Disagreement-Regularized Imitation of Complex Multi-Agent Interactions

Nate Gruver
Stanford University
ngruver@cs.stanford.edu

Jiaming Song
Stanford University
tsong@cs.stanford.edu

Stefano Ermon
Stanford University
ermon@cs.stanford.edu

Abstract

Learning models of multi-agent behavior remains challenging in environments with a large, varying number of agents and complex interactions. To this end, we propose a new algorithm named disagreement-regularized imitation of complex multi-agent interactions (DIMMI) for learning multi-modal distributions over multi-agent trajectories. DIMMI uses a deep latent variable model to capture highly-correlated behavior of a large number of agents (5-50) and disagreement-regularized imitation learning to combat covariate shift. We demonstrate our method on a large collection of multi-agent driving data ($\sim$160 hrs of driving) comprising a wide variety of scenarios. We report improvements of 5-50% over alternative methods when comparing rolled out trajectories to ground truth on common metrics like displacement error and state occupancy measures. We also demonstrate the superior performance of our new method qualitatively by examining interpolations in the latent space, which captures high-level sources of variance such as cautious and aggressive driving styles.

1 Introduction

Understanding multi-agent interactions is fundamental to automated decision making in social settings [23]. In this work we focus on settings with a large, varying number of agents that exhibit complex relationships with each other and their environment and whose intentions are unknown. Many problems of practical significance fit this description, for example subjects of biological experiments, pedestrians or vehicles in public spaces, and even players in MMORPGs.

There are numerous challenges to modeling behavior in these settings, but one is the most salient. When describing a single agent with unknown intentions, a model must capture different responses to the same observation resulting from different preferences. When multiple agents interact, however, each agent can have a variety of responses to the chosen actions of each other agent. This results in an exponential growth of possible outcomes that must be well-represented by the model.

Behavior modeling is often successfully posed as problem of *imitation learning* (IL), in which we learn a distribution of actions conditioned on an agent's observation, called a *policy*. In complex environments, learning this distribution often requires inference in a high-dimensional observation space, which by itself poses a challenge. Unlike other probabilistic models, however, the sequential setting means that resulting policy must be especially robust to covariate shift, as small errors can lead agents to poorly covered areas of the data distribution. In multi-agent settings, the problem of covariate shift can be even more pronounced, as agents also react to each other's actions.

Generative adversarial imitation learning (GAIL) [11] offers one solution to the problem of covariate shift. Using a learned classifier for real and generated trajectories, the policy can be corrected by decreasing the likelihood of actions taken within the environment deemed unlikely by the classifier. GAIL has been extended to capture multimodality [14, 27] and to settings with a small, fixed number of agents [24] but is fundamentally difficult to optimize due to its combination of adversarial training and model-free RL methods.

Machine Learning for Autonomous Driving Workshop at the 34th Conference on Neural Information Processing Systems (NeurIPS 2020), Vancouver, Canada.

The fundamental advantage of GAIL, its ability to identify and correct out-of-distribution (OOD) trajectories, does not, however, necessitate adversarial training. More recently, appealing alternatives based on other methods of identifying OOD examples have emerged [5].

In this work we examine how deep latent variable models can be used in tandem with OOD estimates from model ensembles to learn multimodal distributions over multi-agent behavior. This new approach is significantly simpler to train than GAIL-based alternatives and is able to capture diverse preferences without labels. Unlike some previous work in multi-agent IL [24][28], our approach also scales easily to challenging settings with a large and varying number of agents.

In order to demonstrate our proposed technique, we focus on the domain of driving. Within the last two years there has been a proliferation of high-quality, public datasets of driving trajectories obtained from drone footage [30, 3, 13]. Critically, these datasets contain multiagent trajectories with high levels of interaction in a wide range of environments–such as highway on-ramps, roundabouts, and uncontrolled intersections. This abundance of examples provides good coverage for imitation learning and opens the door for the kind of unsupervised pre-training approaches that have driven recent progress in natural language processing [4, 7].

The primary contributions of this work:

1. We propose a novel algorithm for multi-agent imitation learning in complex environments with varying number of agents, partial observability, and latent (unobserved) variables
2. We evaluate it on a large-scale dataset with rich semantics. When evaluated against baseline techniques common in multi-agent driver modeling, our approach shows 5-50% improvement across standard metrics.

We also plan to release the experimental apparatus used here to encourage more research in this area.

2 Related Work

Multi-agent IL Multi-agent GAIL (MA-GAIL) [24] and Multi-agent AIRL (MA-AIRL) [28] are frameworks for imitation and inverse reinforcement learning in Markov games. Unfortunately, these algorithms do not easily scale to real-world scenarios as they designed for settings with a small, fixed number of agents. Attempts have been made to scale MA-GAIL using shared policies [2] and reward shaping [1], but resulting policies are often unreliable when rolled out in the environment.

IL with latent variables In the single-agent environments, latent variables have been used to learn multi-modal policies [14, 27] and to perform meta-learning [29]. Our work most closely parallels [27], which augments GAIL with a VAE-type latent variable model. We, however, opt for a different type of IL and focus on multi-agent settings.

Multi-agent IL with latent variables There are few works examining the use of latent variable models for multi-agent IL in the general setting. Much of the existing work focuses on the setting of driver trajectory modeling and model-based planning for autonomous vehicles. Multiple futures prediction (MFP) [26] uses an EM-type procedure to learn a distribution over future states of a vehicle in multi-agent scenarios, assuming a differentiable observation space. Prediction conditioned on goals (PRECOG) [19] uses a multivariate Gaussian parameterized by an RNN to learn a generative model of agent states with exact likelihoods. Similarly, multiple probabilistic anchor trajectory hypotheses (MultiPath) [6], uses a gaussian mixture model parameterized by a ResNet given a rich multi-channel image as input. Our experimental approach draws on elements of these works, especially [6], but differs from them in its use of policy corrections from multi-agent rollouts. This difference makes the above approaches more sensitive to covariate shift when used as generative models in complex environments.

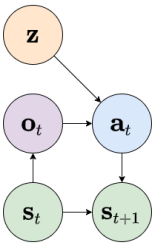
IL regularized by ensembling Model-predictive policy learning with uncertainty regularization (MPUR) [10] learns a pixel-space dynamics model for model-based planning in multi-agent driving scenarios. To capture uncertainty while combating covariate shift, MPUR uses a curiosity penalty derived from an ensemble of latent variable models. Disagreement-regularized imitation learning (DRIL) [5] proposes a similar mechanism in the general single-agent setting, correcting a learned policy based on the variance of ensemble estimates during rollouts in the environment. We adopt the idea of "disagreement-regularization" in our method, but focus on the setting of multi-agent imitation learning. Like MPUR we evaluate our method on multi-agent driving scenarios and employ a deep

latent variable model. Unlike MPUR, however, our method does not require learning a generative model of video, allowing us to easily use high-resolution, multi-channel observations.

3 Problem Definition

We consider the problem of learning a model of the behavior of a *varying number of agents* interacting in a *partially observable* environment from *incomplete* logged data.

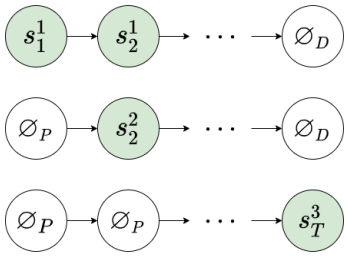
We have up to N agents interacting over T time steps. At each time step, agent i has state (s_t^i, z^i), receives observation o_t^i, and takes action a_t^i. Random variables s_t^i, o_t^i, and a_t^i are known by all agents and to a modeler, while z^i is a latent variable only known to agent i. For example, we might have s_t^i represent the physical position of the agent, o_t^i a rendering of the its surroundings, and z^i the its risk tolerance.



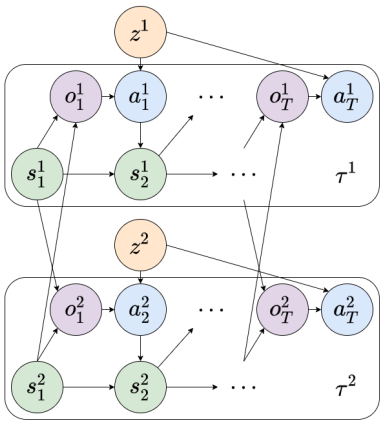
We take the distribution over actions given observations, the joint policy, to be $\boldsymbol{\pi}(\boldsymbol{a}_t|\boldsymbol{o}_t, \boldsymbol{z})$. At each time step, therefore, actions are fully determined by the timestep's observation and the latent variables $\boldsymbol{z}$. The distribution over next states given current states and actions $P(\boldsymbol{s}_{t+1}|\boldsymbol{s}_t, \boldsymbol{a}_t)$, is called the transition probability or dynamics model, and the distribution of observations given states $P(\boldsymbol{o}_t|\boldsymbol{s}_t)$ is called the observation model. A multiagent trajectory $\boldsymbol{\tau}$ is taken to be the sequence of states, observations, and actions for each agent, $\boldsymbol{\tau} = \{(s_t^i, o_t^i, a_t^i)\}_{1:T}^{1:N}$. The probability of a trajectory given unobserved variables $\boldsymbol{z}$ is thus given by

$$P(\boldsymbol{\tau}|\boldsymbol{z}) = \prod_{t=1}^{T} P(\boldsymbol{s}_{t+1}|\boldsymbol{s}_t, \boldsymbol{a}_t)\boldsymbol{\pi}(\boldsymbol{a}_t|\boldsymbol{o}_t, \boldsymbol{z})P(\boldsymbol{o}_t|\boldsymbol{s}_t)P(\boldsymbol{s}_1)$$

Varying number of agents In our formulation, we additionally allow that agents enter and exit the environment at any time. We consider this generalization to capture the common real-world phenomenon of agents entering and exiting the recording area of sensors capturing their states. These changes introduce a "birth-death" cycle in which the number of modelled agents changes over time. One can conceive of these scenarios within the framework above by putting the non-modeled agents of the system in either a "prenatal" state $\varnothing_P$ or a dead state $\varnothing_D$. Agents in $\varnothing_P$ stochastically transition to a first recorded state $s_{t_0}^i$, and agents in $\varnothing_D$ remain there until time T. As some agents enter the environment after others leave, the number of agents increases and decreases stochastically. An example with three agents offset by one time step is shown to the right. The challenges of multi-agent imitation learning are intensified in birth-death systems. Because it is not a closed system, each agent cannot be a assigned a fixed policy a priori. We provide more details and formal description of learning for this setting in the Appendix.



Key assumptions Our focus in this work is modeling policies and thus we assume $P(\boldsymbol{s}_{t+1}|\boldsymbol{s}_t, \boldsymbol{a}_t)$, $P(\boldsymbol{o}_t|\boldsymbol{s}_t)$, and $P(\boldsymbol{s}_1)$ are either known a priori or well-approximated. In environments with well-studied dynamics and sensor models, these assumptions are reasonable. We further assume fully factorized environmental dynamics $P(\boldsymbol{s}_{t+1}|\boldsymbol{s}_t, \boldsymbol{a}_t) = \prod_i P(s_{t+1}^i|s_t^i, a_t^i)$, and conditional independence of the observation model, $P(\boldsymbol{o}_t|\boldsymbol{s}_t) = \prod_i P(o_t^i|\boldsymbol{s}_t)$. We additionally assume that all latent variables z^i have an isotropic gaussian distribution, $P(\boldsymbol{z}) = \mathcal{N}(\mathbf{0}, \mathbf{1})$ and do not change over time.



Learning from incomplete data Given a set of expert trajectories $\mathcal{D}$ sampled from an expert policy $\boldsymbol{\pi}_E$, the goal is to learn a parameterized policy distribution $\boldsymbol{\pi}_\theta(\boldsymbol{a}_t|\boldsymbol{o}_t, \boldsymbol{z})$ using trajectories from $\mathcal{D}$ that matches $\boldsymbol{\pi}_E$ *without knowledge of* z^i.

Directly learning a joint distribution over all agents actions is often infeasible because of difficulties in representation and sparsity. We therefore factorize the policy over agents at each timestep,

$\boldsymbol{\pi}_\theta(\boldsymbol{a}_t|\boldsymbol{o}_t, \boldsymbol{z}) = \prod_i \pi_\theta(a_t^i|o_t^i, z^i)$. Given all our independence assumptions, the probability of a trajectory under the model becomes

$$P(\boldsymbol{\tau}|\boldsymbol{z}) = \prod_{t=1}^{T}\prod_{i=1}^{N} P(s_{t+1}^i|s_t^i, a_t^i)\pi_\theta(a_t^i|o_t^i, z^i)P(o_t^i|\boldsymbol{s}_t)P(\boldsymbol{s}_1)$$

4 Method

Behavior cloning To learn the parameters of the policy from logged data, the most immediate solution is maximum likelihood estimation (MLE), also known as behavior cloning (BC) in the setting of IL. With known assignments to $\boldsymbol{z}$, the optimal parameters are solutions to the objective

$$\max_\theta \mathbb{E}_{\boldsymbol{\tau},\boldsymbol{z}\sim\mathcal{D}}[\log P_\theta(\boldsymbol{\tau}|\boldsymbol{z})]$$

However as assignments to $\boldsymbol{z}$ are unavailable, we can only apply MLE by marginalizing out $\boldsymbol{z}$:

$$\max_\theta \mathbb{E}_{\boldsymbol{\tau}\sim\boldsymbol{\pi}_E}[\log P_\theta(\boldsymbol{\tau})], \quad \log P_\theta(\boldsymbol{\tau}) = \log \int_{\boldsymbol{z}} P_\theta(\boldsymbol{\tau}|\boldsymbol{z})P(\boldsymbol{z})\,\mathrm{d}\boldsymbol{z}$$

On its own, $\log P_\theta(\boldsymbol{\tau})$ is intractable as the integral inside the log cannot be computed analytically. A common approach, which we adopt here, is to instead optimize a lower bound on the log probability by introducing a variational approximation of the posterior $Q_\phi(\boldsymbol{z}|\boldsymbol{o}_{1:T})$ [12]:

$$\log P_\theta(\boldsymbol{\tau}) \geq \mathbb{E}_{\boldsymbol{z}\sim Q_\phi(\boldsymbol{z}|\boldsymbol{o}_{1:T})}[\log P_\theta(\boldsymbol{\tau}|\boldsymbol{z}) - D_{\mathrm{KL}}(Q_\phi(\boldsymbol{z}|\boldsymbol{o}_{1:T})\|P(\boldsymbol{z}))]$$

Although this modified version of BC is able to learn a distribution over the possible values of $\boldsymbol{z}$ and thereby generate samples from the distribution $P(\boldsymbol{\tau})$, BC on its own is known to suffer from compounding errors at test time [20]. BC also under-utilizes dynamics models and environment simulators which are assumed known in this problem setting. By extrapolating from the training data to simulated trajectories, useful learning inputs can be created without additional expert supervision. Rolling out additional trajectories can provide a way of learning from states that do not appear in the training data but that are likely at test time, thus mitigating covariate shift and compounding errors.

Disagreement-regularization DRIL [5] offers one way to mitigate compounding errors by leveraging an environmental simulator. DRIL explores likely states and then lowers the probability of reaching any that are judged to be out of distribution (OOD) given the training data. To estimate whether a particular state is OOD, an ensemble of policies, $\{\pi\}_{1:K}$, is learned by bootstrapping [8] behavior cloning on subsets of the training data. The ensemble is then used to derive a regularization cost function via the empirical variance of the action probability, $\pi(a|s)$, over the ensemble

$$C_U(s, a; \{\pi\}_{1:K}) = \frac{1}{K}\sum_{k=1}^{K}\left(\left(\frac{1}{K}\sum_{k=1}^{K}\pi_k(a|s)\right) - \pi_k(a|s)\right)^2$$

The authors of [5] also note that normalizing the uncertainty cost aids convergence, putting

$$C_U^{\text{clip}}(s, a; \{\pi\}_{1:K}, q) = \begin{cases} 1 & \text{if } C_U(s, a; \{\pi\}_{1:K}) > q \\ -1 & \text{else} \end{cases} \tag{1}$$

where q is the β-th percentile of the set

$$\mathcal{C}_\mathcal{D} = \{C_U(s, a; \{\pi\}_{1:K}) \mid (s, a) \sim \mathcal{D}\}$$

and β is taken to be a hyperparameter. A final policy is learned through alternating batches of BC and reinforcement learning minimize C_U^{clip}. Intuitively, as each model in the ensemble is randomly initialized, the ensemble's estimates for points far from the training distribution will be divergent, causing high variance and heavy regularization.

DRIL was originally designed for imitating optimal policies in fully observable settings, and thus we must modify it slightly to account for partial observability. In our setting, we take the uncertainty cost to be

$$C_U(o_t, a_t|z; \{\pi\}_{1:K}) = \frac{1}{K}\sum_{k=1}^{K}\left(\left(\frac{1}{K}\sum_{k=1}^{K}\pi_k(a_t|o_t, z)\right) - \pi_k(a_t|o_t, z)\right)^2 \tag{2}$$

We also use the alternative clipping strategy

$$C_U^{\text{clip}}(o_t, a_t|z; \{\pi\}_{1:K}, q) = \begin{cases} 1 & \text{if } C_U(o_t, a_t|z; \{\pi\}_{1:K}) > q \\ 0 & \text{else} \end{cases} \tag{3}$$

with q taken to be the β-th percentile of the set

$$\mathcal{C}_{\mathcal{D}} = \{C_U(o_t, a_t|z; \{\pi\}_{1:K}) \mid (o_{1:T}, a_{1:T}) \sim \mathcal{D}, z \sim Q_\phi(\cdot|o_{1:T})\} \tag{4}$$

In contrast with equation (1), we use 0 in equation (3), instead of -1, for typical examples because we do not want to implicitly bias the model to prefer longer trajectories.

In the remainder of the section, we show how the BC and disagreement regularization methods above fit together in a unified algorithm for imitation of multi-agent interactions. We call this algorithm disagreement-regularized imitation of complex multi-agent interactions, or DIMMI in short.

Multi-agent disagreement regularization In multi-agent settings, covariate shift problems can compound over time and across agents. Rolling out joint trajectories across all agents and performing disagreement regularization can aid in stabilizing the policy along both axes. In some previous work [26, 10], policies are learned in multi-agent settings, but rollouts are performed with other agents independent of the ego-agent's actions. Learning in this setting introduces false independence assumptions that biases imitation of ground truth behavior.

To apply our modified formulation of DRIL to multi-agent rollouts, we have to generalize the cost function in Equation 2. In principle, learning a centralized $C_U(\boldsymbol{o}_t, \boldsymbol{a}_t|\boldsymbol{z})$, would provide the clearest reward signal with no ambiguity surrounding credit assignment between agents, as in multi-agent actor-critic methods [15]. However, as our target settings are characterized by large N, learning and representation in such a model would be very challenging. Instead, we opt for a decentralized approach, learning a shared but single-agent $C_U(o_t^i, a_t^i|z^i)$. Given an assignment of $\boldsymbol{z}$, the policy regularizer's objective thus becomes

$$\min_\theta \mathbb{E}_{\boldsymbol{\tau} \sim \pi_\theta(\cdot|\cdot, \boldsymbol{z})} \left[\sum_{i=1}^{N} C(\tau^i|z^i; \{\pi\}_{1:K}, q) \right], \quad C(\tau^i|z^i) = \sum_{t=1}^{T} C_U^{\text{clip}}(o_t^i, a_t^i|z^i; \{\pi\}_{1:K}, q)$$

where trajectories $\boldsymbol{\tau}$ are rolled out using the policy and known dynamics/observation models. We descend on the objective using a Monte-Carlo approximation of the gradient on a batch of $\{\tau^i\}$ (such as using REINFORCE [25] and PPO [21]).

DIMMI In our proposed method, DIMMI, we improve latent variable BC with multi-agent disagreement regularization. To accomplish this synthesis, we must learn an ensemble of policies, $\{\pi_{\omega_1}(\cdot|o, z), ..., \pi_{\omega_k}(\cdot|o, z)\}$, to be used for regularization in addition to the final policy, $\pi_\theta(\cdot|o, z)$. All of these models must share an inference model, Q_ϕ, so that different models for $P(\boldsymbol{\tau}|\boldsymbol{z})$ are conditioned on the same values. To this end, we break the training procedure into three phases (full psuedocode shown in Algorithm 1):

Learning the posterior First we must learn a shared posterior inference model, Q_ϕ, that can be used to train an ensemble of the z^i-conditioned policies. To this end, we train a model with behavior cloning (BC) on the entire dataset, minimizing the loss

$$\mathcal{L}_{\text{BC}}(\theta, \phi) = -\mathbb{E}_{\boldsymbol{z} \sim Q_\phi(\boldsymbol{z}|\boldsymbol{o}_{1:T}), \boldsymbol{\tau} \sim \pi_E(\cdot|\boldsymbol{z})} \left[\log P_\theta(\boldsymbol{\tau}|\boldsymbol{z}) - D_{\text{KL}}(Q_\phi(\boldsymbol{z}|\boldsymbol{\tau}) \| P(\boldsymbol{z})) \right] \tag{5}$$

Learning the ensemble Sample K subsets of the data, $\{\mathcal{D}_k\}$, with replacement such that $|\mathcal{D}_k| = |\mathcal{D}|$. Train an ensemble $\{\pi_{\omega_1}, ..., \pi_{\omega_K}\}$, by training each π_{ω_k} to convergence on $\min_{\omega_k, \phi} \mathcal{L}_{\text{BC}}(\omega_k, \phi)$. Calculate q as the β-th percentile of $\mathcal{C}_{\mathcal{D}}$ (Equation (4)). Take

$$C_{\omega, q}(\tau^i|z^i) = \sum_{t=1}^{T} C_U^{\text{clip}}(o_t^i, a_t^i|z^i; \{\pi_{\omega_1}, ..., \pi_{\omega_K}\}, q)$$

Learning the final policy To learn the final policy, we train across the entire dataset using alternating phases of minimization of $\mathcal{L}_{\text{BC}}$ and the disagreement-regularization loss

$$\mathcal{L}_{\text{DR}}(\theta; \omega, q) = \mathbb{E}_{\boldsymbol{z} \sim P(\boldsymbol{z}), \boldsymbol{\tau} \sim \pi_\theta(\cdot|\boldsymbol{z})} \left[\sum_{i=1}^{N} C_{\omega, q}(\tau^i|z^i) \right] \tag{6}$$

These alternating phases ensure every batch of BC is properly regularized as training proceeds.

Algorithm 1 DIMMI: Disagreement-regularized Imitation of Multi-Agent Interactions

Input: $\mathcal{D} = \{\boldsymbol{\tau}\}, \boldsymbol{\tau} \sim \pi_E$,
Output: $\pi_\theta(a_t^i | o_t^i, z^i)$

Train π_θ, Q_ϕ to convergence on $\mathcal{D}$, $\min_{\theta,\phi} \mathcal{L}_{\text{BC}}(\theta, \phi)$ from Eq.(5)

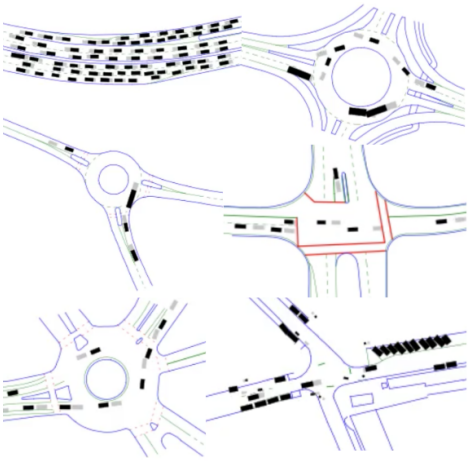
for $k = 1{:}K$ **do**
  Sample $\mathcal{D}_k$ from $\mathcal{D}$ with replacement, with $|\mathcal{D}_k| = |\mathcal{D}|$.
  Train π_{ω_k} on $\mathcal{D}_k$, $\min_{\omega_k} \mathcal{L}_{\text{BC}}(\omega_k, \phi)$ from Eq.(5)
Calculate q from the β-percentile according to (4)

for $i = 1{:}N_{\text{epoch}}$ **do**
  Sample batch of $\boldsymbol{\tau}_E \sim \mathcal{D}, \boldsymbol{z} \sim Q_\phi(\cdot | \boldsymbol{o}_{1:T})$
  Train π_θ on $\{\boldsymbol{\tau}_E\}$, $\min_\theta \mathcal{L}_{\text{BC}}(\theta, \phi)$ from Eq.(5)
  Sample batch $\boldsymbol{\tau}_R \sim \pi_\theta(\cdot \mid \boldsymbol{s}, z^i, s_{t'}^i)$, in $\mathcal{E}$
  Train π_θ on $\{\boldsymbol{\tau}_R\}$, $\min_\theta \mathcal{L}_{\text{DR}}(\theta; \omega, q)$ from Eq.(6)

5 Experiment

5.1 Setup

Large multitask dataset We form our train and test set from the combined trajectories of the INTERACTION [30], inD [3], and rounD [13] datasets. We chose not to include the widely used NGSIM dataset because of well-known issues with data quality [18, 17]. A selection of representative scenarios from our aggregate dataset are shown to the right. All together, the data comprise more than 6 million (state, action) pairs or approximately 160 hours of driving. We initialize our model by training jointly across all scenarios with a vanilla MLE objective (without $\boldsymbol{z}$). Given the diversity of tasks represented in the training data, this pre-training phase can be viewed as an implicit form of multitask learning over scenarios. As some scenarios are over-represented in the aggregated training data, performance can be improved by fine-tuning on an individual scenario.



Goal states In the setting of multiagent driving, we posit there are two largely independent sources of uncertainty in an agent's actions to an outsider: high-level planning, such as left or right turns at an intersection, and low-level control decisions conditioned on these plans, for example the decision to proceed or yield at any intersection before turning. Though we could model both sources of uncertainty, in this work we choose to direct our attention to the latter (actions conditioned on goals). To do this, we include a distance future location of the vehicle as part an agents observation (details in Appendix). One highly important use case for such a model is log replay for validation of autonomous vehicles, during which other vehicles have recorded goal states but must be simulated with alternative responses to tested planner.

Observation space We render agent states and road semantics into 6-channel images similar that in [6] (details in Appendix). This rendering preserves key geometric information in the state space and also ensures invariance to agent indexing.

Dynamics model A kinematic bicycle model converts steering and acceleration inputs, β, a, from the model into updates to the vehicle state: $[\dot{x}, \dot{y}, \dot{\psi}, \dot{v}] = [v\cos(\psi + \beta), v\sin(\psi + \beta), \frac{v}{l/2}\sin(\beta), a]$

Model architecture We extract visual features from the 6-channel images using a ResNet34 network [9, 16]. Our inference network uses the final hidden states of a BiRNN [22] over these visual features. Our policy combines the visual features and latent variable, z^i, and calculates the actions using a fully connected network.

Method	$\min_k \Delta$		$\min_k \chi^2(\pi_\theta, \pi_E) \times 10^{-4}$				
	avg.	final	*v*	IAD	DHW	TTC	BFS
resnet	4.08	12.52	25	51	73	3.1	1400
resnet-rnn	4.25	13.1	31	53	82	3.6	1900
resnet-gmm	3.98	12.30	27	47	63	2.8	1200
deep-latent-discrete	4.03	12.81	24	45	56	2.4	1100
deep-latent-continuous	3.87	10.34	24	45	55	2.3	1100
DIMMI-sa	3.76	**10.29**	23	32	50	1.7	450
DIMMI-ma	**3.75**	10.31	**23**	**31**	**48**	**1.5**	**420**

Table 1: We compare binned χ^2 distance between rollouts and demonstrations with respect to the velocity (*v*), *inter-agent distance* (IAD), *distance headway* (DHW), *time-to-collision* (TTC) and *bad final state* (BFS). For all metrics, smaller is better.

5.2 Evaluation metrics and baselines

Baseline models We baseline our approach against two unimodal models without regularization (resnet, resnet-rnn), three multimodal models without regularization (resnet-gmm, deep-latent-discrete, deep-latent-continuous), and compare them with two versions of our proposed method (DIMMI-*). Unimodal models exhibit only a single predominant action in response to a given observation, while multimodal models capture multiple kinds of responses.

We carefully chose these models to parallel the design of state-of-the-art multi-agent driver models and adapt them to our slightly modified setting. The "resnet" model is simply a resnet parameterizing a isotropic gaussian, trained using MLE. The "resnet-rnn" is the same as the above model with the addition of an autoregressive RNN, evoking the architecture of the ESP model from [19]. The "resnet-gmm" model is a resnet parametrizing a GMM, evoking [6]. "deep-latent-discrete" is a discrete latent variable model using the EM-type algorithm of [26] for learning. "deep-latent-continuous" is our proposed latent variable model without disagreement regularization. "DIMMI-sa" is a variant of DIMMI in which other agents are simply rolled out from logged data and thus are unresponsive to the model, and "DIMMI-ma" denotes the fully multi-agent variant of the algorithm. We do not compare with MA-GAIL [24] and MA-AIRL [28] because they require the number of agents to be fixed. Details of all baseline models are provided in the appendix.

Displacement statistics Displacement statistics simply measure the average or final distance (l_2) between simulated trajectories and ground truth given a shared starting point. In the case of generative models, ground truth might correspond to one of many possible modes of the distribution, and thus displacement is measured over k draws and the min value is considered, denoted $\min_k \Delta$. In multi-agent settings, this shortcoming becomes more pronounced, as individual modes are replaced with multi-agent modes. For this reason, we also examine state occupancy statistics.

State occupancy statistics We take a few metrics about agent states to be reflective of the emergent features. In particular, we examine *speed of vehicle*: (*v*), *inter-agent distance* (IAD), *distance headway* (DHW), *time-to-collision* (TTC), and *bad final state* (BFS). Detailed explanations of these metrics are given in the Appendix. In order to gauge the similarity between emergent features in expert and rolled out trajectories, we calculate the χ^2 value (calculation details in Appendix) between the histogram of values in the expert trajectories, $\tau \sim \pi_E$, and the histogram of values in rolled out trajectories, $\tau \sim \pi_\theta$. We use χ^2 values here because they are a natural way to measure discrepancy due to chance. As with displacement error calculations, we also calculate the values $\min_k$ to account for variability in the ground truth distribution. This allows us to compare the statistics of the trajectories more holistically than displacement statistics as the behavior of entire distributions is concerned.

5.3 Results

Quantitative evaluations Table 1 shows results for displacement and state occupancy metrics. We see that accounting for multi-modality (as in resnet-gmm, deep-latent-discrete, deep-latent-continuous) greatly reduces both displacement error and the chi-squared values. Notably, DIMMI also slightly reduces displacement error and significantly improves chi-squared values. By inspecting trajectories we can see that this is accomplished primarily by reducing the rate of collisions (affecting

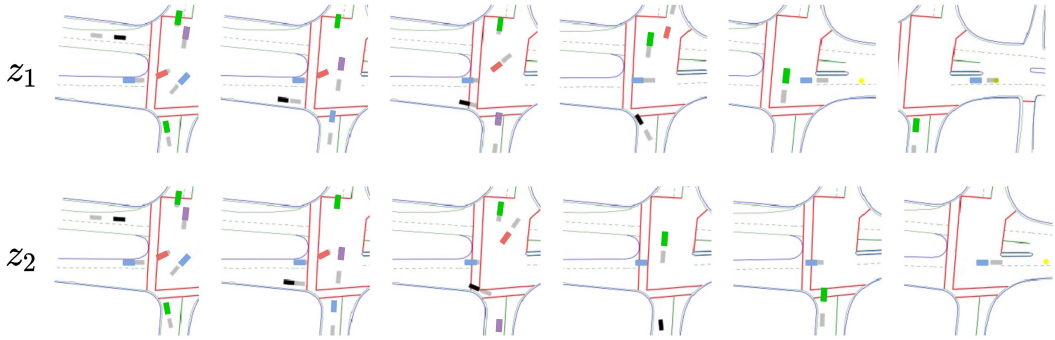


Figure 1: An example of the diverse behavior learned by the model. Grey blocks represent future state of vehicle given current velocity. Top: blue car at center proceeds through intersection. Bottom: blue car at center yields to green car in moving top to bottom through the intersection.

Figure 2: A spectrum of observations ranked by associated C_U. Grey blocks represent future state of vehicle given current velocity. The ego car is at the center of each frame. At the left, there are typical observations, and at the right there are state far from the data distribution, such as driving off-road.

IAD, DHW, and TTC) and off-road driving. Because DIMMI-ma can better capture how cars respond to each other when rolled out, it is able to more accurately model the distances between vehicles and avoid bad final states than DIMMI-sa.

Qualitative evaluations Figure 1 shows two example trajectories rolled out from a DIMMI model with latent variables z_1 and $z_2 = -z_1$. Not only are the two trajectories different conditioned on the same goal and start locations of all vehicles, but also the difference is intuitive. In the first example, the blue car drives through the intersection without yielding to an oncoming car (top green), and in the second example it acts more cautiously. This result is very impressive, as it shows the model is capable of not just obeying the rules of the road but also capturing styles of driving behavior. Equivalent diversity and accuracy cannot easily be accomplished in other current IL models.

Figure 2 shows example observations ranked by their associated cost C_U under a learned DIMMI ensemble. We see that the C_U estimates can successfully differentiate between banal states, like a full stop behind another car, and unusual states, like driving off-road or on a collision course with other cars.

Example videos To provide more examples from a wider variety of scenarios, we provide videos at the following url: `http://anonjohn256.github.io`.

6 Conclusion

We investigated the use of latent variable models and disagreement-regularization for imitating multi-agent interactions. Drawing on previous work, we proposed a novel algorithm for learning models in the regime of a large varying number of agents. To demonstrate the effectiveness of the algorithm, we aggregated a diverse collection of driving data (that contains around 160 hours of driving data) and trained different types of models (unimodal, multimodal, model with latent variables) at scale. Evaluated on this dataset, our approach outperformed baselines significantly in terms of displacement and state occupancy statistics, and we show that the latent variables learned by our model is able to control qualities like driver aggressiveness. In future work, we are interested in considering DIMMI in other domains such as air traffic modeling as well as exploring new approaches to interpret and explain the demonstrated and learned behavior via unsupervised learning.

References

[1] R. P. Bhattacharyya, D. J. Phillips, C. Liu, J. K. Gupta, K. R. Driggs-Campbell, and M. J. Kochenderfer. Simulating emergent properties of human driving behavior using multi-agent reward augmented imitation learning. pages 789–795, 2019.

[2] R. P. Bhattacharyya, D. J. Phillips, B. Wulfe, J. Morton, A. Kuefler, and M. J. Kochenderfer. Multi-agent imitation learning for driving simulation. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1534–1539. IEEE, 2018.

[3] J. Bock, R. Krajewski, T. Moers, S. Runde, L. Vater, and L. Eckstein. The ind dataset: A drone dataset of naturalistic road user trajectories at german intersections. 2019.

[4] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146, 2017.

[5] K. Brantley, W. Sun, and M. Henaff. Disagreement-regularized imitation learning. In *International Conference on Learning Representations*, 2020.

[6] Y. Chai, B. Sapp, M. Bansal, and D. Anguelov. Multipath: Multiple probabilistic anchor trajectory hypotheses for behavior prediction. *arXiv preprint arXiv:1910.05449*, 2019.

[7] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[8] B. Efron. Bootstrap methods: another look at the jackknife. In *Breakthroughs in statistics*, pages 569–593. Springer, 1992.

[9] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[10] M. Henaff, A. Canziani, and Y. LeCun. Model-predictive policy learning with uncertainty regularization for driving in dense traffic. *arXiv preprint arXiv:1901.02705*, 2019.

[11] J. Ho and S. Ermon. Generative adversarial imitation learning. In *Advances in Neural Information Processing Systems*, pages 4565–4573, 2016.

[12] D. P. Kingma and M. Welling. Auto-Encoding variational bayes. *arXiv preprint arXiv:1312.6114v10*, December 2013.

[13] R. Krajewski, T. Moers, J. Bock, L. Vater, and L. Eckstein. The round dataset: A drone dataset of road user trajectories at roundabouts in germany. submitted.

[14] Y. Li, J. Song, and S. Ermon. Infogail: Interpretable imitation learning from visual demonstrations. In *Advances in Neural Information Processing Systems*, pages 3812–3822, 2017.

[15] R. Lowe, Y. Wu, A. Tamar, J. Harb, P. Abbeel, and I. Mordatch. Multi-Agent Actor-Critic for mixed Cooperative-Competitive environments. *arXiv preprint arXiv:1706.02275*, June 2017.

[16] S. Marcel and Y. Rodriguez. Torchvision the machine-vision package of torch. In *Proceedings of the 18th ACM international conference on Multimedia*, pages 1485–1488, 2010.

[17] M. Montanino and V. Punzo. Making ngsim data usable for studies on traffic flow theory: Multistep method for vehicle trajectory reconstruction. *Transportation Research Record*, 2390(1):99–111, 2013.

[18] V. Punzo, M. T. Borzacchiello, and B. Ciuffo. On the assessment of vehicle trajectory data accuracy and application to the next generation simulation (ngsim) program data. *Transportation Research Part C: Emerging Technologies*, 19(6):1243–1262, 2011.

[19] N. Rhinehart, R. McAllister, K. Kitani, and S. Levine. Precog: Prediction conditioned on goals in visual multi-agent settings. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2821–2830, 2019.

[20] S. Ross and D. Bagnell. Efficient reductions for imitation learning. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 661–668, 2010.

[21] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, July 2017.

[22] M. Schuster and K. K. Paliwal. Bidirectional recurrent neural networks. *IEEE transactions on Signal Processing*, 45(11):2673–2681, 1997.

[23] W. Schwarting, J. Alonso-Mora, and D. Rus. Planning and decision-making for autonomous vehicles. *Annual Review of Control, Robotics, and Autonomous Systems*, 2018.

[24] J. Song, H. Ren, D. Sadigh, and S. Ermon. Multi-agent generative adversarial imitation learning. In *Advances in Neural Information Processing Systems*, pages 7461–7472, 2018.

[25] R. S. Sutton, D. A. McAllester, S. P. Singh, and Y. Mansour. Policy gradient methods for reinforcement learning with function approximation. In *Advances in neural information processing systems*, pages 1057–1063, 2000.

[26] C. Tang and R. R. Salakhutdinov. Multiple futures prediction. In *Advances in Neural Information Processing Systems*, pages 15398–15408, 2019.

[27] Z. Wang, J. S. Merel, S. E. Reed, N. de Freitas, G. Wayne, and N. Heess. Robust imitation of diverse behaviors. In *Advances in Neural Information Processing Systems*, pages 5320–5329, 2017.

[28] L. Yu, J. Song, and S. Ermon. Multi-agent adversarial inverse reinforcement learning. *arXiv preprint arXiv:1907.13220*, 2019.

[29] L. Yu, T. Yu, C. Finn, and S. Ermon. Meta-inverse reinforcement learning with probabilistic context variables. *NeurIPS*, 2019.

[30] W. Zhan, L. Sun, D. Wang, H. Shi, A. Clausse, M. Naumann, J. Kummerle, H. Konigshof, C. Stiller, A. de La Fortelle, et al. Interaction dataset: An international, adversarial and cooperative motion dataset in interactive driving scenarios with semantic maps. *arXiv preprint arXiv:1910.03088*, 2019.

Appendix

Problem Definition

Birth-death formalism We introduce a formalism that is useful for modeling "birth-death" processes with a *varying number of agents*. We concern ourselves with finite intervals of the birth-death process, which contain a bounded number of unique agents.

In this setting, we assume there are N agents that are part of the process and that the state space $\mathcal{S}$ is $\mathcal{S}^1 \times \cdots \times \mathcal{S}^N$, where each $\mathcal{S}^i$ is an agent-specific state space. We augment the state space by adding a discrete ontological state per agent $\varnothing^i$ with $\varnothing^i \in \{\varnothing_P, \varnothing_A, \varnothing_D\}$, and a null state, $s_\varnothing$, to each $\mathcal{S}^i$. Here $\varnothing_P$ is a prenatal state, $\varnothing_A$ an alive state and $\varnothing_D$ a death state. This yields a new state space $\mathcal{S}^\pm = (\mathcal{S}^1 \cup \{s_\varnothing\}) \times \cdots \times (\mathcal{S}^N \cup \{s_\varnothing\}) \times \varnothing^1 \times \cdots \times \varnothing^N$.

We also add a null observation to each observation set $\mathcal{O}^i = \mathcal{O}^i \cup \{o_\varnothing\}$ and a null action to each action set $\mathcal{A}^i = \mathcal{A}^i \cup \{a_\varnothing\}$. In this setting the state transition dynamics $T : \mathcal{S}^\pm \times \mathcal{A}^1 \times \cdots \times \mathcal{A}^N \mapsto \mathcal{P}(\mathcal{S}^\pm)$ is the product of $(T^+, T^\rightarrow, T^-)$, where T^+ is "birth" process, T^- is a "death" process, and $T^\rightarrow$ is the dynamics model over states of alive agents. T^+ and T^- govern the dynamics of $\{\varnothing^i\}$, while $T^\rightarrow$ and T^+ govern the dynamics of $\{\mathcal{S}^i \cup \{s_\varnothing\}\}$. The data generating process can thus be described by the following phases:

1. **Initialization** Sample $s_1 \in \mathcal{S}^\pm \sim P(\boldsymbol{s}_1)$. The only constraint on $P(\boldsymbol{s}_1)$ is
$$\exists i\,,\, \varnothing_1^i = \varnothing_D \implies P(\boldsymbol{s}_1) = 0$$
where $\varnothing_t^i$ is taken to be the ontological state of the i-th agent at time t. All agents in state s_1 are thus either prenatal or alive, and those that are prenatal have $s_1^i = s_\varnothing$.
2. **Agents receive observations** Observations, $\boldsymbol{o}_t \sim P(\boldsymbol{o}_t|\boldsymbol{s}_t)$. One constraint operates on $P(\boldsymbol{o}_t|\boldsymbol{s}_t)$,
$$\exists i, s_i^t = s_\varnothing, o_i^t \neq o_\varnothing \implies P(\boldsymbol{o}_t|\boldsymbol{s}_t) = 0$$
i.e. unborn or dead agents receive null observations.
3. **Agents take actions** The set of actions is sampled from $\boldsymbol{\pi}(\boldsymbol{a}_t|\boldsymbol{o}_t)$. We take as part of the setting that
$$\exists i\,,\, o_t^i = o_\varnothing, a_t^i \neq a_\varnothing \implies P(\boldsymbol{a}_t|\boldsymbol{o}_t) = 0$$
and thus any unborn or dead agents take null actions.
4. **Agents are born** T^+ is a joint transition distribution over all agents describing transitions to and from $\varnothing_P$. These transitions follow a few intuitive constraints by default:
$$\begin{aligned}
\exists i\,,\, \varnothing_t^i \neq \varnothing_P, \varnothing_{t+1}^i = \varnothing_P &\implies P(\boldsymbol{s}_{t+1}|\boldsymbol{s}_t, \boldsymbol{a}_t) = 0 \\
\exists i\,,\, \varnothing_t^i = \varnothing_P, \varnothing_{t+1}^i = \varnothing_D &\implies P(\boldsymbol{s}_{t+1}|\boldsymbol{s}_t, \boldsymbol{a}_t) = 0 \\
\forall i\,,\, \varnothing_t^i \neq \varnothing_P, \varnothing_{t+1}^i \neq \varnothing_P &\implies P(\boldsymbol{s}_{t+1}|\boldsymbol{s}_t, \boldsymbol{a}_t) = 1
\end{aligned}$$
In other words, T^+ only allows transitions from prenatal to alive states and only affects transitions to or from prenatal states.
5. **Alive agents transition** For s_t^i, a_t^i with $\varnothing_t^i = \varnothing_A$, transition to the next state $\boldsymbol{s}_{t+1} \sim T^\rightarrow(\boldsymbol{s}_{t+1}|\boldsymbol{s}_t, \boldsymbol{a}_t)$. The only constraint on $T^\rightarrow$ is
$$\forall i\,,\, \varnothing_t^i \neq \varnothing_A, \varnothing_{t+1}^i \neq \varnothing_A \implies P(\boldsymbol{s}_{t+1}|\boldsymbol{s}_t, \boldsymbol{a}_t) = 1$$
This is the transition model of a standard POMDP.
6. **Agents die** T^- describes transitions to and from $\varnothing_D$ and encodes the following constraints by default:
$$\begin{aligned}
\exists i\,,\, \varnothing_t^i = \varnothing_D, \varnothing_{t+1}^i = \varnothing_A &\implies P(\boldsymbol{s}_{t+1}|\boldsymbol{s}_t, \boldsymbol{a}_t) = 0 \\
\forall i\,,\, \varnothing_t^i \neq \varnothing_D, \varnothing_{t+1}^i \neq \varnothing_D &\implies P(\boldsymbol{s}_{t+1}|\boldsymbol{s}_t, \boldsymbol{a}_t) = 1 \\
\exists i\,,\, s_{t+1}^i \neq s_\varnothing &\implies P(\boldsymbol{s}_{t+1}|\boldsymbol{s}_t, \boldsymbol{a}_t) = 0
\end{aligned}$$
Non-default parameters of T^- define the death conditions for an agent.

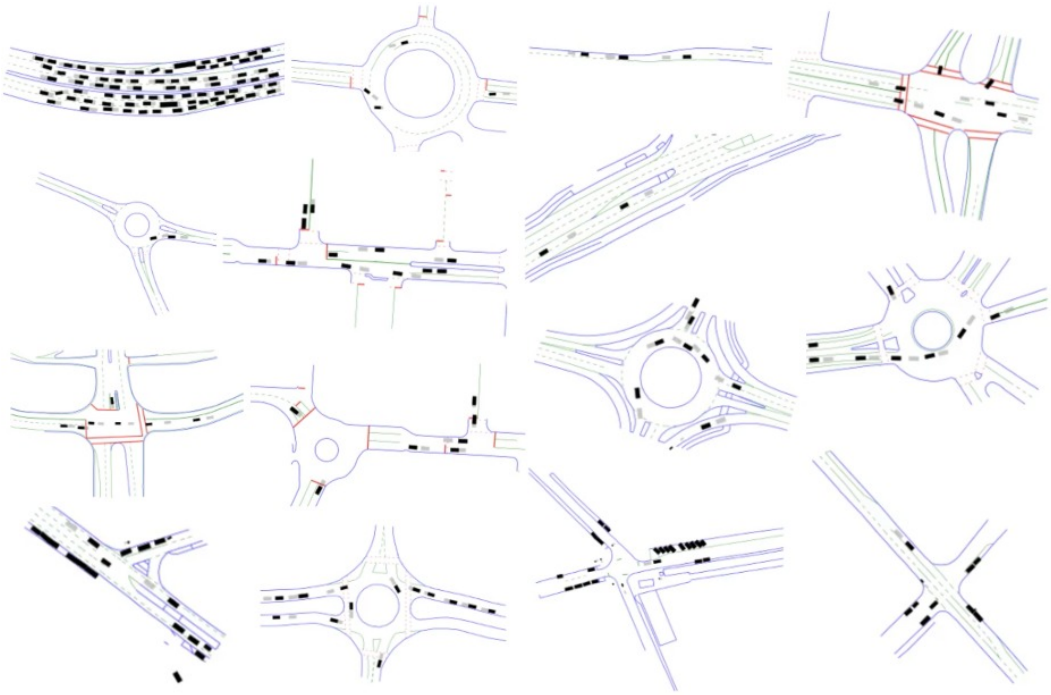


Figure 3: Renderings for all of the scenarios in the training data, using track files and open street map files from [30], [3], and [13].

Experiments – Inputs

Training scenarios Figure 3 shows a visualization of all the scenarios used in the training data. The impressive diversity of this aggregated dataset aids the generalization capabilities of the model and leads to latent variables with high-level structure, like aggressive vs. passive preferences.

All the scenarios are sampled at 10 Hz or 8 Hz, with approximately 6 million unique timestamps (160 hrs) collectively, counting each agents individually.

Agent types There are 4 classes of agents (divided by size) considered in the dataset, with the name of each sub-type as provided in the original dataset:

1. Pedestrians: “person”, “pedestrian”
2. Motorcycles (FHWA class 1) and bicycles: “motorcycle”, “bicycle”
3. Passenger cars (FHWA class 2): “car”
4. Large vehicles (FHWA class 3-13): “bus”, “trailer”, “truck”, “truck-bus”, “van”

Policies for class 3 and 4 agents are learned by our model, while agents for class 1 and 2 are rolled out from logs.

Goal states Goal locations are calculated as the closest position of the vehicle that is more than 45 meters away. We chose this distance to be at the edge of the vehicle’s field-of-view. The field-of-view was chosen to capture relevant road features, and thus the goal naturally captures intermediate steps like right-hand turns while leaving interactions with other agents up to the policy.

Observation representation Our chosen observation representation is key to the success of our model. We take inspiration from [6] and create a 6-channel image comprising road semantics, agent states, and an goal location, projected to inertial reference frame of the modeled vehicle:

1. Stop and yield lines
2. Lane and traffic flow markings
3. Road edges and curbs
4. Current bounding boxes of vehicles (as closed shapes)
5. Projected bounding boxes of vehicles given current speed 1 sec in future (as closed shapes)
6. Future goal location rendered as small circle

Road markings are obtained from OpenStreetMap (OSM) format maps available with the datasets, along with vehicle types and sizes. Bounding boxes are created from the center location, length, and width of agents from class 3 and 4. Pedestrians are taken to have length and width of 0.75 meters, while class 2 agents have length 1.75 meters and width 0.75 meters.

Because speeds are encoded in our representation, the observation can, in practice, be used without stacking multiple frames.

Experiments – Models

Assumed distributions We assumed knowledge of $P(\boldsymbol{o}_t|\boldsymbol{s}_t)$, $P(\boldsymbol{s}_{t+1}|\boldsymbol{s}_t, \boldsymbol{a}_t)$, and $P(\boldsymbol{s}_1)$. In our experimental setting $P(\boldsymbol{o}_t|\boldsymbol{s}_t)$ and $P(\boldsymbol{s}_{t+1}|\boldsymbol{s}_t, \boldsymbol{a}_t)$ are deterministic and given by the observation model above and the kinematic bicycle model introduced in Section 5. $P(\boldsymbol{s}_1)$ is taken to be the empirical distribution in the training dataset. In rollouts, agent start states are thus taken to be the start states of held-out trajectories.

Baseline models We create baselines inspired by PRECOG [19], Multiple Futures Prediction (MFP) [26], and MultiPath [6], adapting them to our setting where there is a pixel-space input, a goal location and no conditioning on a fixed number of past states in the trajectory (as is common in single-agent trajectory prediction tasks).

All of these models use neural network models to parameterize a gaussian distribution/s, $\mathcal{N}(\mu_\theta, \Sigma_\theta)$, over the next state of the vehicle. The algorithms differ largely in input representation and modeling of uncertainty.

1. **PRECOG** PRECOG-ESP uses an RNN to generate $(\mu_\theta, \Sigma_\theta)$ for each time step and agent autoregressively. Uncertainty is captured by the density of the unimodal normal distribution independently at each timestep. As a comparable baseline we take an RNN encoding $(\mu_\theta, \Sigma_\theta)$ that takes outputs of our feature-extracting resnet as input.
2. **MFP** MFP maps a sequences of past states X to a distribution over targets Y, $P_\theta(Y|X)$, using discrete latent variable $Z \in \{1, ..., k\}$, with loss function:

   $$\mathcal{L}(\theta, \mathcal{D}) = \log P_\theta(Y|X) = \log \left(\sum_Z P_\theta(Z|Y, X) \log \frac{P_\theta(Y, Z|X)}{P_\theta(Z|Y, X)} \right)$$

   This loss function is maximized using an EM-type procedure, updating θ_t on loss

   $$\sum_Z P_{\theta_{t-1}}(Z|Y, X) \log P_{\theta_t}(Y|Z, X) + D_{\mathrm{KL}}(P_{\theta_{t-1}}(Z|Y, X)||P_{\theta_t}(Z|X))$$

   where marginalization is performed exactly over the discrete Z.

   In our setting, we take the stepwise loss to be

   $$\sum_{z^i} P_{\theta_{t-1}}(z^i|o_t^i, s_{1:T}^i) \log P_{\theta_t}(s_{t+1:T}^i|z^i, o_t^i, s_{1:t}^i) + D_{\mathrm{KL}}(P_{\theta_{t-1}}(z^i|o_t^i, s_{1:T}^i)||P_{\theta_t}(z^i|o_t^i, s_{1:t}^i))$$

   Sequences of states are encoded using an RNN and combined with observation encodings from our resnet.
3. **MultiPath** MultiPath factorizes trajectory likelihood as

   $$P(s_{1:T}^i|o_1^i) = \sum_{k=1}^K P_\theta(a^k|o_1^i) \prod_{t=1}^T P(s_t^i|a^k, o_1^i)$$

where $\{a^k\}$ are a set of prelearned "anchor" trajectories, $a^k = \{s_{1:T}\}$, that are representative of high-level agent intentions. With these prelearned anchors, a Gaussian mixture model (GMM) can be learned using a hard-assignment learning algorithm in which each trajectory is assigned to its nearest anchor in distance, and class probabilities $P_\theta(a^k|o_1^i)$ and offsets from the anchor $(\mu_\theta, \Sigma_\theta)$ are learned through MLE. The parameters of the probabilistic model are output by a resnet.

Inspired by MultiPath's approach, we learn a GMM from prefit anchor means with parameters output by our same resnet model.

DIMMI details We provide further details to facilitate reproducing our experimental results. These detail are broken down into sections for clarity,

1. **Pretraining** In order to learn useful visual features for use in many models, we train a resnet34 on the entire training dataset with mean squared error from ground truth actions.

BC hyperparameters

parameter	value
lr	1e-3
lr schedule	cosine
batch size	64

2. **Latent variable models** In the latent variable models, the convolutional layers of the resnet are preserved, and a new fully connected layer is learned output an observation "encoding". In the inference network, this encoding is mapped to a distribution over the latent variable, and in the action model, it is mapped to a distribution over the actions. For training the RNN inference model, we downsampling encodings in time to decrease the difficulty of backpropagation through time.

VAE hyperparameters

parameter	value
encoding dim	128
action hidden dim	512
latent dim	8
rnn hidden dim	128
downsample rate	4
grad clip	10
lr	1e-3
lr scheduler	cosine
batch size	16

3. **Regularization** To regularize the policy, we must train an ensemble and optimize the policy against this ensemble with RL. For our experiments, we used vanilla policy gradient optimization with a baseline. Exploration can be achieve through maximizing entropy of the output model when action are non-deterministic given the latent variable and observation. When they are deterministic, we use parameter noise to ensure exploration.

Ensemble hyperparameters:

parameter	value
k	5
q	0.95

RL hyperparameters

parameter	value
lr	5e-4
batch size	256

Experiments – Evaluation

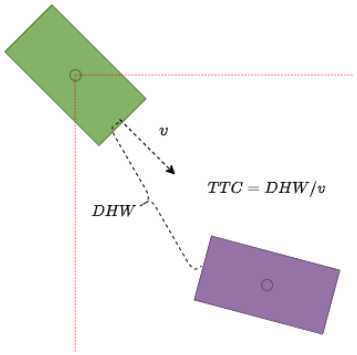
Evaluation dataset Because of limited time, we performed quantitative evaluations using held-out trajectories from on a subset of the training scenarios, namely 2 roundabout and 2 intersection

scenarios with dense traffic. These scenarios were chosen because they were challenging but not the most computationally intensive–so multiple iterations of each experiment could easily be run.

Evaluation metrics For our state-occupancy metrics we calculated: speed (v), *inter-agent distance* (IAD), *distance headway* (DHW), *time-to-collision* (TTC), and *bad final state* (BFS). Speed was calculated as expected. IAD at time t was taken to be

$$\sum_{i=1}^{N}\sum_{j=1}^{N} d(i,j)\ \mathbb{1}[d(i,j) < D]$$

$$d(i,j,t) = \sqrt{(x^i - x^j)^2 + (y^i - y^j)^2}$$



The DHW and TTC calculations are shown in the figure to the right. DHW is taken to be

$$\sum_{i=1}^{N}\sum_{j=1}^{N} d^*(i,j)\ \mathbb{1}\left[d(i,j) < D, |\omega_{ij}| < \frac{\pi}{4}\right]$$

$$\omega_{ij} = \arctan\left(\frac{y^i - y^j}{x^i - x^j}\right) - \phi^j$$

$$d^*(i,j) = \sqrt{(f(x^i,\phi^i) - f(x^j,\phi^j))^2 + (f(y^i,\phi^i) - y^j)^2}$$

$$f(x^i,\phi^i) = x^i + \frac{l^i}{2}\cos(\phi^i)$$

$$f(y^i,\phi^i) = y^i + \frac{l^i}{2}\sin(\phi^i)$$

and TTC is

$$\sum_{i=1}^{N}\sum_{j=1}^{N} \frac{d^*(i,j)}{v}\ \mathbb{1}\left[d(i,j) < D, |\omega_{ij}| < \frac{\pi}{4}\right]$$

Lastly, BFS is given by

$$\sum_{i=1}^{N} \text{crash}(i)$$

$$\text{crash}(i) = \begin{cases} 1 & \text{bbox}_i \text{ intersects bbox}_j,\ i \neq j \\ 1 & \text{centerline}_i \text{ intersects road border} \\ 0 & \text{else} \end{cases}$$