Multi-Agent Imitation Learning for Driving Simulation

Raunak P. Bhattacharyya

***Abstract*— Simulation is an appealing option for validating the safety of autonomous vehicles. Generative Adversarial Imitation Learning (GAIL) has recently been shown to learn representative human driver models. These human driver models were learned through training in single-agent environments, but they have difficulty in generalizing to multi-agent driving scenarios. We argue these difficulties arise because observations at training and test time are sampled from different distributions. This difference makes such models unsuitable for the simulation of driving scenes, where multiple agents must interact realistically over long time horizons. We extend GAIL to address these shortcomings through a parameter-sharing approach grounded in curriculum learning. Compared with single-agent GAIL policies, policies generated by our PS-GAIL method prove superior at interacting stably in a multi-agent setting and capturing the emergent behavior of human drivers.**

I. INTRODUCTION

Validating the safety of autonomous vehicles represents an unsolved yet crucial challenge for regulators and manufacturers alike. Autonomous driving systems are typically evaluated on real-world drive tests, which are expensive, time-consuming, and potentially dangerous. Furthermore, it is likely infeasible to build a statistically significant case for the safety of a system solely through real-world testing [1], [2]. Validation through simulation provides a promising alternative to real-world testing, with the ability to evaluate vehicle performance in large numbers of scenes safely and economically. Simulations must accurately reflect real-world driving to be useful, and therefore require realistic models of human drivers to govern the behavior of non-autonomous vehicles that occupy the roadway.

Imitation learning (IL) represents a promising avenue for learning models of driver behavior from real-world data. Behavioral cloning, a variant of imitation learning that relies on a supervised learning procedure, is easy to implement but tends to perform poorly in practice due to a problem known as covariate shift [3]. Inverse Reinforcement Learning (IRL) approaches formulate the imitation learning task as a Markov Decision Process (MDP) with a stationary dynamics model, which can be solved approximately using batch policy optimization [4]. IRL addresses the covariate shift problem by learning a cost function and allowing the agent to interact with the environment while training. As a result, the agent encounters similar states during training and testing, which in practice enables the agent to better learn the consequences of—and correct for—its mistakes.

Imitation learning approaches have been shown to successfully learn human driving policies for individual vehicles in car-following and highway-driving contexts [5], [6]. However, when evaluated in a multi-agent setting, the policies learned through single-agent imitation learning fail to exhibit realistic behavior, rendering them inadequate for use in simulation. As we argue later, this deterioration in performance occurs because transitioning from single-agent to multi-agent settings effectively reintroduces the covariate shift problem.

We derive an algorithm that addresses the deficiencies of single-agent imitation learning for learning human driver models. We extend Generative Adversarial Imitation Learning (GAIL) [7] and Parameter Sharing Trust Region Policy Optimization (PS-TRPO) [8] to enable imitation learning in the multi-agent context, yielding a new algorithm called PS-GAIL. PS-GAIL generates policies capable of controlling multiple vehicles simultaneously, enabling the simulation of complex roadway scenes.

The effectiveness of the proposed PS-GAIL method is demonstrated by comparing the performance of its learned policies to single-agent policies learned through GAIL. PS-GAIL policies are shown to generate driving trajectories that better match those of human drivers. Furthermore, vehicles driven by PS-GAIL policies are shown to interact with each other in a more stable manner over long horizons, and they are less prone to the collisions and off-road events that often arise during interactions between single-agent GAIL policies.

II. BACKGROUND

A. Markov Decision Processes

An infinite horizon MDP is defined by the tuple $(\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \gamma)$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $\mathcal{T}$ is the transition model, $\mathcal{R}$ is the reward function, and γ is the discount factor. The reward function $\mathcal{R}$ provides the rewards received while interacting in the environment, where $\mathcal{R}(s, a, s')$ denotes the reward for transitioning from s to s' when action a is taken. The transition model $\mathcal{T}(s' \mid s, a)$ gives the probability over next states given action a taken in state s. The discount factor governs how much future rewards are valued relative to immediate rewards.

A stochastic policy, $\pi : \mathcal{S} \rightarrow P(\mathcal{A})$, maps each state to the probability of taking each action. The sum of discounted rewards, or return, from state s_t is defined as $g_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1}$, where t is a time index, and r_t is the corresponding reward. The objective in an MDP is to find a policy that maximizes the expected return, or value, of each state $V_\pi(s) = \mathbb{E}_\pi[g_t \mid s_t = s]$.

B. Imitation Learning

The goal in imitation learning (IL) is to learn a policy π that imitates an expert policy π_E given demonstrations from that expert [9], [10]. A demonstration is defined as a sequence of state-action pairs that result from a policy

interacting with the environment: $d = \{s_1, a_1, s_2, a_2, \ldots\}$. The reward function is typically assumed to be unknown.

The state-occupancy distribution is defined as:

$$\rho_\pi(s) = (1-\gamma)\sum_{t=0}^{\infty}\gamma^t p(s_t = s \mid \pi), \tag{1}$$

which gives the average discounted probability of the agent being in state s. The supervised learning approach to imitation learning, behavioral cloning (BC), learns a policy by minimizing some loss function ℓ over the set of demonstrations with respect to the policy [10]:

$$\pi_{sup} = \underset{\pi}{\operatorname{argmin}} \underset{s\sim\rho_{\pi_E}}{\mathbb{E}}[\ell(\pi, s)]. \tag{2}$$

During training, behavioral cloning samples states from the state-occupancy distribution of the expert, ρ_{π_E}. However, when interacting with the environment, the policy samples states from the state-occupancy distribution of the learned policy, $\rho_{\pi_{sup}}$. This change in distribution between training and test time is referred to as covariate shift [3], and results in the agent making cascading errors.

Allowing the agent to interact with the environment during training time addresses the underlying cause of covariate shift, but this interaction requires a reward function since the agent may encounter states not contained in the training data. We focus on Generative Adversarial Imitation Learning (GAIL) due to its scalability, low sample-complexity as measured in expert demonstrations [7], and previous success in learning human driver models [6].

GAIL formulates imitation learning as the problem of matching the state-action occupancy distribution of the expert policy. Ho et al. [7] show that generative adversarial networks can be used to approximately accomplish this task. A discriminator D_ψ, parametrized by ψ learns to distinguish expert from non-expert behavior, while a policy π_θ, parameterized by θ attempts to emulate that behavior. In the case where D_ψ represents the probability the state-action pair came from π_E, the GAIL objective is given by [11]:

$$\begin{aligned}\min_\theta \max_\psi \; & \mathbb{E}_{\pi_E} \log D_\psi(s,a) + \\ & \mathbb{E}_{\pi_\theta} \log(1 - D_\psi(s,a)).\end{aligned} \tag{3}$$

Minimizing different divergences between state-action occupancy distributions yields different objectives. We use the Wasserstein distance [12] because it mitigates the vanishing gradient problem observed when minimizing Jensen-Shannon divergence [12]. The objective becomes:

$$\min_\theta \max_\psi \; \mathbb{E}_{\pi_E}[D_\psi(s,a)] - \mathbb{E}_{\pi_\theta}[D_\psi(s,a)], \tag{4}$$

where the critic, D, learns to output a high score when encountering pairs from π_E, and a low score when encountering generated pairs. The output of the critic $D_\psi(s,a)$ is used as a surrogate reward whose value grows larger as actions sampled from π_θ look similar to those chosen by experts.

After performing rollouts with a given set of policy parameters θ, surrogate rewards $\tilde{r}(o,a;\psi)$ are calculated and Trust Region Policy Optimization (TRPO) [13] is used to perform a policy update. TRPO is used because it can better handle the high variance of the non-stationary reward from the critic. Although $\tilde{r}(o,a;\psi)$ may be quite different from the true reward function optimized by experts, it can be used to drive π_θ into regions of the state-action space similar to those explored by π_E.

GAIL effectively addresses the problem of covariate shift provided that the training and testing environments are identical; however, this is not the case when learning human driver models in a single-agent setting and deploying them in a multi-agent setting. During test time, the policy observes nearby vehicles acting differently than during training, and again makes small errors that compound over time. In this paper, we explicitly formulate learning human driver models as a multi-agent problem in order to address this discrepancy.

C. Multi-agent reinforcement learning

Centralized multi-agent RL traditionally requires learning a policy π mapping from a joint observation space $\mathcal{O} \in \mathbb{R}^{O\times M}$ to action space $\mathcal{A} \in \mathbb{R}^{A\times M}$. Here O is the dimensionality of a single observation, M is the number of agents, and A is the dimensionality of a single action. Centralized approaches grow rapidly in computational complexity as the number of agents increases. Provided that the pool of agents is homogeneous (i.e., each policy must perform essentially the same mapping, $\mathcal{O} \to \mathcal{A}$), the joint distribution $\pi = p(a_0, \ldots, a_M \mid o_0, \ldots, o_M)$ can be factorized as $\prod_i p(a_i \mid o_i)$, which scales linearly with number of agents.

Gupta, Egorov, and Kochenderfer introduced an algorithm called Parameter Sharing Trust Region Policy Optimization (PS-TRPO), which is a policy gradient approach that combines parameter sharing and TRPO [8]. PS-TRPO was shown to produce decentralized parameter sharing neural network policies that exhibit emergent cooperative behavior without explicit communication between agents. PS-TRPO is highly sample-efficient because it reduces the number of parameters by a factor of M, and shares experience across all agents. Notably, PS-TRPO still allows agents to exhibit different behavior because each agent receives unique observations.

For a policy π_θ with parameters θ, PS-TRPO performs an update to the policy parameters by approximately solving the constrained optimization problem:

$$\begin{aligned}\underset{\theta}{\text{maximize}} \quad & \mathbb{E}_{o,a\sim\pi_{\theta_k}}\left[\frac{\pi_\theta(a \mid o)}{\pi_{\theta_k}(a \mid o)} A_{\theta_k}(o,a)\right] \\ \text{subject to} \quad & \mathbb{E}_o[D_{KL}(\pi_{\theta_k}(\cdot \mid o) \| \pi_\theta(\cdot \mid o))] \le \Delta_{KL},\end{aligned} \tag{5}$$

where π_{θ_k} is a rollout-sampling policy and $A_{\theta_k}(o,a)$ is an advantage function quantifying how much the value of an action a taken in response to an observation o differs from the baseline value estimated for o. D_{KL} is the KL divergence between the two policy distributions, and Δ_{KL} is a step size parameter.

III. APPROACH

We propose an extension to GAIL enabling the simultaneous control of multiple human driver models. The following subsections describe the multi-agent driving problem, and detail our approach for combating its associated challenges.

Algorithm 1 PS-GAIL

Input: Expert trajectories $\tau_E \sim \pi_E$, Shared policy parameters Θ_0, Discriminator parameters ψ_0, Trust region size Δ_{KL}
for $k \leftarrow 0, 1, \ldots$ **do**
  Rollout trajectories for all agents $\vec{\tau} \sim \pi_{\theta_k}$
  Score $\vec{\tau}$ with critic, generating reward $\tilde{r}(s_t, a_t; \psi_k)$
  Batch trajectories obtained from all the agents
  Take a TRPO step to find $\pi_{\theta_{k+1}}$ maximizing Eq. (5)
  Update the critic parameters ψ by maximizing Eq. (4)
end for

A. Problem Formulation

In line with recent work in multi-agent imitation learning [14], we formulate multi-agent driving as a Markov game [15] consisting of M agents and an unknown reward function. We make three simplifying assumptions:

1) **Homogeneous agents**: agents have the same observation and action spaces:
$$\mathcal{O}_i = \mathcal{O}_j \text{ and } \mathcal{A}_i = \mathcal{A}_j \ \forall \text{ agents } i, j.$$
2) **Independent rewards**: the reward function is not shared; it depends only on the action of each agent and the state, and not on the actions of other agents or the next state. In particular, agents are not cooperative:
$$\mathcal{R}_i(s, a_1, \ldots, a_i, \ldots, a_k) = \mathcal{R}_i(s, a_i).$$
3) **Identical reward function**: the reward function is the same for all agents:
$$\mathcal{R}_i = \mathcal{R}_j \ \forall \text{ agents } i, j.$$

These assumptions are idealizations and do not hold for real-world driving scenes. For example, different vehicles may permit different accelerations, a driver may only want to change lanes if other drivers are not doing so, and individuals may value different driving qualities such as smoothness or proximity to other vehicles differently. Nevertheless, these assumptions often do apply approximately, and, as we later show, allow for learning of realistic driving policies.

B. Parameter Sharing GAIL

A naive approach to learning human driver policies would be to train a policy in an environment where it controls a single vehicle on the roadway and all remaining vehicles follow a predetermined trajectory. Unfortunately, this approach is often incapable of producing policies that can reliably control many vehicles on the same roadway. By introducing such a controller to other vehicles after training, we reintroduce covariate shift. As a result, small errors in the behavior of a single vehicle can destabilize neighboring vehicles, ultimately leading to the failure of many agents in the scene.

Our proposed approach, PS-GAIL, combines GAIL with PS-TRPO to generate policies capable of driving multiple vehicles, enabling more stable simulation of entire road scenes. Algorithm 1 describes the PS-GAIL approach. We initialize the shared parameters of the policy and select a step size parameter. At each iteration of the algorithm, the policy with shared parameters is used by each agent to generate trajectories. Rewards are then assigned to each state-action pair in these trajectories by the critic. Subsequently observed trajectories are used to perform a TRPO update for the policy, and an Adam update for the critic. PS-GAIL can be viewed as a special case of the algorithms presented by Song et al. [14]. In particular, in PS-GAIL all agents share the same policy and receive rewards from the same critic.

We represent the policy with a recurrent neural network due to the high-dimensional observation space, nonlinearity required in the mapping from observations to actions, and partial observability of the local driving scene. Partial observability arises from (i) sensor noise and occlusions and (ii) unobserved driver latent state in the form of behavioral traits and intended maneuvers.

Our training procedure must also account for non-stationary environment dynamics. In the multi-agent setting, the dynamics of the environment change along with the agent policies. We mitigate this problem by introducing a curriculum, $\mathcal{C}$, which scales the difficulty of the multi-agent learning problem during training. Gupta *et al.* define a multi-agent curriculum as a multinomial distribution over the number of agents controlled by the policy each episode: $\mathcal{C} \sim \text{Multi}(M, \vec{p})$ [8]. The curriculum gradually shifts probability mass to larger numbers of agents. We use a simplified curriculum that increments the number of controlled agents by a fixed number every K iterations during training.

IV. IMPLEMENTATION

A. Simulator

In order to learn the policy in an environment with human drivers, we use a simulator that allows for playing back real trajectories and simulating the movement of controlled vehicles given actions selected by a policy. This process proceeds as follows:

1) The initial scene state is sampled from a dataset of real driver trajectories. This state includes the position, orientation, and velocity of all vehicles in the scene. The trajectory data we use is from the Next-Generation Simulation (NGSIM) dataset. NGSIM contains highway driving trajectories for US Highway 101 [16] and Interstate 80 [17], and consists of 45 minutes of driving at 10 Hz for each roadway.
2) A subset of the vehicles in the scene are randomly selected to be controlled by the policy. For single-agent training only one vehicle is selected, whereas for multi-agent training M vehicles are controlled by the policy.
3) For each vehicle, a set of features are extracted and passed to the policy as the observation. Table I describes the features provided to the policy.
4) The policy outputs longitudinal acceleration and turn-rate values as the vehicle action. These values are used to propagate the vehicle forward in time.
5) This process repeats for a horizon of 200 timesteps at 10 Hz, corresponding to 20 s of driving per episode.

TABLE I: Observation features

Feature	Description
LIDAR Range and Range Rate	20 artificial LIDAR beams output in regular polar intervals, providing the relative position and velocity of intercepted objects.
Ego Vehicle	Lane-relative velocity, heading, offset. Vehicle length and width. Lane curvature, distance to left and right lane makers and road edges.
Temporal	Longitudinal and lateral acceleration, local and global turn and angular rate, timegap, and time-to-collision.
Indicators	Collision occurring, ego vehicle out-of-lane, and negative velocity.
Leading Vehicle	Relative distance, velocity, and absolute acceleration of vehicle in front of fore vehicle, if it exists.

B. Policies

We use recurrent neural network (RNN) policies, in all cases consisting of 64 Gated Recurrent Units (GRUs). The observation is passed directly into the RNN without any initial reduction in dimensionality. We use recurrent policies in order to address the partial observability of the state caused by occluded vehicles. In the multi-agent setting, a single shared policy selects actions for all vehicles, following the parameter sharing approach previously described. Policy optimization is performed using an implementation of TRPO from rllab [18], with a step size of 0.1.

We use two training phases for all of the models. The first phase consists of 1000 iterations with a low discount of 0.95 and a small batch size of 10 000 observation-action pairs. The second phase fine-tunes the models, running for 200 iterations with a higher discount of 0.99 and larger batch size of 40 000. For the multi-agent model, we add 10 agents to the environment every 200 iterations of the first training phase. We use 100 agents in the fine-tune phase for the multi-agent GAIL models.

C. Critic

The critic acts as the surrogate reward function in the environment. The observation-action pairs for each vehicle at each timestep are passed to the critic, which outputs a scalar value that is then used as the reward for that vehicle. The critic is implemented as a feed-forward neural network consisting of (128,128,64) ReLU units. We implemented the critic as a Wasserstein GAN with gradient penalty (WGAN-GP) with a gradient penalty of 2 [19]. Similarly to Li et al. [20], we used a replay memory for the critic in order to stabilize training, which contains samples from the three most recent epochs. For each training epoch of the policy, the critic is trained for 40 epochs using the Adam optimizer [21] with a learning rate of 0.0004, dropout probability of 0.2, and batch size of 2000. Half of each batch consists of NGSIM data, with the remaining half comprised of data from policy rollouts. Finally, the reward values output from the critic are adaptively normalized to have zero mean and unit variance prior to being passed to TRPO.

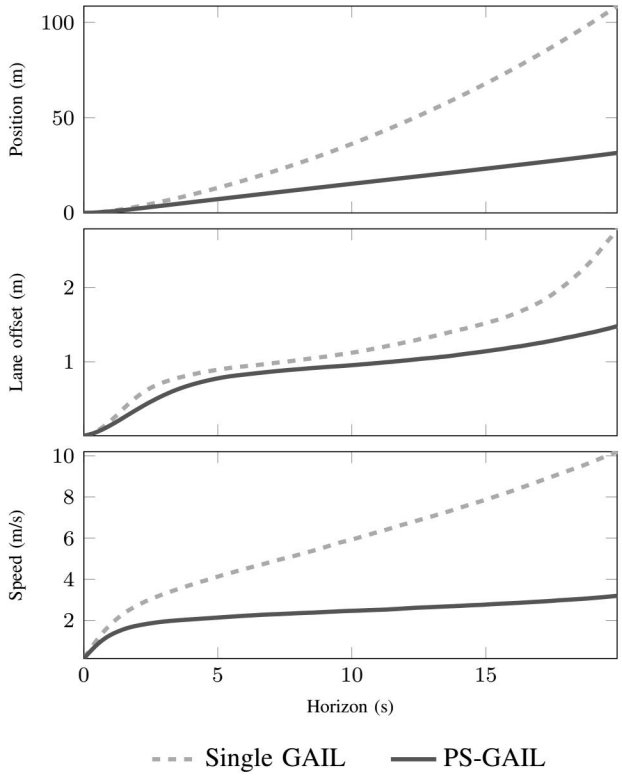

Fig. 1: Root weighted square error vs. prediction horizon for single and multi-agent models. The policy trained in a multi-agent setting more closely resembles the human driver, particularly as the horizon increases.

Because the critic is a feed-forward network and operates on the observations of each vehicle, it is not able to account for the partial observability of the scene. A recurrent critic could address this issue, and has been used in other partially observable settings [22]. We nevertheless use a feed-forward critic because when using a recurrent critic one must decide when to output the reward–at each timestep or only at the end of the sequence. In the former case, the critic outputs large negative values at the beginning of a sequence when an agent closely resembles a real driver (i.e., it overfits), which makes learning difficult. In the latter case, the agent is faced with an extremely challenging credit assignment problem, receiving feedback on 200 timesteps of actions only at the terminal state. Li *et al.* provide a solution for the every-step reward problem, but it entails a highly computationally expensive Monte Carlo sampling procedure [22]. In practice, the feed-forward critic was sufficient to learn driving policies.

V. EXPERIMENTS

We use GAIL and multi-agent GAIL to learn policies for two-dimensional highway driving and compare their performance. We train three of each model and average the results of 10 000 policy rollouts in a 100-agent environment. We present the average results over these rollouts.

A. Evaluation Metrics

The trajectories generated by the policies are evaluated against human driving data using Root Mean Square Error

(RMSE) and emergent behavior. For each of m trajectories, we sample a single rollout and compute the RMSE of each predicted variable v:

$$\text{RMSE} = \sqrt{\frac{1}{m}\sum_{i=1}^{m}\left(v_t^{(i)} - \hat{v}_t^{(i)}\right)^2}, \tag{6}$$

where $v_t^{(i)}$ is the true value in the ith trajectory at time horizon t and $\hat{v}_t^{(i)}$ is the simulated value for the ith trajectory at time horizon t. We extract the RMSE in predictions of global position, lane offset, and speed over time horizons up to 20 s. We used a single trajectory because additional trajectories did not significantly impact the overall RMSE value.

Emergent behavior was captured by extracting emergent features such as offroad duration, collision rate and hard brake rate. We calculate these rates by finding the amount of times a particular constraint is satisfied, and dividing by the total number of instances. For offroad duration, our constraint is when the vehicle is more than 1 m off one edge of the road. Collision rate is when there is a collision, which is easily accessible as one of our features. The threshold for a hard brake event is $-3\,\text{m}\,\text{s}^{-1}$.

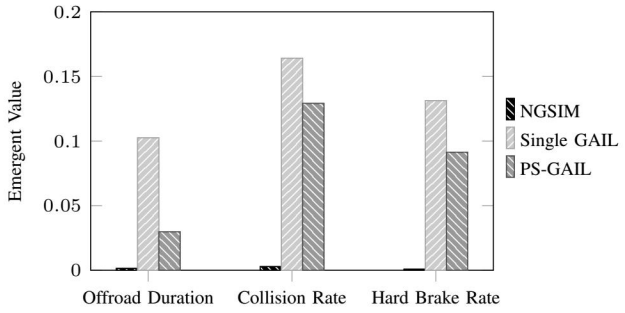

Fig. 2: Emergent values for each model.

B. Results and Discussion

Fig. 1 shows root mean square error results for prediction horizons up to 20 s. These plots indicate that PS-GAIL captures expert behavior more faithfully than single-agent GAIL. This performance discrepancy is especially pronounced for longer prediction horizons, where the errors for single-agent policies begin to accumulate rapidly.

The superior performance of PS-GAIL is further illustrated by Fig. 2. These validation results empirically demonstrate that PS-GAIL policies are less likely to lead vehicles into collisions, extreme decelerations, and off-road driving. This serves as further illustration that the PS-GAIL training procedure encourages stabler interactions between agents, thereby making them less likely to encounter extreme or unlikely driving situations.

The difficulty of the multi-agent task scales with the number of agents controlled in the environment. At what number of controlled agents does the single-agent policy deteriorate beyond a usable point? We address this question in our third set of results. Fig. 3 shows the performance of the two models as a function of the number of agents controlled in the environment. For each number of agents, that many agents are replaced in the environment with the policy, while the remaining agents are left as originally recorded in NGSIM. The results indicate that while the single-agent policy deteriorates rapidly, the multi-agent policy declines in performance much more gradually.

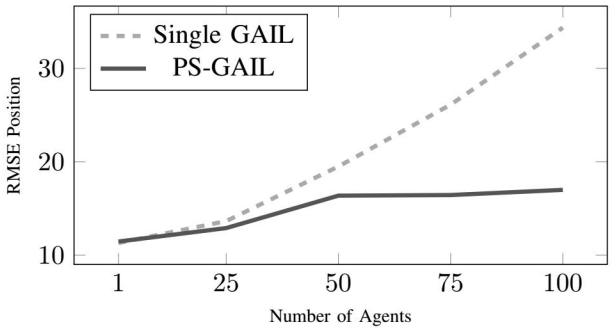

Fig. 3: This plot shows the average RMSE value across all timesteps of an episode as a function of the number of controlled agents. As the policy controls more vehicles, single-agent GAIL performance deteriorates rapidly, while PS-GAIL worsens at a much lower rate.

VI. CONCLUSIONS

Validating the safety of autonomous vehicles in the real world is costly, dangerous, and time consuming. Performing this validation in simulated environments would help address these problems, but requires a highly realistic simulator. This research focused on building human driver models capable of interacting with other controlled vehicles in a manner representative of real data.

We proposed to train these models in a multi-agent setting using a parameter sharing approach that addresses scaling issues associated with multi-agent learning. Furthermore, we addressed challenges of instability in the learning environment through the use of a training curriculum. Experiments comparing the performance of this multi-agent model with existing single-agent models indicated that the former exhibits significantly more realistic behavior, particularly over longer time horizons.

Future work will investigate methods for improving model performance, and applying learned driver models. Three potential methods for improving model performance are (i) reward augmentation, (ii) applying learning algorithms that encourage more diverse behavior [11], and (iii) using a recurrent critic in order to account for partial observability. Ultimately, the goal in learning human driver models is to validate autonomous vehicles in simulation, and we hope to apply these models to that end in the future.

The code associated with this project and relevant videos can be found at https://github.com/sisl/ngsim_env.

ACKNOWLEDGMENTS

This project is a subset of research performed for the Stanford Intelligent Systems Laboratory (SISL) under the supervision of Professor Mykel Kochenderfer. The author thanks Professor Kochenderfer for his guidance. The author also thanks Derek Phillips, Blake Wulfe, Jeremy Morton and Alex Kuefler for their continued support. Finally, the author thanks Zachary Barnes for his mentorship and great ideas.

REFERENCES

[1] ISO, "ISO 26262: Road vehicles-functional safety," *International Standard ISO/FDIS*, 2011.

[2] P. Koopman and M. Wagner, "Challenges in autonomous vehicle testing and validation," *SAE International Journal of Transportation Safety*, vol. 4, no. 1, pp. 15–24, 2016.

[3] H. Shimodaira, "Improving predictive inference under covariate shift by weighting the log-likelihood function," *Journal of statistical planning and inference*, vol. 90, no. 2, pp. 227–244, 2000.

[4] P. Abbeel and A. Y. Ng, "Apprenticeship learning via inverse reinforcement learning," in *International Conference on Machine Learning (ICML)*, 2004.

[5] J. Morton, T. A. Wheeler, and M. J. Kochenderfer, "Analysis of recurrent neural networks for probabilistic modeling of driver behavior," *IEEE Transactions on Intelligent Transportation Systems*, vol. 18, no. 5, pp. 1289–1298, 2017.

[6] A. Kuefler, J. Morton, T. A. Wheeler, and M. J. Kochenderfer, "Imitating driver behavior with generative adversarial networks," in *IEEE Intelligent Vehicles Symposium (IV)*, 2017.

[7] J. Ho and S. Ermon, "Generative adversarial imitation learning," in *Advances in Neural Information Processing Systems (NIPS)*, 2016, pp. 4565–4573.

[8] J. K. Gupta, M. Egorov, and M. J. Kochenderfer, "Cooperative multi-agent control using deep reinforcement learning," in *Adaptive Learning Agents Workshop*, 2017.

[9] S. Schaal, "Is imitation learning the route to humanoid robots?" *Trends in cognitive sciences*, vol. 3, no. 6, pp. 233–242, 1999.

[10] S. Ross, G. Gordon, and D. Bagnell, "A reduction of imitation learning and structured prediction to no-regret online learning," in *International Conference on Artificial Intelligence and Statistics*, 2011, pp. 627–635.

[11] 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 (NIPS)*, 2017, pp. 5326–5335.

[12] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein generative adversarial networks," in *International Conference on Machine Learning (ICML)*, 2017, pp. 214–223.

[13] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, "Trust region policy optimization," in *International Conference on Machine Learning (ICML)*, 2015.

[14] J. Song, H. Ren, D. Sadigh, and S. Ermon, "Multi-agent generative adversarial imitation learning," 2018.

[15] M. L. Littman, "Markov games as a framework for multi-agent reinforcement learning," in *Machine Learning Proceedings 1994*. Elsevier, 1994, pp. 157–163.

[16] J. Colyar and J. Halkias, "US highway 101 dataset," Tech. Rep. FHWA-HRT-07-030, Jan. 2007.

[17] ——, "US highway 80 dataset," Tech. Rep. FHWA-HRT-06-137, Dec. 2006.

[18] Y. Duan, X. Chen, R. Houthooft, J. Schulman, and P. Abbeel, "Benchmarking deep reinforcement learning for continuous control," *arXiv preprint arXiv:1604.06778*, 2016.

[19] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville, "Improved training of wasserstein gans," in *Advances in Neural Information Processing Systems (NIPS)*, 2017, pp. 5769–5779.

[20] Y. Li, J. Song, and S. Ermon, "InfoGAIL: Interpretable imitation learning from visual demonstrations," *Advances in Neural Information Processing Systems (NIPS)*, pp. 3815–3825, 2017.

[21] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[22] J. Li, W. Monroe, T. Shi, A. Ritter, and D. Jurafsky, "Adversarial learning for neural dialogue generation," *arXiv preprint arXiv:1701.06547*, 2017.