualizes a scenario where a nearby car is misclassified as a pedestrian in the middle of an intersection. Another example in Appendix B.2 shows a pedestrian adjacent to the ego-vehicle being misclassified as

TABLE I

CLASS COUNTS AND UNCERTAINTIES FOR PUP COMPARED TO LYFT [5].

Class	PUP (Ours)		Lyft Level 5 [5]	
	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

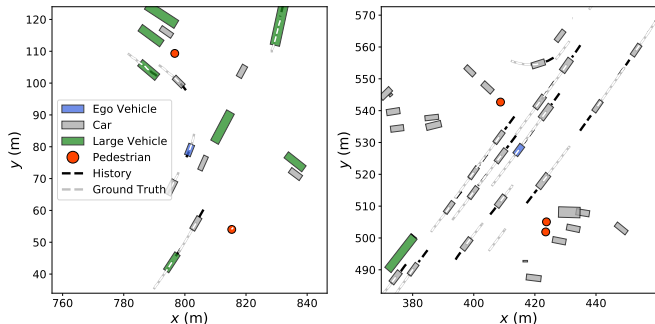


Fig. 3. Example scenes from our new PUP dataset.

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 [38]. Table I (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 B.4 shows that the most-likely class has more than 90% probability on average.

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.

B. PUP Dataset

To provide more options for studying the effects of perceptual uncertainty in downstream tasks, one of our core

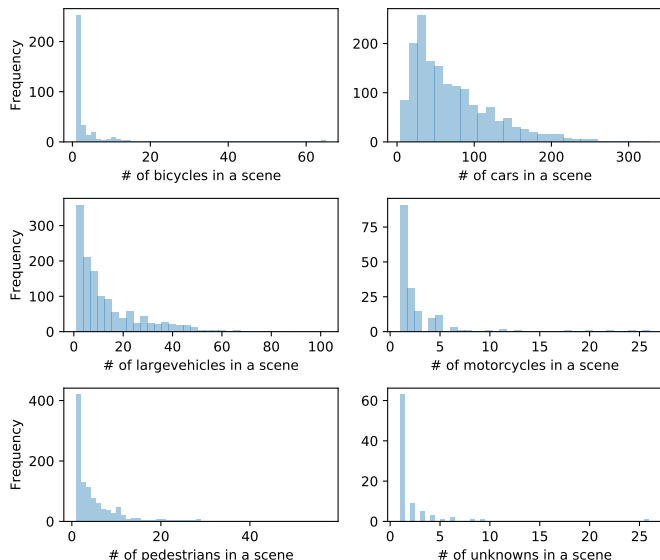


Fig. 4. Distribution of the number of agents of a specific type in a scene. For instance, there are more than 200 scenes with 30 cars in them and seldom any scenes with more than 10 unknown objects.

contributions is PUP, a novel real-world autonomous driving dataset comprised of 1,637 distinct scenes collected with a state-of-the-art self-driving fleet in Tokyo, Japan. Each scene is 10s long, 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 camera- and LiDAR-based perception stack running in production. Fig. 4 shows the distribution of agents within the PUP dataset’s scenes. In particular, it visualizes a histogram over the number of agents of a specific type in a scene, showing that PUP contains scenes with hundreds of unique, diverse agents interacting simultaneously.

Most importantly, since our goal is to quantitatively evaluate the performance of trajectory forecasting in the presence of class uncertainty, we do not perform any post-hoc filtering or smoothing of the perceived agent class probabilities, intentionally releasing the raw frame-by-frame outputs to enable a wide variety of future work (e.g., developing low-latency strategies for temporal smoothing, uncertainty-aware trajectory forecasting, planning under uncertainty). Table I 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, PUP’s class probabilities are more uncertain, with far fewer unknown agents.

To quantify the performance of the perception stack used to collect PUP, we evaluate it on a human-annotated dataset from the same region. Fig. 5 shows the confusion matrix of the object classifier, which is diagonal except for intuitive mistakes, e.g., bicycles and motorcycles. Accordingly, its top- k accuracies are 96.8%, 97.7%, 99.2%, 99.3%, 99.6% for $k = 1, \dots, 5$, respectively. For reference, the best top-1 classification accuracies on similar object detection tasks are 92–97% [5], [39], [40]. Finally, PUP was collected with these AP@0.5 values: 0.75 for cars, 0.49 for largevehicles, 0.80 for pedestrians, and 0.50 for motorcycles and bicycles.

TABLE II

OUR MODEL SIGNIFICANTLY OUTPERFORMS STATE-OF-THE-ART HETEROGENEOUS-AGENT METHODS ON THE LYFT LEVEL 5 DATASET [5]. IT IS EXPECTED THAT OUR METHOD PERFORMS SIMILARLY TO ONE-HOT BECAUSE THE LYFT DATA MOSTLY CONTAINS ONE-HOT CLASS PROBABILITIES.

RESULTS WITH THE MINADE AND MINFDE METRICS CAN BE FOUND IN APPENDIX F. SE = STANDARD ERROR.

Lyft Level 5 [5]	ADE \pm SE		FDE \pm SE (m)		ANLL \pm SE	FNLL \pm SE (nats)		
	Pred. Horizon		3s	1s	2s	3s	1s	2s
MATS [41]	1.21 \pm 0.13	0.34 \pm 0.04	1.22 \pm 0.15	2.90 \pm 0.32	3.78 \pm 0.34	0.25 \pm 0.18	2.47 \pm 0.24	12.06 \pm 1.01
Trajectron++ [4]	0.47 \pm 5e-3	0.26 \pm 3e-3	0.60 \pm 7e-3	1.05 \pm 0.01	-1.06 \pm 0.02	-1.34 \pm 0.02	-0.13 \pm 0.02	0.68 \pm 0.02
Multi-Head [22]	0.45 \pm 5e-3	0.26 \pm 3e-3	0.58 \pm 7e-3	0.99 \pm 0.01	-1.15 \pm 0.02	-1.44 \pm 0.02	-0.25 \pm 0.02	0.54 \pm 0.02
One-Hot	0.44 \pm 5e-3	0.25 \pm 2e-3	0.56 \pm 6e-3	0.95 \pm 0.01	-1.36\pm0.02	-1.56 \pm 0.02	-0.47\pm0.02	0.27\pm0.03
HAICU (ours)	0.43\pm5e-3	0.24\pm2e-3	0.54\pm6e-3	0.94\pm0.01	-1.35 \pm 0.02	-1.57\pm0.02	-0.45 \pm 0.02	0.31 \pm 0.03

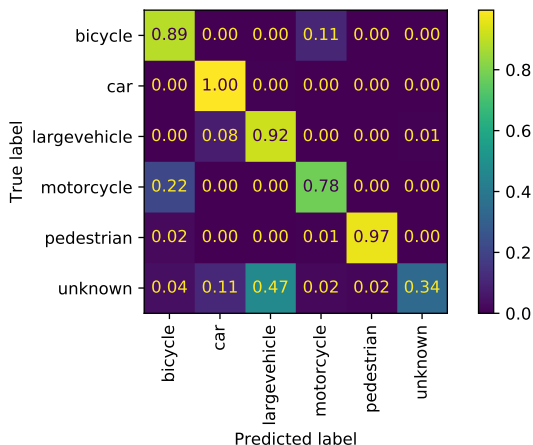


Fig. 5. The (normalized) confusion matrix of the onboard classification system that provides agent class probabilities for the PUP dataset.

VI. EXPERIMENTS

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 [41]: each scene is modeled with a mixture of affine dynamical systems, forward-integrated to produce predictions,

(2) Trajectron++ [4]: 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).

(3) Multi-Head: Rather than encoding agent classes in e_x , methods like MultiXNet [22] are multi-headed and produce an output for each possible agent type. We implement the same, additionally augmenting each head with a dynamics model (i.e., dynamically-extended unicycle [35] for vehicles and single integrator for others, as in [4]); these outputs are then combined in a mixture model where the class probabilities $\hat{c}_i^{(t)}$ are the mixing probabilities.

(4) One-Hot: An ablation of HAICU 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.

Metrics. We evaluate our approach with a variety of deterministic and probabilistic metrics: *Average/Final Displacement Error (ADE/FDE)*: mean/final ℓ_2 distance between the ground truth and predicted trajectories, *Average/Final*

Negative Log-Likelihood (ANLL/FNLL): the mean/final NLL of the ground truth trajectory under the predicted distribution. *minADE/minFDE*: ADE/FDE between the ground truth and best of 20 samples [16].

For ADE/FDE we compare methods’ single most-likely trajectory prediction (establishing accuracy for the deterministic use case), while for ANLL/FNLL we compute likelihoods using their full output distributions (determining performance for probabilistic use cases). For minADE/minFDE, we randomly sample 20 trajectories from each model and compute the ADE/FDE of the best [16].

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. We trained each model until their validation performance stopped improving. Further training details can be found in Appendix E.

A. Lyft Dataset Results

Table II 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 I), 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 (Table I).

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++ (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 class-specific decoder.

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 II, HAICU maintains strong performance at longer time horizons. Additional results per agent class can be found in Appendix D.1.

TABLE III

OUR MODEL SIGNIFICANTLY OUTPERFORMS EXISTING METHODS ON OUR NEW PUP DATASET. RESULTS WITH THE MINADE AND MINFDE METRICS REINFORCE THIS AND CAN BE FOUND IN APPENDIX F. SE = STANDARD ERROR.

PUP	ADE \pm SE		FDE \pm SE (m)				ANLL \pm SE		FNLL \pm SE (nats)		
	Pred. Horizon		3s	1s	2s	3s	3s	1s	2s	3s	
MATS [41]	1.23 \pm 0.21	0.48 \pm 0.12	1.08 \pm 0.19	2.12 \pm 0.35	5.66 \pm 0.57	2.23 \pm 0.65	3.59 \pm 0.54	11.15 \pm 1.02			
Trajectron++ [4]	0.75 \pm 0.02	0.48 \pm 0.01	0.93 \pm 0.02	1.52 \pm 0.03	0.04 \pm 0.05	-0.30 \pm 0.04	0.69 \pm 0.07	1.32 \pm 0.06			
Multi-Head [22]	0.84 \pm 0.06	0.55 \pm 0.04	1.05 \pm 0.08	1.64 \pm 0.16	0.18 \pm 0.09	-0.28 \pm 0.10	0.86 \pm 0.13	1.57 \pm 0.15			
One-Hot	0.69 \pm 0.01	0.42 \pm 9e-3	0.85 \pm 0.02	1.41 \pm 0.03	-0.23 \pm 0.06	-0.59 \pm 0.07	0.44 \pm 0.07	1.08 \pm 0.09			
HAICU (ours)	0.65\pm0.02	0.40\pm0.01	0.79\pm0.02	1.35\pm0.04	-0.35\pm0.06	-0.77\pm0.08	0.32\pm0.07	0.96\pm0.11			

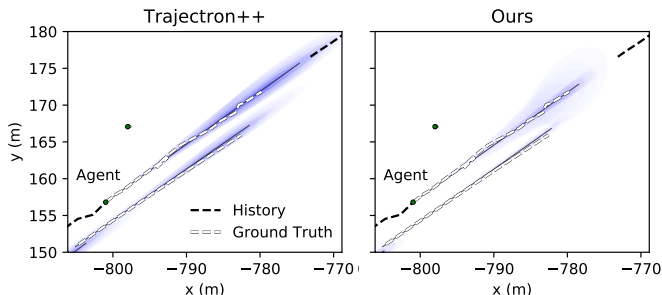


Fig. 6. Our method effectively propagates class probability uncertainty through to its outputs. In this example, the marked agent is a vehicle with high class uncertainty (class probability entropy $S_{\text{probs}} = 2.06$, maximum possible entropy is $\ln(11) \approx 2.40$). Unlike Trajectron++ [4] (left), our method is able to incorporate such information and produces much more accurate predictions (right).

B. PUP Dataset Results

Quantitative Results. Table III 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. Evaluating with minADE/minFDE reinforces this, and is shown in Appendix F.

Notably, in the presence of increased class uncertainty our method now outperforms the one-hot ablated version, 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. The same is true for HAICU’s Top 2 ablation, showing that just adding one more class yields significant benefits over the one-hot ablation. 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 trajectory), leading to a reduction in overall output diversity.

Further, as in Section VI-A, 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 III, our approach maintains its strong performance over longer time horizons. Additional detailed results per agent class can be found in Appendix D.2.

Qualitative Results. Incorporating class uncertainty in trajectory forecasting yields qualitative differences in output behavior. Fig. 6 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 Fig. 7, 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 Fig. 7 (left). In particular, our model predicts that each agent will mostly keep moving along the same heading with some uncertainty. In Fig. 7 (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. Conversely, in Fig. 7 (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 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 Fig. 8, 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

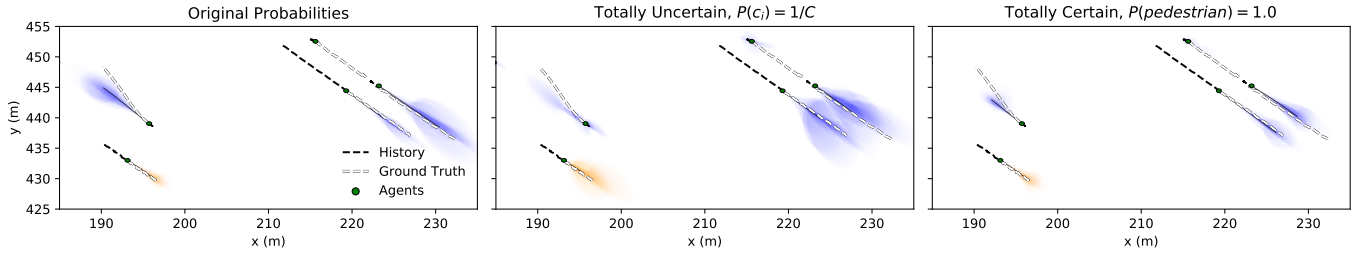


Fig. 7. Our method is able to make counterfactual predictions, i.e., predictions where the input probability distribution is manually modified to produce “what-if” predictions. Color denotes the original class of the agent (orange for pedestrians, blue for vehicles). **Left:** Our model’s predictions with original class probabilities. **Middle:** All agents have fully-uncertain class probabilities. **Right:** All agents have fully-certain pedestrian class probabilities.

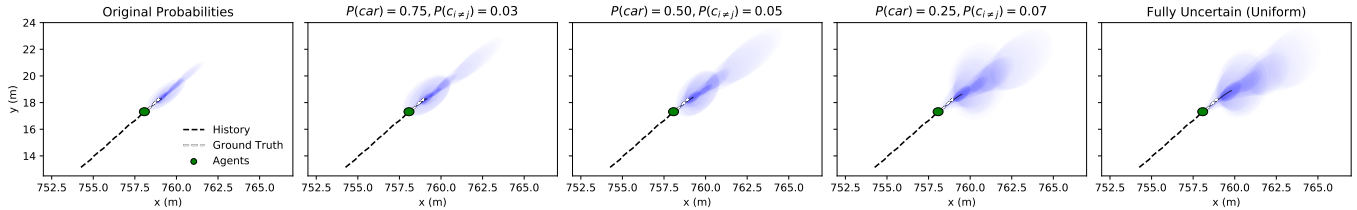


Fig. 8. HAICU’s counterfactual predictions interpolate smoothly across input probabilities.

desirable as it means that our model can smoothly propagate input uncertainty through to its outputs across a wide range of class probability values.

Runtime. On the busiest scene (with 75 agents), HAICU only requires 6.58 GFLOPs to predict all agents’ futures (for reference, this is ~ 1 GFLOP less than executing a forward pass of ResNet-34)². Fig. 9 shows heatmaps visualizing the FLOPs required to run HAICU on the various scene sizes (in terms of node and edge counts) encountered in the PUP dataset. Further, due to our aggressive weight-sharing scheme, HAICU only has 117,389 parameters.

VII. 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. In addition to a more informed representation of uncertainty, our approach also enables new capabilities such as counterfactual predictions, opening up interesting future research in causal reasoning and interpretability for prediction and planning.

REFERENCES

- [1] A. Rudenko, L. Palmieri, M. Herman, K. M. Kitani, D. M. Gavrila, and K. O. Arras, “Human motion trajectory prediction: A survey,” *Int. Journal of Robotics Research*, vol. 39, no. 8, pp. 895–935, 2020.
- [2] R. McAllister, Y. Gal, A. Kendall, M. Van Der Wilk, A. Shah, R. Cipolla, and A. V. Weller, “Concrete problems for autonomous vehicle safety: advantages of Bayesian deep learning,” in *International Joint Conferences on Artificial Intelligence*, 2017.

²See <https://github.com/Lyken17/pytorch-OpCounter> and <https://github.com/sovrasov/flops-counter.pytorch> for other models’ FLOP counts (please note that 1 Multiply-Accumulate (MAC) = 2 FLOPs).

- [3] D. Bhatt, D. Bansal, G. Gupta, H. Lee, K. M. Jatavallabhula, and L. Paull, “Probabilistic object detection: Strengths, weaknesses, opportunities,” in *Workshop on AI for Autonomous Driving at the International Conference on Machine Learning (ICML)*, 2020.
- [4] T. Salzmann, B. Ivanovic, P. Chakravarty, and M. Pavone, “Trajectron++: Dynamically-feasible trajectory forecasting with heterogeneous data,” in *European Conf. on Computer Vision*, 2020.
- [5] J. Houston, G. Zuidhof, L. Bergamini, Y. Ye, A. Jain, S. Omari, V. Iglovikov, and P. Ondruska, “One thousand and one hours: Self-driving motion prediction dataset,” in *Conf. on Robot Learning*, 2020.
- [6] W. Schwarting, J. Alonso-Mora, and D. Rus, “Planning and decision-making for autonomous vehicles,” *Annual Review of Control, Robotics, and Autonomous Systems*, 2018.
- [7] S. Lefèvre, D. Vasquez, and C. Laugier, “A survey on motion prediction and risk assessment for intelligent vehicles,” *ROBOMECH journal*, vol. 1, no. 1, pp. 1–14, 2014.
- [8] Y. Ma, X. Zhu, S. Zhang, R. Yang, W. Wang, and D. Manocha, “Trafficpredict: Trajectory prediction for heterogeneous traffic-agents,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, 2019, pp. 6120–6127.
- [9] 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 *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.
- [10] N. Rhinehart, K. M. Kitani, and P. Vernaza, “R2p2: A reparameterized pushforward policy for diverse, precise generative path forecasting,” in *The European Conference on Computer Vision (ECCV)*, September 2018.
- [11] Y. Chai, B. Sapp, M. Bansal, and D. Anguelov, “Multipath: Multiple probabilistic anchor trajectory hypotheses for behavior prediction,” in *Conference on Robot Learning (CoRL)*, 2019.
- [12] E. Schmerling, K. Leung, W. Vollprecht, and M. Pavone, “Multimodal probabilistic model-based planning for human-robot interaction,” in *Proc. IEEE Conf. on Robotics and Automation*, 2018.
- [13] B. Ivanovic, E. Schmerling, K. Leung, and M. Pavone, “Generative modeling of multimodal multi-human behavior,” in *IEEE/RSJ Int. Conf. on Intelligent Robots & Systems*, 2018.
- [14] B. Ivanovic and M. Pavone, “The Trajectron: Probabilistic multi-agent trajectory modeling with dynamic spatiotemporal graphs,” in *IEEE Int. Conf. on Computer Vision*, 2019.
- [15] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, “Generative adversarial nets,” in *Conf. on Neural Information Processing Systems*, 2014.
- [16] A. Gupta, J. Johnson, F. Li, S. Savarese, and A. Alahi, “Social GAN: Socially acceptable trajectories with generative adversarial networks,” in *IEEE Conf. on Computer Vision and Pattern Recognition*, 2018.
- [17] D. Roy, T. Ishizaka, C. K. Mohan, and A. Fukuda, “Vehicle trajectory prediction at intersections using interaction based generative adversarial networks,” in *IEEE Intelligent Transportation Systems Conference (ITSC)*, 2019, pp. 2318–2323.
- [18] W. Luo, B. Yang, and R. Urtasun, “Fast and furious: Real time end-to-end 3d detection, tracking and motion forecasting with a

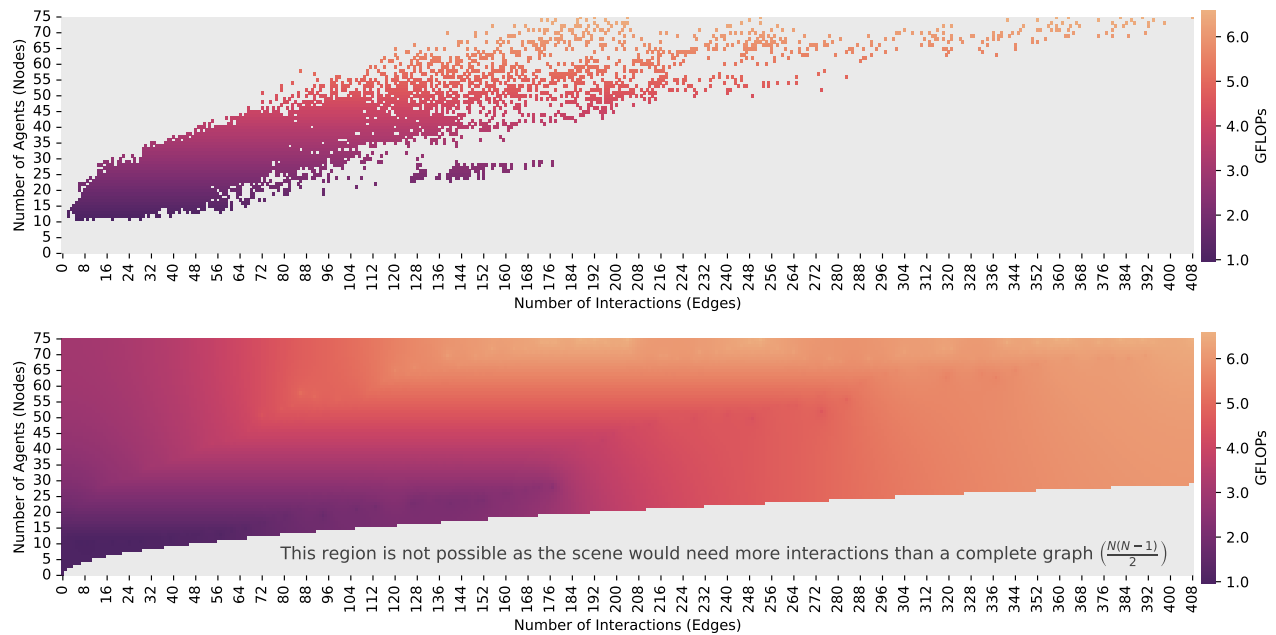


Fig. 9. **Top:** FLOPs required to run HAICU on the problem sizes found in the PUP dataset. **Bottom:** The same, but with an additional optimization-based extrapolation scheme (Laplacian smoothing [42]) to impute values for configurations that were not encountered in the PUP dataset.

single convolutional net,” in *Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 3569–3577.

[19] S. Casas, W. Luo, and R. Urtasun, “Intentnet: Learning to predict intention from raw sensor data,” in *Conference on Robot Learning (CoRL)*. PMLR, 2018, pp. 947–956.

[20] S. Casas, C. Gulino, R. Liao, and R. Urtasun, “SpAGNN: Spatially-aware graph neural networks for relational behavior forecasting from sensor data,” in *International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 9491–9497.

[21] S. Casas, C. Gulino, S. Suo, K. Luo, R. Liao, and R. Urtasun, “Implicit latent variable model for scene-consistent motion forecasting,” in *European Conference on Computer Vision (ECCV)*, 2020.

[22] N. Djuric, H. Cui, Z. Su, S. Wu, H. Wang, F.-C. Chou, L. S. Martin, S. Feng, R. Hu, Y. Xu, A. Dayan, S. Zhang, B. C. Becker, G. P. Meyer, C. Vallespi-Gonzalez, and C. K. Wellington, “MultiXNet: Multiclass multistage multimodal motion prediction,” in *IEEE Intelligent Vehicles Symposium (IV)*, 2021.

[23] W. A. Wright, “Bayesian approach to neural-network modeling with input uncertainty,” *IEEE Transactions on Neural Networks*, vol. 10, no. 6, pp. 1261–1270, 1999.

[24] H. Wang, X. Shi, and D.-Y. Yeung, “Natural-parameter networks: A class of probabilistic neural networks,” in *Conf. on Neural Information Processing Systems*, 2016.

[25] R. F. Astudillo and J. P. S. Neto, “Propagation of uncertainty through multilayer perceptrons for robust automatic speech recognition,” in *Conf. of the Int. Speech Communication Association*, 2011.

[26] N. Rhinehart, R. McAllister, K. Kitani, and S. Levine, “PRECOG: Prediction conditioned on goals in visual multi-agent settings,” in *International Conference on Computer Vision (ICCV)*, October 2019.

[27] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” *Neural Computation*, 1997.

[28] P. W. Battaglia, R. Pascanu, M. Lai, D. Rezende, and K. Kavukcuoglu, “Interaction networks for learning about objects, relations and physics,” in *Conf. on Neural Information Processing Systems*, 2016.

[29] A. Jain, A. R. Zamir, S. Savarese, and A. Saxena, “Structural-RNN: Deep learning on spatio-temporal graphs,” in *IEEE Conf. on Computer Vision and Pattern Recognition*, 2016.

[30] K. Sohn, H. Lee, and X. Yan, “Learning structured output representation using deep conditional generative models,” in *Conf. on Neural Information Processing Systems*, 2015.

[31] M. Itkina, B. Ivanovic, R. Senanayake, M. J. Kochenderfer, and M. Pavone, “Evidential sparsification of multimodal latent spaces in conditional variational autoencoders,” in *Conf. on Neural Information Processing Systems*, 2020.

[32] K. Cho, B. van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, “Learning phrase representations using rnn encoder-decoder for statistical machine translation,” in *Proc. of Conf. on Empirical Methods in Natural Language Processing*, 2014, pp. 1724–1734.

[33] R. E. Kalman, “A new approach to linear filtering and prediction problems,” *ASME Journal of Basic Engineering*, vol. 82, pp. 35–45, 1960.

[34] S. Thrun, W. Burgard, and D. Fox, “The extended Kalman filter,” in *Probabilistic Robotics*. MIT Press, 2005, pp. 54–64.

[35] S. M. LaValle, “Better unicycle models,” in *Planning Algorithms*. Cambridge Univ. Press, 2006, pp. 743–743.

[36] B. Paden, M. Cáp, S. Z. Yong, D. Yershov, and E. Frazzoli, “A survey of motion planning and control techniques for self-driving urban vehicles,” *IEEE Transactions on Intelligent Vehicles*, vol. 1, no. 1, pp. 33–55, 2016.

[37] S. Zhao, J. Song, and S. Ermon, “InfoVAE: Balancing learning and inference in variational autoencoders,” in *Proc. AAAI Conf. on Artificial Intelligence*, 2019.

[38] C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger, “On calibration of modern neural networks,” in *Int. Conf. on Machine Learning*, 2017.

[39] A. Geiger, P. Lenz, and R. Urtasun, “Are we ready for autonomous driving? the kitti vision benchmark suite,” in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[40] H. Caesar, V. Bankiti, A. H. Lang, S. Vora, V. E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan, and O. Beijbom, “nuScenes: A multimodal dataset for autonomous driving,” 2019.

[41] B. Ivanovic, A. Elhafi, G. Rosman, A. Gaidon, and M. Pavone, “MATS: An interpretable trajectory forecasting representation for planning and control,” in *Conf. on Robot Learning*, 2020.

[42] D. Hallac, J. Leskovec, and S. Boyd, “Network lasso: Clustering and optimization in large graphs,” in *ACM Int. Conf. on Knowledge Discovery and Data Mining*, 2015.

[43] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, “Automatic differentiation in PyTorch,” in *Conf. on Neural Information Processing Systems - Autodiff Workshop*, 2017.

[44] S. R. Bowman, L. Vilnis, O. Vinyals, A. M. Dai, R. Jozefowicz, and S. Bengio, “Generating sentences from a continuous space,” in *Proc. Annual Meeting of the Association for Computational Linguistics*, 2015.

[45] C. Schöller, V. Aravantinos, F. Lay, and A. Knoll, “What the constant velocity model can teach us about pedestrian motion prediction,” *IEEE Robotics and Automation Letters*, 2020.

[46] E. Jang, S. Gu, and B. Poole, “Categorical reparameterization with gumbel-softmax,” in *Int. Conf. on Learning Representations*, 2017.

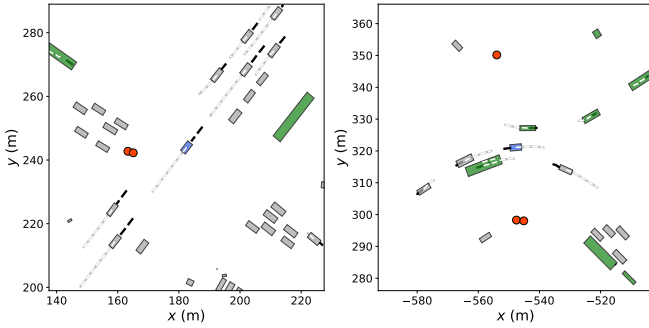


Fig. 10. Additional example scenes from our new PUP dataset.

APPENDIX

A. Additional PUP Scene Visualizations

Additional scenes from our PUP dataset are visualized in Fig. 10.

B. Detailed Lyft Dataset Statistics

1) *Class Switching Histogram*: As mentioned in the main text, we find that 2.1% of all agents in the Lyft Level 5 dataset [5] experience class-switching, i.e., their highest probability class changes at least once during observation. Fig. 11 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.

2) *Examples of Class Switching*: Fig. 12 visualizes a scenario where a nearby car is misclassified as a pedestrian in the middle of an intersection. Another example is visualized in Fig. 13 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 from sensor view.

3) *Types of Class Switching*: Fig. 14 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).

4) *Probability of the Most-Likely Class*: Fig. 15 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.

C. Detailed PUP Dataset Statistics

Note that 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.

D. Additional Results

1) *Lyft Dataset*: Table IV shows the per-class performance of our method, its ablations, and baselines on the Lyft dataset. As in Table II, our method generally performs the best across all classes and is similar to the one-hot ablation

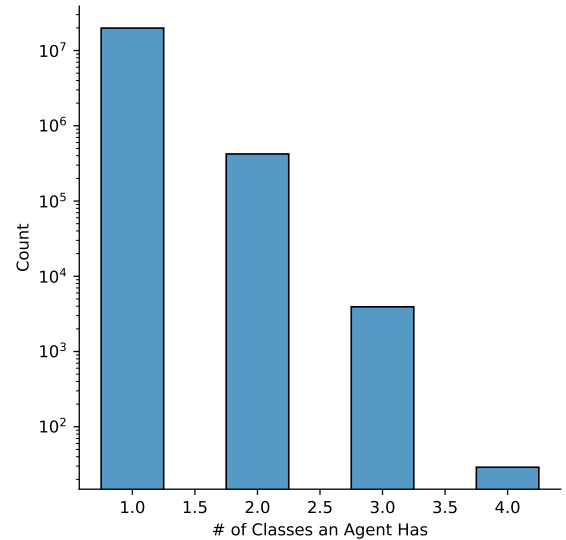


Fig. 11. A histogram of the number of most-likely classes an agent has. For example, if an agent has 4 classes then it experiences at least three class switches between 4 distinct classes during observation

due to the abundance of one-hot class probabilities in the Lyft dataset.

2) *PUP Dataset*: Table VI 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.

Note that we could not evaluate MATS [41] 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.

E. Additional Training Information

All methods were implemented in PyTorch [43] on a computer running Ubuntu 18.04 containing an AMD Ryzen 1800X CPU and two NVIDIA GTX 1080 Ti GPUs.

We anneal the β hyperparameter in Eq. (1) following an increasing sigmoid [44]. 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 $q_\phi(z | \mathbf{x}, \mathbf{y})$ to $p_\theta(z | \mathbf{x}, \hat{\mathbf{c}})$.

To avoid overfitting to environment-specific characteristics, such as the general directions that agents move, we augment the data from each scene with rotation [45]. 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 [4]. We apply this augmentation to autonomous driving datasets because most of them are recorded in cities whose streets are roughly orthogonal and

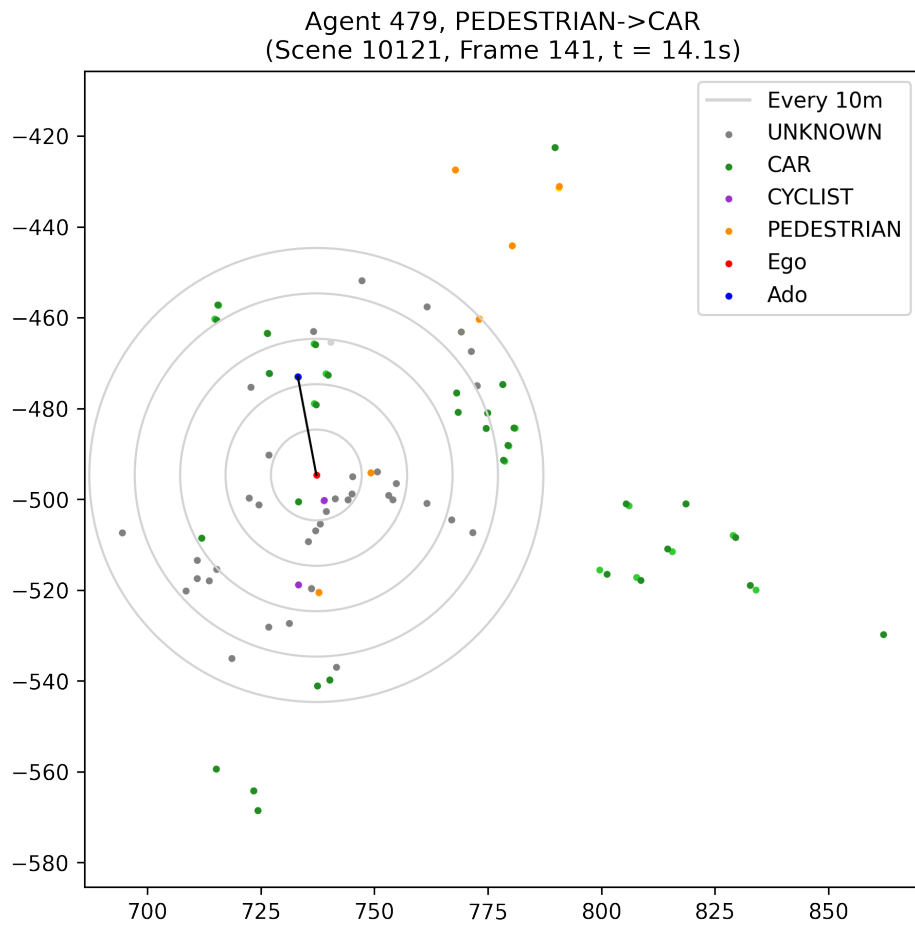


Fig. 12. A scenario in the Lyft Level 5 dataset [5] where a nearby car is misclassified as a pedestrian in the middle of an intersection. In this example, the misclassified agent (in blue) is only around 20m away from the ego-vehicle (in red). The solid black line indicates the distance between the two 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.

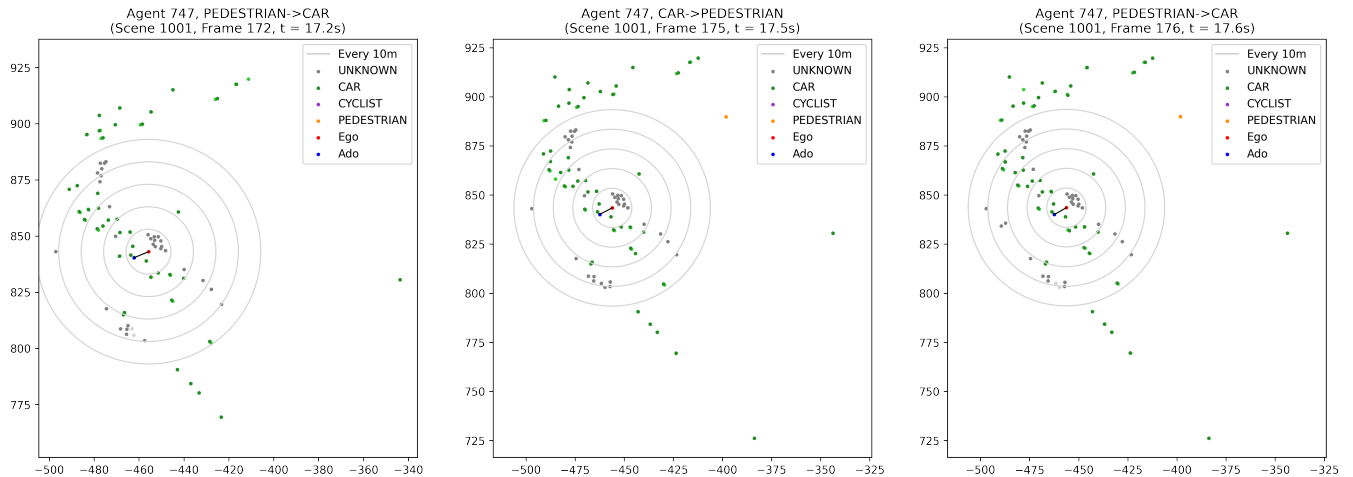


Fig. 13. A scenario in the Lyft Level 5 dataset [5] where a pedestrian next to the ego-vehicle is misclassified as a car while waiting to cross the street. In this example, the misclassified agent (in blue) is less than 10m away from the ego-vehicle (in red). The solid black line indicates the distance 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.

TABLE IV
PER-CLASS PERFORMANCE ON THE LYFT LEVEL 5 DATASET [5]. LOWER IS BETTER, BOLD IS BEST.

Lyft – Car	ADE	FDE (m)			ANLL	FNLL (nats)		
Pred. Horizon	3s	1s	2s	3s	3s	1s	2s	3s
MATS [41]	2.33	0.46	2.21	6.17	4.75	0.85	4.10	13.67
Trajectron++ [4]	0.77	0.43	0.97	1.74	-0.19	-0.63	0.75	1.71
Multi-Head [22]	0.80	0.43	1.00	1.81	-0.20	-0.63	0.73	1.71
One-Hot	0.76	0.42	0.95	1.71	-0.66	-0.91	0.24	1.09
HAICU (ours)	0.75	0.40	0.94	1.71	-0.64	-0.91	0.27	1.17
Lyft – Cyclist	ADE	FDE (m)			ANLL	FNLL (nats)		
Pred. Horizon	3s	1s	2s	3s	3s	1s	2s	3s
MATS [41]	1.14	0.33	1.25	2.45	3.20	0.16	2.59	11.14
Trajectron++ [4]	0.64	0.29	0.81	1.58	-0.45	-0.93	0.66	1.70
Multi-Head [22]	0.55	0.28	0.71	1.26	-0.67	-1.08	0.38	1.33
One-Hot	0.51	0.27	0.66	1.17	-1.07	-1.34	-0.04	0.78
HAICU (ours)	0.48	0.25	0.62	1.12	-1.07	-1.37	-0.04	0.78
Lyft – Pedestrian	ADE	FDE (m)			ANLL	FNLL (nats)		
Pred. Horizon	3s	1s	2s	3s	3s	1s	2s	3s
MATS [41]	0.68	0.28	0.70	1.42	3.29	0.06	1.61	10.69
Trajectron++ [4]	0.30	0.18	0.39	0.62	-1.14	-1.40	-0.09	0.74
Multi-Head [22]	0.30	0.18	0.39	0.62	-1.36	-1.64	-0.30	0.53
One-Hot	0.30	0.17	0.38	0.63	-1.33	-1.60	-0.24	0.64
HAICU (ours)	0.30	0.17	0.38	0.65	-1.31	-1.64	-0.30	0.53
Lyft – Unknown	ADE	FDE (m)			ANLL	FNLL (nats)		
Pred. Horizon	3s	1s	2s	3s	3s	1s	2s	3s
MATS [41]	0.71	0.28	0.71	1.57	3.87	-0.09	1.57	12.76
Trajectron++ [4]	0.18	0.15	0.22	0.28	-2.43	-2.43	-1.85	-1.42
Multi-Head [22]	0.17	0.15	0.21	0.27	-2.38	-2.40	-1.83	-1.39
One-Hot	0.18	0.16	0.22	0.28	-2.38	-2.38	-1.83	-1.40
HAICU (ours)	0.18	0.15	0.22	0.28	-2.38	-2.38	-1.81	-1.38

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.

Finally, the Gumbel-Softmax reparameterization [46] is *not* used to backpropagate through the Categorical latent variable z because z is not sampled during training. Instead, the first term of Eq. (1) is directly enumerated and summed since the latent space has only $|Z| = 25$ discrete elements.

F. minADE and minFDE Results

Table V summarizes evaluation on the Lyft and PUP datasets with the minADE and minFDE metrics (over 20 samples).

TABLE V

EVALUATING WITH THE MINADE AND MINFDE METRICS (OVER 20 SAMPLES) SHOWS THAT HAICU STILL OUTPERFORMS EXISTING APPROACHES IN THE FACE OF UNCERTAINTY (PUP DATASET). FURTHER, AS EXPECTED, PERFORMANCE IS SIMILAR ON THE LYFT DATA (WHICH HAS VIRTUALLY NO CLASS UNCERTAINTY). LOWER IS BETTER, BOLD IS BEST. SE = STANDARD ERROR.

Lyft Level 5 [5]	minADE \pm SE		minFDE \pm SE (m)		
Pred. Horizon	3s	1s	2s	3s	
Trajectron++ [4]	0.23 \pm 3e-3	0.10 \pm 1e-3	0.23 \pm 3e-3	0.40 \pm 6e-3	
Multi-Head [22]	0.24 \pm 3e-3	0.10 \pm 1e-3	0.22 \pm 3e-3	0.38 \pm 6e-3	
One-Hot	0.26 \pm 3e-3	0.09 \pm 1e-3	0.20 \pm 3e-3	0.36 \pm 6e-3	
HAICU (ours)	0.26 \pm 3e-3	0.09 \pm 1e-3	0.21 \pm 3e-3	0.38 \pm 6e-3	

PUP	minADE \pm SE		minFDE \pm SE (m)		
Pred. Horizon	3s	1s	2s	3s	
Trajectron++ [4]	0.36 \pm 0.02	0.21 \pm 0.02	0.37 \pm 0.02	0.52 \pm 0.04	
Multi-Head [22]	0.37 \pm 0.01	0.21 \pm 0.01	0.33 \pm 0.02	0.53 \pm 0.03	
One-Hot	0.40 \pm 0.02	0.17 \pm 9e-3	0.29 \pm 0.02	0.53 \pm 0.05	
HAICU (ours)	0.35 \pm 0.01	0.15 \pm 9e-3	0.29 \pm 0.02	0.48 \pm 0.03	

TABLE VI
PER-CLASS PERFORMANCE ON OUR PUP DATASET. LOWER IS BETTER, BOLD IS BEST.

PUP – Bicycle		ADE		FDE (m)			ANLL		FNLL (nats)		
Pred. Horizon		3s	1s	2s	3s	3s	1s	2s	3s		
MATS [41]		2.33	0.76	2.03	4.20	7.02	3.54	5.81	11.70		
Trajectron++ [4]		0.61	0.35	0.74	1.37	-0.63	-1.01	-0.08	0.60		
Multi-Head [22]		0.71	0.45	0.88	1.42	-0.53	-0.95	0.06	0.81		
One-Hot		0.56	0.32	0.69	1.23	-0.99	-1.40	-0.41	0.31		
HAICU (ours)		0.54	0.33	0.66	1.13	-0.98	-1.41	-0.38	0.32		
PUP – Car		ADE		FDE (m)			ANLL		FNLL (nats)		
Pred. Horizon		3s	1s	2s	3s	3s	1s	2s	3s		
MATS [41]		1.01	0.42	0.89	1.73	5.89	1.57	2.63	13.47		
Trajectron++ [4]		0.57	0.37	0.68	1.12	-1.10	-1.39	-0.73	-0.14		
Multi-Head [22]		0.70	0.50	0.86	1.31	-0.98	-1.34	-0.55	0.15		
One-Hot		0.55	0.36	0.65	1.10	-1.29	-1.58	-0.90	-0.27		
HAICU (ours)		0.54	0.36	0.65	1.09	-1.31	-1.60	-0.92	-0.29		
PUP – Largevehicle		ADE		FDE (m)			ANLL		FNLL (nats)		
Pred. Horizon		3s	1s	2s	3s	3s	1s	2s	3s		
MATS [41]		1.42	0.62	1.27	2.39	8.77	5.02	5.83	15.46		
Trajectron++ [4]		0.88	0.66	1.04	1.54	0.31	-0.03	0.65	1.27		
Multi-Head [22]		1.07	0.83	1.28	1.80	0.42	0.03	0.82	1.51		
One-Hot		0.85	0.62	1.00	1.53	0.03	-0.27	0.43	1.07		
HAICU (ours)		0.88	0.63	1.03	1.60	0.02	-0.29	0.42	1.07		
PUP – Motorcycle		ADE		FDE (m)			ANLL		FNLL (nats)		
Pred. Horizon		3s	1s	2s	3s	3s	1s	2s	3s		
MATS [41]		0.59	0.28	0.51	0.98	4.14	0.74	1.34	10.34		
Trajectron++ [4]		0.52	0.36	0.64	0.96	-0.57	-0.98	-0.10	0.49		
Multi-Head [22]		0.53	0.34	0.63	1.05	-0.43	-0.92	0.13	0.92		
One-Hot		0.43	0.27	0.51	0.86	-1.02	-1.41	-0.52	0.07		
HAICU (ours)		0.47	0.29	0.56	0.98	-0.94	-1.36	-0.43	0.21		
PUP – Pedestrian		ADE		FDE (m)			ANLL		FNLL (nats)		
Pred. Horizon		3s	1s	2s	3s	3s	1s	2s	3s		
MATS [41]		0.79	0.32	0.72	1.32	2.47	0.27	2.35	4.79		
Trajectron++ [4]		0.50	0.32	0.62	0.97	0.36	-0.17	1.13	2.05		
Multi-Head [22]		0.65	0.41	0.82	1.29	0.66	0.14	1.45	2.31		
One-Hot		0.58	0.36	0.72	1.18	0.41	-0.11	1.18	2.06		
HAICU (ours)		0.62	0.38	0.77	1.24	0.48	-0.04	1.25	2.15		
PUP – Unknown		ADE		FDE (m)			ANLL		FNLL (nats)		
Pred. Horizon		3s	1s	2s	3s	3s	1s	2s	3s		
Trajectron++ [4]		1.44	0.79	1.86	3.14	1.89	1.80	3.28	3.65		
Multi-Head [22]		1.41	0.79	1.83	2.97	1.96	1.32	3.28	3.74		
One-Hot		1.16	0.59	1.51	2.58	1.46	1.25	2.86	3.26		
HAICU (ours)		0.86	0.40	1.09	2.10	0.64	0.07	2.00	2.32		



Fig. 14. The 15 most common class switches (out of 45 total) in the Lyft Level 5 dataset [5], 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.

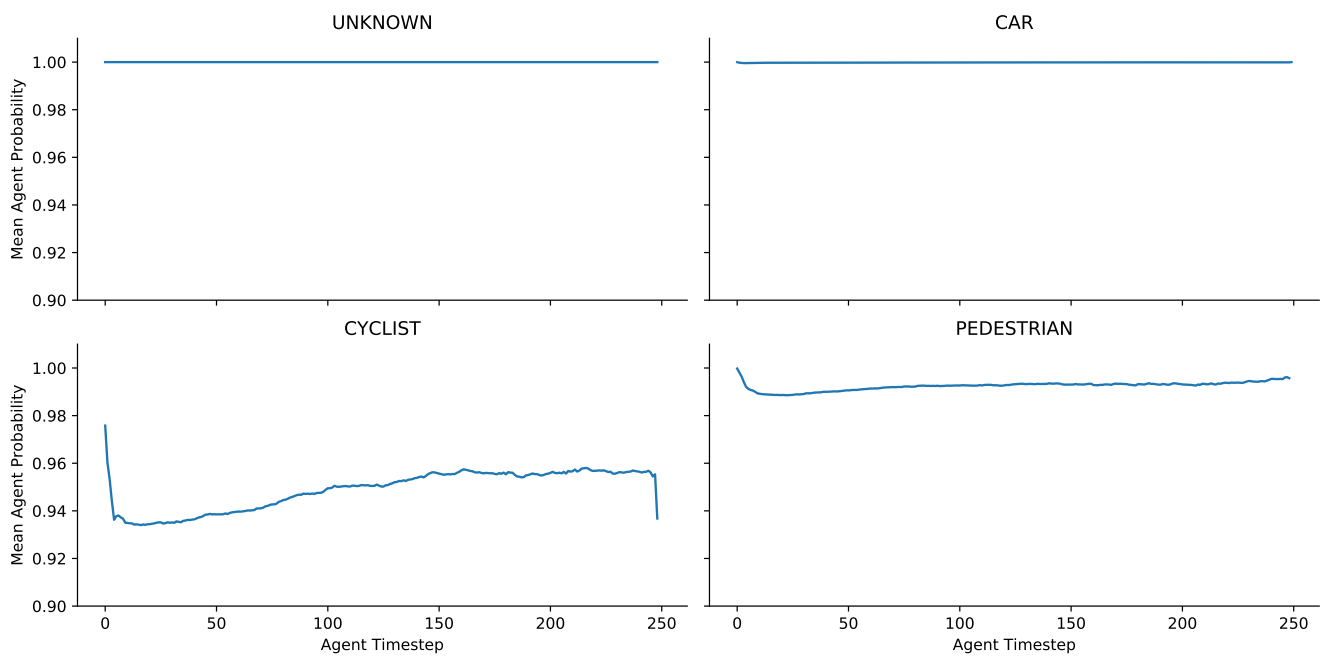


Fig. 15. The average probability of the agent's most-likely class over time for each class in the Lyft Level 5 dataset [5].