Multi-agent Adversarial Inverse Reinforcement Learning with Latent Variables

Paper #1356

ABSTRACT

We introduce an algorithm for inferring reward functions from expert human trajectories in multiagent environments. Current techniques exhibit poor sample-efficiency, lack stability in training, or scale poorly to large numbers of agents. We focus on settings with a large, variable number of agents and attempt to resolve these settings by exploiting similarities between agent behaviors. In particular, we learn a shared reward function using a variant of adversarial inverse reinforcement learning. This reward function is able to fit a broad array of behavior by means of a latent variable learned using a variational autoencoder. The efficacy of our algorithm is demonstrated on two large real-world datasets and instances of traffic congestion, including cars on highways and aircraft in terminal airspace. To combat the complexity associated with a large, varying number of agents, we describe neural architectures including attentional and recurrent modules. Our model demonstrates that latent variable discrimination is a compelling approach to learning reward functions in crowded environments at scale.

KEYWORDS

Adversarial inverse reinforcement learning; Human trajectory modeling; Latent variable model

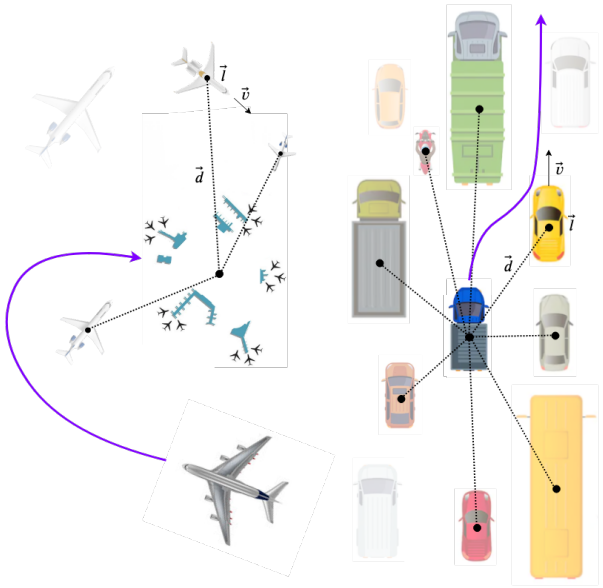

Figure 1: Navigation through highway traffic and terminal air traffic. Our model uses location-based attention to encode the multiagent information within a region of interest.

1 INTRODUCTION

Accurate models of human behavior are increasingly important to safe and effective deployment of autonomous systems. Despite this need, behavior modeling remains difficult for various common problem settings. Urban environments, for example, still pose significant challenges for autonomous planning because of the uncertainty resulting from a high density of people [37]. To describe these scenarios robustly, a model must capture multi-modality in agent motivations and complex interactions that often scale superlinearly with the number of agents.

Two common approaches to modeling human behavior are imitation learning and inverse reinforcement learning (IRL). Imitation learning aims to produce trajectories that match a given expert distribution and can be attempted with techniques as simple as supervised learning – a.k.a. behavior cloning [31]. Inverse reinforcement learning, on the other hand, seeks to learn a reward function that can rationalize expert demonstrations [32]. The latter task is often much more challenging, but offers a concise description of the data generating process when accomplished. If the features used for IRL are sufficiently abstract, learned reward functions can also be transferred across problems more easily than learned mappings from states to actions [46].

Recent research in imitation and reward learning for multi-agent trajectories is largely divided into two mostly distinct fields of research. The first of these is the large body of work on trajectory modeling from the computer vision community, with application to both pedestrians [1] [14] [41] [36] and vehicles [7] [8] [26] [34] [48]. These approaches make few assumptions about the agents' environment and attempt to reduce trajectory modeling to general analysis and forecasting of time series (of states and actions). The focus is on developing neural network architectures that have inductive biases matching the unique challenges of multi-agent modeling. In many cases, this generality and embrace of state-of-the-art deep learning have lead to impressive performance on real-world data; yet these methods require more samples and are less interpretable, making it difficult to incorporate prior knowledge about agents' behavior into the model design.

A second group of researchers from the RL and robotics communities have focused more on decision-making frameworks that account for agent rationality [38] [45] [23]. These approaches can leverage ideas from game theory by explicitly modeling the data generating process as Markov Decision Process (MDP) [17] or Markov Game [30]. This added structure, for instance, allows for explicit modeling of competitive/cooperative behavior that might otherwise be difficult to infer from scratch. For the most part, these

Proc. of the 19th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2020), B. An, N. Yorke-Smith, A. El Fallah Seghrouchni, G. Sukthankar (eds.), May 2020, Auckland, New Zealand

https://doi.org/doi

works have focused on smaller problem settings and are less likely to analyze real-world datasets. This trend is due at least in part to the increased difficulty of scaling these techniques to large numbers of agents and less structured and/or lower-quality data [38]. Progress can also be held back by the poor training stability of model-free deep reinforcement learning [15], which is often a necessary piece of these frameworks.

One goal of this work is to offer a solution that can speak to both communities by offering some degree of scale and abstraction. We propose a framework, "birth-death" Markov games, for describing environments with a large, varying number of agents, and suggest a class of models that can be used effectively within this framework. We focus of on the task of multiagent trajectory modeling and demonstrate how highly scalable solutions from the deep learning community can be employed. We improve existing techniques by proposing a method for learning a shared multimodal reward function. This addition significantly improves interpretability by explicitly modeling agent rationality. The scope of these models is also widened, as learned reward functions can be used for a variety of downstream tasks beyond generating plausible trajectories. To test our models, we focus on multiagent interactions as they manifest in traffic congestion, both for cars on highways and aircraft in terminal airspace (Figure 1). These scenarios are characterized by a flow of agents that share goals but must negotiate their spatial relation to each other. Beyond the challenge of a large number of agents (10–100), traffic flows contain the additional complexity of agents entering and leaving the frame of observation. This relatively small difference makes many previous algorithms difficult to apply, but is easily handled within our framework.

In what follows we propose our solutions to these challenges, first within the theoretical framework of adversarial inverse reinforcement learning (AIRL) and then from a practical engineering standpoint, detailing the structure of our models and method of optimization. We then examine a few of our learned policies and reward functions, which we show to be highly plausible.

2 BACKGROUND

2.1 Markov Games

A N-player Markov game [30] is a generalization to Markov decision processes (MDPs, where $N = 1$). A Markov game $\Gamma = (\mathcal{S}, \mathcal{A}, P, \eta, \boldsymbol{r}, K)$ is defined via state sets $\mathcal{S}$, action sets $\mathcal{A}_{i=1}^{N}$ for each agent, transition process between states $T : \mathcal{S} \times \mathcal{A}^1 \times \cdots \times \mathcal{A}^N \to \mathcal{P}(\mathcal{S})$ [1], the initial state distribution $\eta \in \mathcal{P}(\mathcal{S})$, a set of reward functions $r^i : \mathcal{S} \times \mathcal{A}^1 \times \cdots \times \mathcal{A}^N \to \mathbb{R}$ for each agent, and the horizon K. Let $\boldsymbol{\pi} = [\pi_1, \ldots, \pi_N]$ denote the joint policy of all agents and $\boldsymbol{a} = [a_1, \ldots, a_N]$ denote the joint actions; we assume that the agents are stationary, i.e. $\boldsymbol{\pi}$ is time independent.

2.2 Adversarial Imitation Learning and Inverse Reinforcement Learning

In a single agent imitation learning setup ($N = 1$), we do not assume access to the ground truth reward r, but have access to a dataset of demonstrations $\mathcal{D}$ provided by an expert policy denoted as π_E. Specifically, $\mathcal{D}$ is a set of M trajectories $\{\tau_j\}_{j=1}^{M}$, where τ_j is an expert trajectory sampled from $\boldsymbol{s}_0 \sim \eta(\boldsymbol{s}), \boldsymbol{a}_t \sim \pi_E(\boldsymbol{a}_t|\boldsymbol{s}_t), \boldsymbol{s}_{t+1} \sim T(\boldsymbol{s}_{t+1}|\boldsymbol{s}_t, \boldsymbol{a}_t)$. Imitation learning (IL) aims to directly learn policies that have similar behaviors to the demonstrations, whereas inverse reinforcement learning (IRL [32, 35] seeks to identify the reward function under which the expert policies are "optimal".

The maximum causal entropy RL framework (MaxEnt RL [49]) considers an additional (causal) entropy regularization to the RL objective:

$$\mathrm{RL}(r) = \arg\max_{\pi} \mathcal{H}(\pi) + \mathbb{E}_{\pi}\left[\sum_{t=0}^{K} r(s_t, a_t)\right] \tag{1}$$

where $\mathcal{H}(\pi) = \mathbb{E}_{\pi}[-\log \pi(a|s)]$ is the policy entropy. The corresponding inverse reinforcement learning problem (MaxEnt IRL) is then defined as:

$$\mathrm{IRL}(\pi_E) = \arg\max_{\pi} \mathbb{E}_{\pi_E}\left[\sum_{t=0}^{K} r(s_t, a_t)\right] - \mathrm{RL}(r) \tag{2}$$

which assumes that π_E is the optimal (MaxEnt) policy under some reward r. In the context of imitation learning, prior work have considered utilizing generative adversarial training [13] to perform optimization over π and r in an iterative fashion [16, 28, 38].

The MaxEnt RL framework is interesting because it relates the probability of sampling a trajectory by the optimal policy to the reward; when the dynamics T are deterministic, the probability of sampling a trajectory $\tau = \{(s_t, a_t\}_{t=0}^{\infty}$ is determined by the following energy function:

$$p(\tau) = \frac{\exp(\sum_{t=0}^{K} r(s_t, a_t))}{Z(r)} \tag{3}$$

where $Z(r)$ is a normalization constant that sums over all valid trajectories, and γ is assumed to be 1. One could then perform maximum likelihood estimation (MLE) of r (or a suitable parameterization of r) over the demonstrations to recover the reward,

$$\max_{\theta} \mathbb{E}_{\pi_E}\left[\sum_{t=0}^{K} \log \pi_\theta(a_t|s_t)\right] = \max_{r} \mathbb{E}_{\pi_E}\left[\sum_{t=0}^{K} \log r(s_t, a_t) - \log Z(r)\right] \tag{4}$$

assuming that the transition dynamics is deterministic. Unfortunately, it is often difficult to estimate the normalization constant Z for large or continuous state spaces (which requires summing / integrating over all valid trajectories). The adversarial inverse reinforcement learning approach [11, 12] addresses this issue by training a generator / discriminator pair (G, D) where G is neural-network parametrized policy π_θ and $D_{\theta,\phi}$ is the following odds-ratio

$$D_{\theta,\phi}(s, a) = \frac{\exp(r_\phi(s, a))}{\exp(r_\phi(s, a)) + \pi_\theta(a|s)} \tag{5}$$

where we involve r_ϕ to parametrize the reward.

The policy is trained to maximize the expected log probability assigned by the discriminator:

$$\max_{\theta} \mathbb{E}_{a_t \sim \pi_\theta(a_t|s_t),\, s_{t+1} \sim T(s_{t+1}|s_t, a_t)}[\log D_{\theta,\phi}(s, a)]$$

and the discriminator minimizes its misclassification of trajectories from the expert policy π_E:

$$\max_{\phi} \mathbb{E}_{\tau \sim \pi_E}[\log D_{\theta,\phi}(s, a)] + \mathbb{E}_{\tau \sim \pi_\theta}[\log(1 - D_{\theta,\phi}(s, a))]$$

[1] $\mathcal{P}(\mathcal{S})$ denotes the set of probability distributions over $\mathcal{S}$.

At the equilibrium of the solution, it can be shown that $r_\phi(s,a)$ will be the advantage function of the policy π_E. Multi-agent analogues of adversarial imitation learning and inverse reinforcement learning have been proposed [38, 45].

3 PROBLEM SETUP

Birth-Death Markov Games

Motivated by transportation systems and massive multiplayer online games, we introduce a new setting 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 Markov game framework by adding a discrete ontological state per agent O^i with $O^i \in \{O[p], O[a], O[d]\} = O$, and a null state, s_O, to each $\mathcal{S}^i$. Here $O[p]$ is a prenatal state, $O[a]$ an alive state and $O[d]$ a death state. This yields a new state space $\mathcal{S}^\pm = (\mathcal{S}^1 \cup \{s_O\}) \times \cdots \times (\mathcal{S}^N \cup \{s_O\}) \times O^1 \times \cdots \times O^N$. We also add a null action to each action set $\mathcal{A}^i = \mathcal{A}^i \cup \{a_O\}$. 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 taken to be factorized as 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 $\{O^i\}$, while $T^\rightarrow$ and T^+ govern the dynamics of $\{\mathcal{S}^i \cup \{s_O\}\}$. As before, there is a set of reward functions over $\mathcal{S}^\pm \times \mathcal{A}^1 \times \cdots \times \mathcal{A}^N$ and an initial state distribution $\eta \in \mathcal{P}(\mathcal{S}^\pm)$. The data generating process can thus be described by the following phases:

Initialization. Sample $s_0 \in \mathcal{S}^\pm \sim \eta$. The only constraint on η is

$$\exists i\,,\ O_0^i = O[d] \implies P(s_0) = 0$$

where O_t^i is taken to be the ontological state of the i-th agent at time t. All agents in state s_0 are thus either prenatal or alive, and those that are prenatal have $s_0^i = s_O$, where s_t^i is the state of agent i at time t.

Agents take actions. The set of actions $\boldsymbol{a}_t = \{a_t^i\}$ is sampled jointly from $\boldsymbol{\pi}(a_t|s_t)$. We take as part of the setting that $\pi^i(a_t^i = a_O | s_t^i = s_O) = 1$, and thus any unborn or dead agents take null actions.

Agents are born. T^+ is a joint transition distribution over all agents describing transitions to and from $O[p]$. These transitions follow a few intuitive constraints by default:

$$\exists i\,,\ O_t^i \neq O[p], O_{t+1}^i = O[p] \implies P(s_{t+1}|s_t, a_t) = 0$$
$$\exists i\,,\ O_t^i = O[p], O_{t+1}^i = O[d] \implies P(s_{t+1}|s_t, a_t) = 0$$
$$\forall i\,,\ O_t^i \neq O[p], O_{t+1}^i \neq O[p] \implies P(s_{t+1}|s_t, a_t) = 1$$

In other words, T^+ only allows transitions from prenatal to alive states and only affects transitions to or from prenatal states.

Alive agents transition. For s_t^i, a_t^i with $O_t^i = O[a]$, transition to the next state $s_{t+1} \sim T^\rightarrow(s_{t+1}|s_t, \{a^i\})$. The only constraint on $T^\rightarrow$ is

$$\forall i\,,\ O_t^i \neq O[a], O_{t+1}^i \neq O[a] \implies P(s_{t+1}|s_t, a_t) = 1$$

This is the transition model of a standard Markov game.

Agents die. T^- describes transitions to and from $O[d]$ and encodes the following constraints by default:

$$\exists i\,,\ O_t^i = O[d], O_{t+1}^i = O[a] \implies P(s_{t+1}|s_t, a_t) = 0$$
$$\forall i\,,\ O_t^i \neq O[d], O_{t+1}^i \neq O[d] \implies P(s_{t+1}|s_t, a_t) = 1$$
$$\exists i\,,\ s_{t+1}^i \neq s_O \implies P(s_{t+1}|s_t, a_t) = 0$$

Non-default parameters of T^- define the death conditions for an agent.

Learning Problem

In the imitation learning / inverse reinforcement learning setup, we are given a set of trajectories $\tau = [(s_0, a_0), (s_1, a_1), \ldots]$ containing the state-action pairs that serve as demonstrations of the underlying behavior we wish to learn (imitation) or rationalize (inverse reinforcement learning). We assume perfect knowledge of $(\eta, T^+, T^\rightarrow, T^-)$ and that $T^\rightarrow$ and π are factorizable over individual agents:

$$T^\rightarrow(s_{t+1}|s_t, a_t) = \prod_i T'(s_{t+1}^i | s_t^i, a_t^i)$$
$$\pi(a_t|s_t) = \prod_i \pi^i(a_t^i|s_t)$$

We denote the distribution over possible π^i and their respective r^i as $P_\theta(\pi)$ and $P_\phi(r)$.

Our goal is to learn some parameterized policy distribution $P_\theta(\pi)$ and reward distribution $P_\phi(r)$ such that the data generated by a birth-death Markov game with $\pi^i \sim P_\theta(\pi)$ and $r^i \sim P_\phi(r)$ is indistinguishable from the set of expert trajectories. We further assume that the expert policies form a Logistic Stochastic Best Response Equilibrium (LSBRE, [45]) where each agent applies a stochastic (entropy-regularized) best response mechanism. In LSBRE, imitation learning can be casted as maximum likelihood estimation over the expert trajectories as follows:

$$\max_\theta \mathbb{E}_{\tau \sim \pi_E}\left[\sum_{i=1}^N \log\left(\sum_\pi P_\theta(\pi) \prod_t \pi^i(a_t^i|s_t)\right)\right] \tag{6}$$

which we can also rewrite in terms of r_ϕ via Equation (4). The only modification that must be made for the birth-death setting is that for all $\pi^i \sim P(\pi)$,

$$\exists i,\ ,\ a_t^i \neq a_O, s_t^i = s_O \implies \pi^i(a_t^i|s_t) = 0$$

We can accomplish this by taking $P(\pi)$ to be a distribution over the subset of π^i satisfying this constraint.

4 METHOD

4.1 Shared structure with latent variables

In the above problem, learning can be difficult as there are almost no structural assumptions among the agents, and thus the joint policy π can be arbitrarily complex. In particular, the objective in Equation 6 is difficult to optimize due to marginalization over the

sampled policy. To address the marginalization over $P_\theta(\pi)$ in Equation 6, we propose to model the policy and reward functions with latent variables. Specifically, the agent's policy and reward function are uniquely determined once a low dimensional latent variable z^i is given. Each agent has its own latent variable z_i unknown to other agents and the environment. For example, in a driving environment, all the cars have similar physical conditions but the driver could have different styles in driving (passive / aggressive) that determines the policy.

Therefore, we may assume the policy and reward function for agent i can be modeled using a latent variable z^i, and are defined as $\pi_\theta(a_t^i|s_t, z^i)$ and $r_\phi(a_t^i, s_t|z^i)$ with parameters θ and ϕ respectively. We further assume that the latent variable has the prior $p(z)$, which is a multivariate Gaussian distribution:

$$p(z) = \mathcal{N}(\mu_z, \sigma_z);$$

combined with the policy and reward function, this defines a process for sampling policies and reward functions for each agent from $P_\theta(\pi)$ and $P_\phi(r)$ by sampling $z^i \sim p(z)$ and using the policy and reward determined by z^i for each agent.

With the latent variables and the corresponding policy models, the objective in Equation 6 becomes:

$$\mathcal{L}(\theta) = \mathbb{E}_{\tau\sim\pi_E}\left[\sum_{i=1}^{N}\log\left(\int_{z^i} p(z^i)\prod_t \pi_\theta(a_t^i|s_t, z^i)\mathrm{d}z^i\right)\right] \quad (7)$$

To remove the summation over $p(z^i)$ in the log, we introduce a inference model $q_\omega(z^i|\tau^i)$, where $\tau^i = \{(s_t^i, a_t^i)\}$ is the trajectory for agent i. This leads to an evidence lower bound to $L(\theta)$ [21]:

$$\mathcal{L}(\theta) \geq \mathbb{E}_{\tau\sim\pi_E}\left[\sum_{i=1}^{N}\mathbb{E}_{z^i\sim q_\omega(z^i|\tau^i)}[\mathrm{ELBO}_{\theta,\omega}(\tau^i, z^i)]\right] \quad (8)$$

where $\mathrm{ELBO}_{\theta,\omega}(\tau^i, z^i)$ is defined as:

$$\sum_t \log\pi_\theta(a_t^i|s_t, z^i) - \log q_\omega(z^i|\tau^i) + \log p(z^i) \quad (9)$$

Given z^i, we can then optimize the first term in $\mathrm{ELBO}_{\theta,\omega}(\tau^i, z^i)$ with AIRL, which provides both the policy and the corresponding reward function as discussed next.

4.2 Multi-agent Adversarial Inverse Reinforcement Learning with Latent Variables

We proceed to propose an AIRL algorithm that maximizes the evidence lower bound objective in Equation 9. First, for the latent variable model q_ω, we introduce a inference network $q_\omega(z|\tau)$ that predicts the latent variable from trajectories. From Equation 9, this corresponds to the following objective:

$$\mathcal{L}_{q_\omega} = -\mathbb{E}_{\tau\sim\pi_E, z\sim q_\omega(z|\tau)}[\log q_\omega(z|\tau) - \log p(z)] \quad (10)$$

Then, conditioned on the latent variable z, we can transform the first term of Equation 9 using an AIRL approach. Here we need an additional discriminator $D_{\theta,\phi}(s, a, z)$ that depends on state s, action a and the latent variable z, whose goal is to discriminate generated trajectories and the demonstrations. Specifically, one provides a reward model $r_\phi(\tau, z)$, and the discriminator is denoted as:

$$D_{\theta,\phi}(s, a, z) = \frac{\exp(r_\phi(s, a))}{\exp(r_\phi(s, a)) + \pi_\theta(a|s)} \quad (11)$$

The discriminator then minimizes the following objective:

$$\mathcal{L}_D = -\mathbb{E}_{\tau_E\sim\pi_E, z_E\sim q_\omega(z|\tau_E)}[\log D_{\theta,\phi}(s, a, z_E)] \quad (12)$$
$$-\mathbb{E}_{\tau\sim\pi_\theta(z), z\sim\mathcal{N}(0,1)}[\log(1 - D_{\theta,\phi}(s, a, z))] \quad (13)$$
$$-\mathbb{E}_{\hat{\tau_E}\sim\pi_\theta(z_E), z_E\sim q_\omega(z|\tau_E)}[\log(1 - D_{\theta,\phi}(s, a, z_E))] \quad (14)$$

where the first term encourages higher $D_{\theta,\phi}$ for demonstrations, and the second and third term encourages lower $D_{\theta,\phi}$ for trajectories generated by the policy when the latent variables are sampled from $p(z)$ or inferred from demonstrations.

The learned policy $\pi_\theta(a|s, z)$ produces an action distribution based on the latent variable and the current state, and its primary objective is to reach higher $D_{\theta,\phi}$ values. We use $\pi_\theta(z)$ for the shorthand notation for the policy $\pi_\theta(a|s, z)$ with latent variable z. To encourage that the latent variable are informative for generating the trajectories, we add an reconstruction loss such that trajectories in τ_E could be reconstructed via $q_\omega(z|\tau_E)$ (encoder) and $\pi_\theta(a|s, z)$ (decoder), similar to InfoGAIL [28]. This leads to the following objective:

$$\mathcal{L}_G = -\mathbb{E}_{\tau\sim\pi_\theta(z), z\sim p(z)}[D_{\theta,\phi}(s, a, z)] \quad (15)$$
$$+\mathbb{E}_{\hat{\tau_E}\sim\pi_\theta(z_E), z_E\sim q_\omega(z|\tau_E), \tau_E\sim\pi_E}[\|\hat{\tau_E} - \tau_E\|_2] \quad (16)$$

Our full algorithm for IRL is shown in Algorithm 1, which perform iterative updates on the three objectives.

One important difference between our approach here and previous work in GAIL [16]/AIRL [12] is our approach to generating trajectories. Our policy is an autoregressive model from which we sample full trajectories using a deterministic dynamics model and optimize using the reparametrization trick. This approach is in contrast to generating trajectories using a policy acting within an environmental simulator and optimizing using policy gradients (e.g. PPO). We opted for this simpler approach in order to avoid the added instability of deep RL and because the focus of this work is new techniques for learning reward functions not policies.

Algorithm 1 AIRL with Latent Discrimination

Input: initial π_θ, r_ϕ, q_ω, learning rate α
Output: final π_θ, r_ϕ, q_ω
repeat
  Sample batch $\tau_E \sim \pi_E$, $S_{t_0} \sim P(s_{t_0})$
  Sample batch $z \sim p(z)$, $z_E \sim q_\omega(z|\tau_E)$
  Sample trajectories τ, $\hat{\tau}_E$ from π_θ with z, z_E
  Denote $D_{\theta,\phi}$ according to Equation 11
  $\mathcal{L}_D \leftarrow \log D_{\theta,\phi}(\tau_E, z_E) + \log(1 - D_{\theta,\phi}(\hat{\tau}_E, z_E)) + \log(1 - D_{\theta,\phi}(\tau, z))$
  $\mathcal{L}_G \leftarrow -\log D_{\theta,\phi}(\tau, z) + \|\hat{\tau_E} - \tau_E\|_2$
  $\mathcal{L}_Q \leftarrow \log q_\omega(z_E|\tau_E) - \log p(z_E)$

  $\theta \leftarrow \theta - \alpha\nabla_\theta\mathcal{L}_G$
  $\phi \leftarrow \phi + \alpha\nabla_\phi\mathcal{L}_D$
  $\omega \leftarrow \omega - \alpha\nabla_\omega\mathcal{L}_Q$
until converged

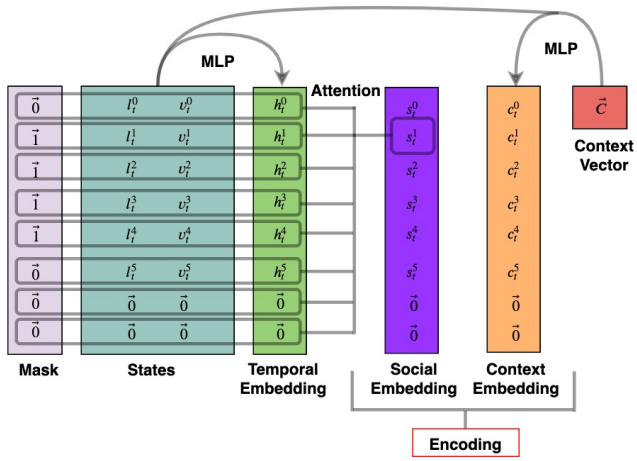

Figure 2: The encoding network used to capture contextual information. Masking and attention are key to scaling our learning techniques to settings with a large and variable number of agents.

4.3 State Encoder

All three parts of our model (π_θ, r_ϕ, and q_ω) use the encoding scheme depicted in Figure 2. Though each agent is provided with full observations of the environment, including position and velocity of each agent, the highly dynamic form of this state information necessitates a **permutation-invariant embedding** [27, 47], as the "birth-death" process will make it difficult to keep the same order for agent indices in the state representation.

Self-attention is the most sensible solution as it both resolves the technical challenges and fits with our intuitive notions of human decision making. For each agent (indexed by k), we calculate the social embedding

$$h_s^i = \sum_{j \neq k} w^{k \to j} h^{k \to j}$$

where $w^{k \to j}$'s are weights with $\sum_j w^{k \to j} = 1$ and $h^{k \to j}$ is the relative embedding of each agent. Although multi-head scaled dot-product attention has become the state-of-the-art solution in NLP [9], we found it added unnecessary overhead and instead opted for a simple scheme akin to the architectures in [7, 14, 36]. We calculate

$$\begin{aligned}
h^i_{\text{state}} &= e_s(s^i) \\
h^{k \to j} &= g_h(h^j_{\text{state}}, f(s^i, s^j)) \\
w^{k \to j} &= \text{softmax}_j(g_w(f(s^i, s^j)) \cdot (1 - m^j)) \\
f(s^i, s^j) &= \begin{cases} (\vec{l}^j - \vec{l}^i,\ \vec{v}^j - \vec{v}^i) & \text{when car} \\ (\vec{l}^j - \vec{l}_{\text{airport}},\ \vec{v}^j) & \text{when aircraft} \end{cases}
\end{aligned}$$

where $\vec{l}$, $\vec{v}$ are location and velocity vectors, e_s, g_h, and g_w are feed-forward neural networks and m^j is a masking bit indicating whether agent j is in the frame of observation.

In addition to the social embedding, each agent also receives a context embedding c^i obtained by applying a feed-forward network to s^i and a context vector $\vec{c}$). As the vessel for environment conditions independent of the agent, $\vec{c}$ comprises lane markings when modeling cars and weather conditions when modeling aircraft. At each timestep the three inputs to the encoder, E, are thus the record of all agent states s_t, the mask m_t, and context vector $\vec{c}$, and the output is the concatenation of h_s^i and h_c^i for each agent. The output encodings thus serve as the **permutation invariant state representation** for our policy, reward and inference networks.

4.4 Inference Network, Policy, and Reward Function

Once a permutation invariant representation for the state is found for each agent, we use these representations to model the policy, reward and the inference network.

Policy π_θ Specifically, the policy embeds sequential encodings with an LSTM and then a feed-forward network maps this embedding and the sampled latent variable z^i to output a_t^i.

Inference model q_ω The inference network employs a separate LSTM that embeds state encodings in reverse. The resulting embedding and z^i are then mapped to the mean and standard deviation of q_ω using a feed-forward network.

Reward r_ϕ Lastly, the reward function at each time t maps the encoding at time t and the encoding at time $t+1$ to a scalar using a final feed-forward network. In calculating the discriminator value,

$$D_{\theta,\phi}(s_t, a_t^i) = \frac{\exp(r_\phi(s_t, a_t^i))}{\exp(r_\phi(s_t, a_t^i)) + \pi_\theta(a_t^i | s)}$$

we calculate the action probability by taking the output of the policy to be encoding the mean of an isotropic Gaussian with unit variance. This approach is essentially the same as in other applications of AIRL except the trainable, non-unit variance used for exploration in RL settings is not necessary here.

5 RELATED WORK

Single-agent imitation learning and inverse reinforcement learning have been relatively well studied, with applications such as robotics [16] and driving [28] [24]. IL / IRL under multi-agent interactions are less considered, and most existing works assume specific reward structures, such as cooperative [3, 6, 25, 39], zero-sum [29], or linear rewards [33, 42]. Multi Agent-GAIL [38] and Multi Agent-AIRL [45] are multi-agent extensions of GAIL [16] and AIRL [12] which applies to Markov games without specific prior assumptions to the reward structure, but they are not scalable to real-world scenarios where there are many more agents in question. Attempts have been made to scale these technique to large driving datasets [5] where reward-shaping is utilized to aid convergence [4].

Recent work in multi-agent reinforcement learning have investigated cases where there are many agents in the system [20, 44] and / or the agents can enter or exit the environment at any time [18, 40] (birth-death processes). These cases, however, are less investigated in the imitation / inverse reinforcement learning; our work specifically discusses the imitation and IRL problems in large scale "birth-death" Markov games, which requires the algorithm to adapt to the number of agents.

The use of latent variables to learn multi-modal policies have been considered in single agent imitation learning, such as learning multi-modal policies [28] or meta-learning [46]. Our approach makes two distinctions from these previous ones: different from [28], our latent variables are continuous since there is no explicit prior about the number of different policies; different from [46], we assume a prior distribution on the latent variable so as to generate new policies unconditionally. These differences allow us to generate diverse multi-agent behaviors within the same environment; using latent variables would also decrease sample complexity as opposed to learning the policy of each agent separately.

6 EXPERIMENTS

6.1 Deterministic Transition Model

In the transportation environments, we assume that the state s_t contains the location $\vec{l}$ and velocities $\vec{v}$ of all agents:

$$s_t = \{s_t^i\}$$
$$s_t^i = \begin{cases} (\vec{l}_t, \vec{v}_t)^i & O_t^i = O[a] \\ s_O & \text{else} \end{cases}$$

where we consider $\vec{l}, \vec{v} \in \mathbb{R}^2$ for car trajectories (2d) and $\vec{l}, \vec{v} \in \mathbb{R}^3$ for airplane trajectories (3d). We assume that the action for each agent is to change its location and velocity (denoted as $\Delta\vec{l}$ and $\Delta\vec{v}$ respectively). The extra null state s_O and action a_O are taken to be values far from the data distribution such as the largest number possible in 64-bit floating point. The transition model of alive agents is defined as

$$T'((\vec{l}_{t+1}, \vec{v}_{t+1})^i \mid (\vec{l}_t,\ \vec{v}_t)^i,\ (\Delta\vec{l},\ \Delta\vec{v})_t^i) = (\vec{l}_t + (\Delta\vec{l})_t,\ \vec{v}_t + (\Delta\vec{v})_t)^i$$

which is deterministic. Learning $\Delta\vec{l}$ as well as $\Delta\vec{v}$ (instead of using second order ODEs) is useful in this case because it compensates for temporal downsampling of the data and discretization of time.

6.2 Data

HighD Dataset The HighD dataset [22] comprises over 100,000 short vehicle trajectories gathered from drone footage of German highways. We used the locations with 3 lanes (majority of dataset), and preprocessed by making all velocities positive and rescaling and shifting to a shared coordinate system. As lane markings differed slightly due to drone camera angle, we took the true location to be the average across all the recordings after scaling. To aid our optimization objective, we also downsampled the trajectories without lane changes so that the resulting trajectories were 30% lane changes. This prevented the highly biased dataset from biasing the generative model in the early phases of its training. Additionally, for convenience, we aggressively downsampled the trajectories in time, sacrificing a smoother dynamics model for a smaller computational burden.

As vehicles are not point particles in the HighD dataset and have lateral dimensions, we found training could be accelerated by including the relative locations of the edges making up each of the two vehicle's bounding boxes as inputs to the state encoder. Though not strictly necessary, this proved to aid convergence.

FAA Dataset The aircraft trajectory dataset consists of tracks gathered from the Federal Aviation Administration (FAA) multi-sensor fusion tracker [19]. The dataset contains six months of flights from locations in Central Florida, New York City, and Southern California, though for this work we only used flights to and from JFK airport in New York City. Raw tracks contain the lat long and altitude of each aircraft. These (x, y, z) points are smoothed using the optimization procedure described in [2] and approximate velocities are calculated by temporal differences. All trajectories are bounded at 40 km (~25 miles) from the airport laterally and 3 km (~10000 ft) in altitude. As with the HighD vehicle trajectories, we also temporally downsampled the aircraft trajectories which in their raw form often had sequence lengths on the order of hundreds or thousands. In order to provide as much contextual information to the model as possible, we also joined trajectory data with hourly historical weather data from the NOAA, which included wind speed, wind direction, and visibility, among other features.

6.3 Results

Learned Policies We used two types of metrics to evaluated the policies learned by our models. To measure how effectively the learned distribution can cover the expert distribution, we calculate the average and final displacement between sampled trajectories drawn from a particular starting state $s_{t_0}^i$ and the ground truth trajectory, holding the policies of the other agents fixed as expert. The reported number is the minimum over 10 sampled latent variables. Additionally, we measure "emergent" properties of the learned policies in the setting of multi-agent control. These emergent properties include the distribution of speeds, distance between agents, and rate of anomalous events (such as driving off the road or going to zero altitude far from the airport). For the HighD data distance between agents is reported as distance headway (DHW)–the distance to the preceding car in the lane if it exists–and time-to-collision (TTC) as these are the typical metrics used in driving simulation. The benchmark our models, we compare them with the policy with the LSTM removed as well as our full model with the latent variable z removed. Results for the HighD and FAA datasets are shown in Table 1. From these ablations, *we see that the proposed method is able to achieve low displacement from expert trajectory and emergent properties similar to those of expert trajectories.* The rate of bad final states is perhaps the most relevant among these emergent properties and this is minimal with the full model on both datasets.

As the learned latent variables are a focus of this work, we show visualizations of rollouts where latent variables are drawn from across the distribution. Figure 3 shows a distribution of highway driving trajectories. This visualization illustrates the primary role of the latent variable for the HighD dataset is modeling lane-changing behavior. Figure 4 shows a distribution of takeoff trajectories out of JFK airport. Here we see the latent variable captures the shape of the trajectory, which contains information about direction and altitude changes[2].

Learned Reward Functions

Verifying learned reward function in an IRL setting is challenging in general. In our setting, the challenge remains because we

[2]Video demonstrations of our learned policies in http://bit.ly/multi-agent-traffic.

Table 1: The two metrics used to benchmark learned policies. The top two metrics capture occupancy-measure matching for raw positions, while the bottom metrics capture matching of high-level features. For the final displacement metric on the FAA dataset, only takeoffs were considered, as all landings have basically the same final destination, biasing the metric in this case.

	HighD				FAA			
metric	w/o lstm & latent	w/o latent	w/ latent	expert	w/o lstm & latent	w/o latent	w/ latent	expert
avg. displacement (m)	5.89	5.03	**4.93**		823	670	**596**	
final displacement (m)	6.57	5.68	**5.61**		897	808	**785**	
min speed (m/s)	22.4	21.6	23.7	21.2	58.7	61.0	59.3	64.8
max speed (m/s)	44.8	40.3	37.2	39.5	310	296	302	287
rate bad final state (%)	2.6	1.15	1.24	0.0	7.52	6.01	5.32	0.0
avg. DHW (m)	51.3	58.2	63.8	56.7				
avg. TTC (s)	143	152	168	159				
avg. inter-agent dist. (m)					28000	26100	25200	24300

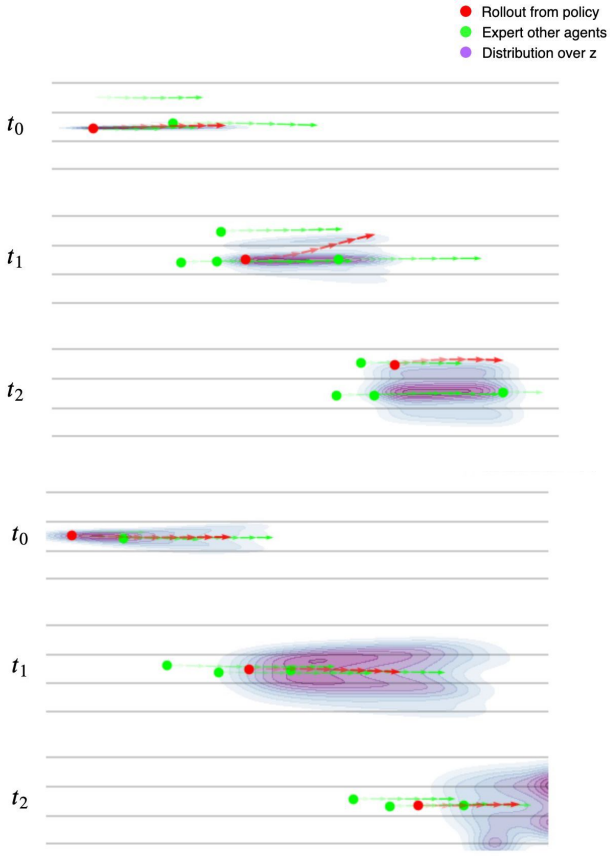

Figure 3: Expert trajectories are shown in green while a rollout for one value of z is shown in red. A full distribution of positions from trajectories resulting from many $z \sim \mathcal{N}(0, 1)$ are shown as a density. The latent variable primarily captures the driver's willingness to change lanes, with concentrations of density moving out to the adjacent lanes as well as behind the preceding car. Agents further away from the rolled out agent are not shown for visual clarity.

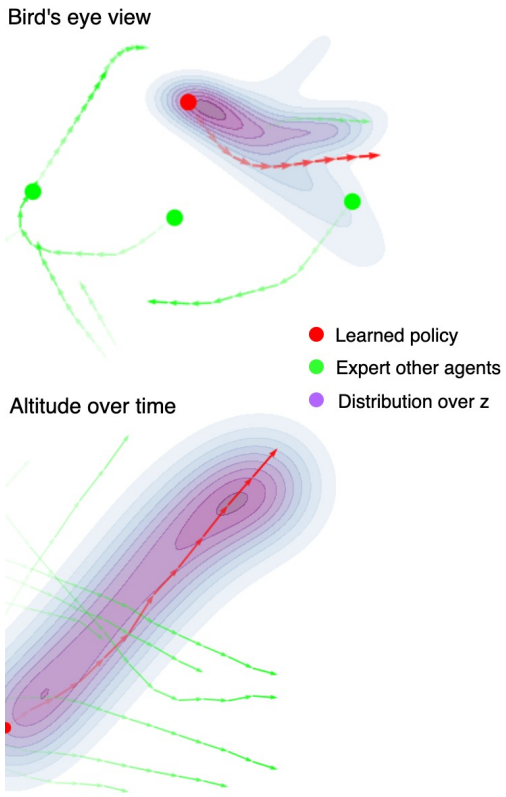

Figure 4: Expert trajectories are shown in green while a rollout for one value of z is shown in red. A full distribution of positions from trajectories resulting from many $z \sim \mathcal{N}(0, 1)$ are shown as a density. Here the latent variable captures the path of ascent chosen by the pilot.

do not have access to any ground truth rewards. Luckily, however, visualization in this setting is feasible and human intuition can be used for qualitatively validating the rewards. We visualize learned rewards by plotting the distribution of rewards around a vehicle.

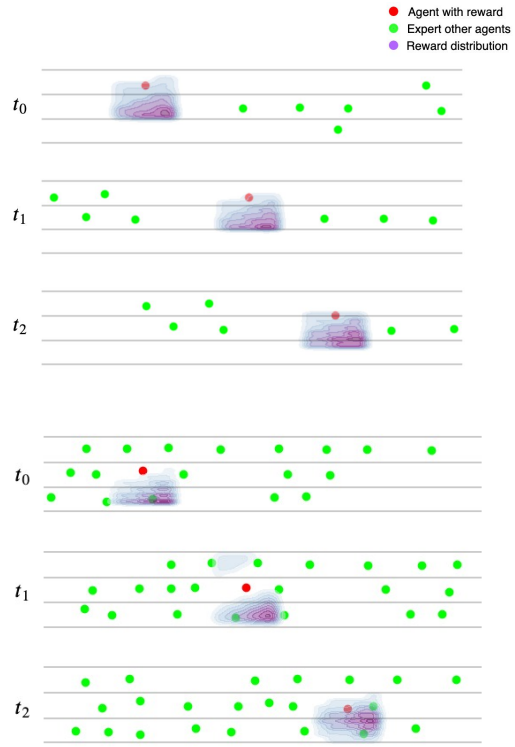

Figure 5: Visualization of a learned reward function when the state of the vehicle is rolled forward in time. Cars appear closer horizontally than they are in real life, as the true aspect ratio of the road is much higher.

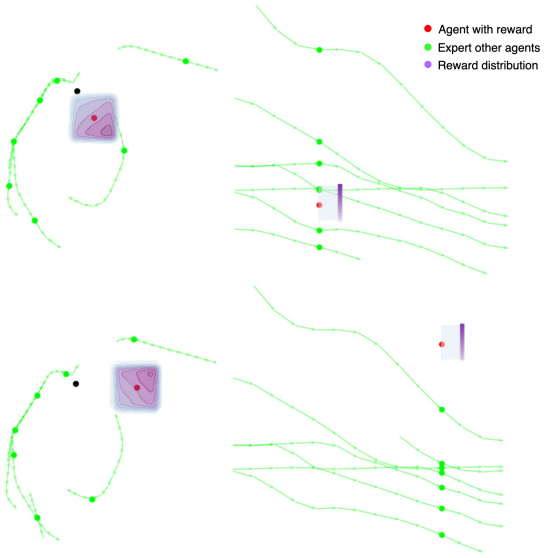

Figure 6: Visualization of reward function learned from the FAA dataset. Rewards are clearly correlated with the flight path of the pilot, in this case a looping ascent to the NE.

Specifically, we calculate the rewards over a grid of possible locations surrounding the vehicle, leaving the locations of the other vehicles fixed. Rewards are normalized between zero and one and the resulting density is smoothed with a Gaussian kernel.

Figure 5 shows two visualizations of reward functions learned from the HighD dataset. Both reward functions reflect a lane-change objective–one in less congestion and another in a higher level of congestion where rewards are more affected by the locations of other agents. Figure 6 shows a learned reward function on the FAA dataset. We see a similar reward function that reflects near-term navigational goals. In general, reward functions learned from the FAA dataset show less dramatic dependence on other agents, as we would expect given the implicit effect of terminal air traffic control.

7 SCOPE AND LIMITATIONS

High congestion Many of our simplifying assumptions degrade under conditions of higher congestion. Under these conditions, learned policies have trouble balancing goals as encoded by the latent variable with constraints imposed by other agents. As congestion increases, this balancing act becomes more challenging, especially when training data is biased towards less congestion.

Boundary conditions Behavior around the boundary of the environment is still very challenging to learn, due to residual effects of unobserved agents on those still under consideration. While our use of recurrent rollouts and a latent variable can compensate for some of these effects, it might be appropriate to model them more explicitly within a belief-state framework.

Reward vagueness Because the generator in our AIRL framework performs multi-objective optimization over a linear scalarization of VAE [21] and GAN [13] losses, trade-offs must be made by the model. Poor reward functions can result from local minima in which VAE losses are prioritized and the discriminator obtains poorer importance samples. In practice, this tends to result in more "vague" reward functions that lack desired constraints. This effect might be lessened by training on a purely adversarial objective such as with a BiGAN [10] in which there is a method of learning latent variables but only a single type of generator objective.

8 CONCLUSIONS

In this paper, we present an imitation learning / inverse reinforcement learning approach to "birth-death" Markov games that describes environments with a large, varying number of agents. By incorporating latent variables, we are able to learn policies and reward functions with different styles from demonstrations. We further introduce a permutation-invariant encoding that is suited for learning representations in our birth-death Markov games and other large-scale multi-agent environments. Empirical results on highway and flight control datasets demonstrate the validity with interpretable behavior and inferred rationales for agents. In future work, we are interested in considering more complex interactions between the actions of agents [43], such that the agents' actions are non-independent given the state.

REFERENCES

[1] Alexandre Alahi, Kratarth Goel, Vignesh Ramanathan, Alexandre Robicquet, Li Fei-Fei, and Silvio Savarese. 2016. Social lstm: Human trajectory prediction in crowded spaces. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 961–971.

[2] Shane T Barratt, Mykel J Kochenderfer, and Stephen P Boyd. 2018. Learning Probabilistic Trajectory Models of Aircraft in Terminal Airspace From Position Data. *IEEE Transactions on Intelligent Transportation Systems* (2018).

[3] Samuel Barrett, Avi Rosenfeld, Sarit Kraus, and Peter Stone. 2017. Making friends on the fly: Cooperating with new teammates. *Artificial Intelligence* 242 (2017), 132–171.

[4] Raunak P. Bhattacharyya, Derek J. Phillips, Changliu Liu, Jayesh K. Gupta, Katherine Rose Driggs-Campbell, and Mykel J. Kochenderfer. 2019. Simulating Emergent Properties of Human Driving Behavior Using Multi-Agent Reward Augmented Imitation Learning. *International Conference on Robotics and Automation, ICRA 2019, Montreal, QC, Canada, May 20-24, 2019*, 789–795.

[5] Raunak P Bhattacharyya, Derek J Phillips, Blake Wulfe, Jeremy Morton, Alex Kuefler, and Mykel J Kochenderfer. 2018. Multi-agent imitation learning for driving simulation. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 1534–1539.

[6] Kenneth Bogert and Prashant Doshi. 2014. Multi-robot inverse reinforcement learning under occlusion with interactions. In *Proceedings of the 2014 international conference on Autonomous agents and multi-agent systems*. 173–180.

[7] Nachiket Deo and Mohan M Trivedi. 2018. Convolutional social pooling for vehicle trajectory prediction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*. 1468–1476.

[8] Nachiket Deo and Mohan M Trivedi. 2018. Multi-modal trajectory prediction of surrounding vehicles with maneuver based lstms. In *2018 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 1179–1184.

[9] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* (2018).

[10] Jeff Donahue, Philipp Krähenbühl, and Trevor Darrell. 2016. Adversarial Feature Learning. *arXiv preprint arXiv:1605.09782* (May 2016). arXiv:cs.LG/1605.09782

[11] Chelsea Finn, Sergey Levine, and Pieter Abbeel. 2016. Guided Cost Learning: Deep Inverse Optimal Control via Policy Optimization. In *International Conference on Machine Learning*. 49–58.

[12] Justin Fu, Katie Luo, and Sergey Levine. 2017. Learning Robust Rewards with Adversarial Inverse Reinforcement Learning. *arXiv preprint arXiv:1710.11248* (2017).

[13] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *Advances in neural information processing systems*. 2672–2680.

[14] Agrim Gupta, Justin Johnson, Li Fei-Fei, Silvio Savarese, and Alexandre Alahi. 2018. Social gan: Socially acceptable trajectories with generative adversarial networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2255–2264.

[15] Peter Henderson, Riashat Islam, Philip Bachman, Joelle Pineau, Doina Precup, and David Meger. 2018. Deep reinforcement learning that matters. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

[16] Jonathan Ho and Stefano Ermon. 2016. Generative adversarial imitation learning. In *Advances in Neural Information Processing Systems*. 4565–4573.

[17] Ronald A Howard. 1960. Dynamic programming and markov processes. (1960).

[18] Max Jaderberg, Wojciech M Czarnecki, Iain Dunning, Luke Marris, Guy Lever, Antonio Garcia Castaneda, Charles Beattie, Neil C Rabinowitz, Ari S Morcos, Avraham Ruderman, Nicolas Sonnerat, Tim Green, Louise Deason, Joel Z Leibo, David Silver, Demis Hassabis, Koray Kavukcuoglu, and Thore Graepel. 2018. Human-level performance in first-person multiplayer games with population-based deep reinforcement learning. *arXiv preprint arXiv:1807.01281* (July 2018). arXiv:cs.LG/1807.01281

[19] Chris Jagodnik, Joseph Stella, and Dan Varon. 2008. Fusion tracking in air traffic control. *Journal of Air Traffic Control* 50, 1 (2008).

[20] Kathy Jang, Eugene Vinitsky, Behdad Chalaki, Ben Remer, Logan Beaver, Andreas A Malikopoulos, and Alexandre Bayen. 2019. Simulation to scaled city: zero-shot policy transfer for traffic control via autonomous vehicles. In *Proceedings of the 10th ACM/IEEE International Conference on Cyber-Physical Systems*. ACM, 291–300.

[21] Diederik P Kingma and Max Welling. 2013. Auto-Encoding Variational Bayes. *arXiv preprint arXiv:1312.6114v10* (December 2013). arXiv:stat.ML/1312.6114v10

[22] Robert Krajewski, Julian Bock, Laurent Kloeker, and Lutz Eckstein. 2018. The highD Dataset: A Drone Dataset of Naturalistic Vehicle Trajectories on German Highways for Validation of Highly Automated Driving Systems. In *2018 IEEE 21st International Conference on Intelligent Transportation Systems (ITSC)*.

[23] Henrik Kretzschmar, Markus Spies, Christoph Sprunk, and Wolfram Burgard. 2016. Socially compliant mobile robot navigation via inverse reinforcement learning. *The International Journal of Robotics Research* 35, 11 (2016), 1289–1307.

[24] Alex Kuefler, Jeremy Morton, Tim Wheeler, and Mykel Kochenderfer. 2017. Imitating driver behavior with generative adversarial networks. In *2017 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 204–211.

[25] Hoang M Le, Yisong Yue, and Peter Carr. 2017. Coordinated Multi-Agent Imitation Learning. *arXiv preprint arXiv:1703.03121* (March 2017). arXiv:cs.LG/1703.03121

[26] Namhoon Lee, Wongun Choi, Paul Vernaza, Christopher B Choy, Philip HS Torr, and Manmohan Chandraker. 2017. Desire: Distant future prediction in dynamic scenes with interacting agents. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 336–345.

[27] Yangyan Li, Rui Bu, Mingchao Sun, Wei Wu, Xinhan Di, and Baoquan Chen. 2018. Pointcnn: Convolution on x-transformed points. In *Advances in Neural Information Processing Systems*. 820–830.

[28] Yunzhu Li, Jiaming Song, and Stefano Ermon. 2017. Infogail: Interpretable imitation learning from visual demonstrations. In *Advances in Neural Information Processing Systems*. 3812–3822.

[29] Xiaomin Lin, Peter A Beling, and Randy Cogill. 2014. Multi-agent inverse reinforcement learning for zero-sum games. *arXiv preprint arXiv:1403. 6508* (2014).

[30] Michael L Littman. 1994. Markov games as a framework for multi-agent reinforcement learning. In *Proceedings of the eleventh international conference on machine learning*, Vol. 157. 157–163.

[31] D Michie, M Bain, and J Hayes-Miches. 1990. Cognitive models from subcognitive skills. *IEE control engineering series* 44 (1990), 71–99.

[32] Andrew Y Ng, Stuart J Russell, et al. 2000. Algorithms for inverse reinforcement learning.. In *Icml*. 663–670.

[33] Tummalapalli Sudhamsh Reddy, Vamsikrishna Gopikrishna, Gergely Zaruba, and Manfred Huber. 2012. Inverse reinforcement learning for decentralized non-cooperative multiagent systems. In *Systems, Man, and Cybernetics (SMC), 2012 IEEE International Conference on*. 1930–1935.

[34] Andrey Rudenko, Luigi Palmieri, Michael Herman, Kris M Kitani, Dariu M Gavrila, and Kai O Arras. 2019. Human Motion Trajectory Prediction: A Survey. *arXiv preprint arXiv:1905.06113* (2019).

[35] Stuart Russell. 1998. Learning agents for uncertain environments. In *Proceedings of the eleventh annual conference on Computational learning theory*. ACM, 101–103.

[36] Amir Sadeghian, Vineet Kosaraju, Ali Sadeghian, Noriaki Hirose, Hamid Rezatofighi, and Silvio Savarese. 2019. Sophie: An attentive gan for predicting paths compliant to social and physical constraints. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 1349–1358.

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

[38] Jiaming Song, Hongyu Ren, Dorsa Sadigh, and Stefano Ermon. 2018. Multi-agent generative adversarial imitation learning. In *Advances in Neural Information Processing Systems*. 7461–7472.

[39] Adrian Šošić, Wasiur R KhudaBukhsh, Abdelhak M Zoubir, and Heinz Koeppl. 2017. Inverse reinforcement learning in swarm systems. In *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems*. International Foundation for Autonomous Agents and Multiagent Systems, 1413–1421.

[40] Joseph Suarez, Yilun Du, Phillip Isola, and Igor Mordatch. 2019. Neural MMO: A Massively Multiagent Game Environment for Training and Evaluating Intelligent Agents. *arXiv preprint arXiv:1903.00784* (2019).

[41] Anirudh Vemula, Katharina Muelling, and Jean Oh. 2018. Social attention: Modeling attention in human crowds. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 1–7.

[42] Kevin Waugh, Brian D Ziebart, and J Andrew Bagnell. 2013. Computational Rationalization: The Inverse Equilibrium Problem. *arXiv preprint arXiv:1308.3506* (August 2013). arXiv:cs.GT/1308.3506

[43] Yaodong Yang, Rui Luo, Minne Li, Ming Zhou, Weinan Zhang, and Jun Wang. 2018. Mean field multi-agent reinforcement learning. *arXiv preprint arXiv:1802.05438* (2018).

[44] Yaodong Yang, Lantao Yu, Yiwei Bai, Ying Wen, Weinan Zhang, and Jun Wang. 2018. A Study of AI Population Dynamics with Million-agent Reinforcement Learning. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*. International Foundation for Autonomous Agents and Multiagent Systems, 2133–2135.

[45] Lantao Yu, Jiaming Song, and Stefano Ermon. 2019. Multi-Agent Adversarial Inverse Reinforcement Learning. *arXiv preprint arXiv:1907.13220* (2019).

[46] Lantao Yu, Tianhe Yu, Chelsea Finn, and Stefano Ermon. 2019. Meta-Inverse Reinforcement Learning with Probabilistic Context Variables. *NeurIPS* (2019).

[47] Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Ruslan R Salakhutdinov, and Alexander J Smola. 2017. Deep sets. In *Advances in neural information processing systems*. 3391–3401.

[48] Eric Zhan, Stephan Zheng, Yisong Yue, Long Sha, and Patrick Lucey. 2018. Generating Multi-Agent Trajectories using Programmatic Weak Supervision. *arXiv preprint arXiv:1803.07612* (2018).

[49] Brian D Ziebart, Andrew L Maas, J Andrew Bagnell, and Anind K Dey. 2008. Maximum Entropy Inverse Reinforcement Learning. In *AAAI*, Vol. 8. 1433–1438.