MixGAIL: Autonomous Driving Using Demonstrations with Mixed Qualities

Gunmin Lee*, Dohyeong Kim*, Wooseok Oh, Kyungjae Lee, and Songhwai Oh

Abstract— **In this paper, we consider autonomous driving of a vehicle using imitation learning. Generative adversarial imitation learning (GAIL) is a widely used algorithm for imitation learning. This algorithm leverages positive demonstrations to imitate the behavior of an expert. In this paper, we propose a novel method, called mixed generative adversarial imitation learning (MixGAIL), which incorporates both of expert demonstrations and negative demonstrations, such as vehicle collisions. To this end, the proposed method utilizes an occupancy measure and a constraint function. The occupancy measure is used to follow expert demonstrations and provides a positive feedback. On the other hand, the constraint function is used for negative demonstrations to assert a negative feedback. Experimental results show that the proposed algorithm converges faster than the other baseline methods. Also, hardware experiments using a real-world RC car shows an outstanding performance and faster convergence compared with existing methods.**

I. Introduction

Designing a reactive controller for a high-speed mobile robot, such as a racing car, has been a long-standing challenging problem [1], [2]. Recent state-of-the-art approaches [3], [4] have employed a learning-based method, which trains a reactive controller from expert driving data while conventional approaches often manually designed a rule-based feedback controller [5], [6]. Especially, imitation learning (IL) [7]–[11] has drawn attention due to its benefits of training a controller by exploring an environment autonomously and following the given training data. Generally, the main goal of IL is to find an optimal controller to follow the given training data, expert data.

However, conventional IL is shortsighted in point of data. This algorithm only takes advantage of expert data, but doesn't take advantage of non-expert data. We will call a non-expert demonstration a negative demonstration. One of the main drawback of only using expert demonstrations is the redundancy of expert data. In data collecting processes, an expert often tries to follow an optimal policy and avoid hazardous state-action pairs. Hence, the collected expert data are often distributed near the optimal state-action pairs and hardly visit hazardous state-action pairs. The lack of training data near hazardous state-action pairs may lead poor generalization performance in IL. Furthermore, using negative demonstrations alleviates these drawbacks. Negative demonstrations include hazardous state-action pairs that expert demonstration does not have. Hence, when demonstrations with mixed qualities are presented, the agent is exposed to a wide range of experiences, accelerating the learning process.

In this paper, we propose a neural network-based imitation learning algorithm which can learn from both expert and negative demonstrations. In the proposed method, expert demonstrations give the agent a positive feedback (what to do), and negative demonstrations give a negative feedback (what not to do). By giving a negative feedback, the agent can avoid undesirable or dangerous situations recorded in negative demonstrations.

Generally, negative demonstrations and expert demonstrations have overlapping state-action pairs, causing instability in imitation learning. During the gradient-based training process, the agent is given a gradient not to imitate from the negative demonstrations and another gradient to imitate the expert demonstrations. The overlapping pairs make these two gradients to conflict with each other during training, causing the learning process to become unstable.

To resolve this problem, we use the idea from Heim et al. [12] to stably introduce the effect of negative demonstrations during training. [12] introduced *constraint critic*, which prevents a generative model from sampling from user-defined negative data. We extend this idea to a sequential data, which is called *constraint reward*. While *constraint critic* affects the gradient of the generator network during training, *constraint reward* does not affect the policy gradient during the training step. This implies that an overlapping state-action pair has no effect in gradient. An overlapping state-action pair is present in both expert and negative demonstrations, and its gradient makes the update process unstable. The proposed algorithm is called *mixed demonstration generative adversarial imitation learning* (MixGAIL).

We use two different environments to evaluate the proposed MixGAIL algorithm, the TORCS racing car simulator and real-world experiments using a RC car, comparing MixGAIL against the generative adversarial imitation learning (GAIL) [7]. In the TORCS environment, the proposed algorithm shows about 26% faster convergence than GAIL. In RC car experiments, MixGAIL can reach the performance level of the expert while GAIL fails to reach.

II. Related Work

A number of existing imitation learning or behavior cloning methods [13]–[19] have been developed under the assumption that data are collected from both expert and non-expert.

Shiarlis et al. [13] proposed an inverse reinforcement learning method using negative demonstrations through the feature expectations used in [14]. This method updates the feature weights to match the expectations to the expert's ones and to make distance from the negative's ones. They derive the policy as closed form using soft Bellman equation, which reflects the influence of the negative demonstrations naturally, but this method is limited to discrete action space and it is hard to express complex state space due to using the feature expectations. Choi et al. [15] proposed a method of how to evaluate how much a data point is reliable for Gaussian process regression (GPR). This method is called leveraged Gaussian process regression (LGPR), and shows

*Equal contribution

G. Lee, D. Kim, W. Oh, K. Lee, and S. Oh are with the Department of Electrical and Computer Engineering and ASRI, Seoul National University, Seoul 08826, Korea (e-mail: gunmin.lee@rllab.snu.ac.kr, dohyeong.kim@rllab.snu.ac.kr, wooseok.oh@rllab.snu.ac.kr, kyungjae.lee@rllab.snu.ac.kr, songhwai@snu.ac.kr).

This work was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Science and ICT (NRF-2017R1A2B2006136).

better performence than a normal GPR, even though the dataset is less reliable. Choi et al. [16] also extended this method to a deep neural network and showed that this leverage method works with a neural network. Lee et al. [17] have applied an LGPR to a imitation learning problem by modeling a reward estimation as an LGP while the methods proposed in [15], [16] are behavior cloning.

The usage of negative demonstrations were inspired by [18], where Choi et al. introduced a new paradigm that update a model using some demonstrations that included not only skillful driver's but also aggressive driver's demonstrations. As similarly mentioned in [12], we want our agent to imitate expert data and not to imitate negative data. However, with similar state-action pair existing in both data, their effects on the agent will not be harmonic, rather conflicting. This restrains the usage of both data in imitation learning.

III. Background

A. Generative Adversarial Imitation Learning

GAIL gets expert demonstrations of an agent and reproduces a policy that can work as well as given demonstrations in the same environment. The objective function of this algorithm can be written as:

$$\min_{\pi\in\Pi}(\max_{D}(E_{\pi}[\log(D(s,a))]+E_{\pi_E}[\log(1-D(s,a))])-H(\pi)), \tag{1}$$

where E_{π} and E_{π_E} are the expected returns based on the policy π and π_E, respectively, S and A are state and action space, state $s \in S$, action $a \in A$, $D : S \times A \mapsto (0, 1)$ is discriminator, and $H(\pi)$ is the causal entropy of π. When the policy, π, generates a demonstration, the discriminator, D, discriminates whether this input is the original expert demonstration or a generated demonstration. After training both the policy and the discriminator competitively, the policy generates a demonstration that is similar to demonstrations of an expert.

In deep learning, a gradient step is needed to find the optimal policy. GAIL [7] calculates the gradient step based on the following equation.

$$\hat{E}_{\tau_i}[\nabla_{\theta}\log\pi_{\theta}(a|s)Q(s,a)]-\lambda\nabla_{\theta}H(\pi_{\theta}), \tag{2}$$

where $\hat{E}_{\tau_i}$ is the expected return of the ith sampled demonstration, Q is the average estimated reward function of the sampled demonstration, and π_{θ} is an approximation of π with weights θ.

B. Constrained Generative Adversarial Networks

CONGAN is a generative model using Wasserstein GAN [20] which is one of the GAN framework. CONGAN not only generates high-quality images, but also restricts output to a user-defined semantic space through user interaction. In the training process, the user is presented with several generator outputs. The user defines the output closest to the user-defined semantic space as a positive output and the most distant output as a negative output. These two outputs are added to the constraint set as a pair, and used to compute the *constraint critic*. The *constraint critic* is a term in CONGAN's objective function, which helps to train the generator to satisfy all the constraints in the set. The *constraint critic* can be calculated as below:

$$l_{\phi,S}(\hat{X},C)=-\frac{1}{|C|}\sum_{(X_+,X_-)\in C}p_S(\phi(\hat{X}),\phi(X_+),\phi(X_-)), \tag{3}$$

where $p_S(a, b, c)$ is the t-Distributed Stochastic Triplet Embedding.

$l_{\phi,S}$ is subtracted from the objective's discriminator term and optimized as the following:

$$\min_{\theta}\max_{W}E_{X\sim P_D}[d_W(X)]-E_{\hat{X}\sim P_g}\left[d_W(\hat{X})-\gamma l_{\phi,S}(\hat{X},C)\right], \tag{4}$$

where d_W is the approximate Wasserstein discriminator [20] with weights W using Wasserstein-1 distance, P_D is a data distribution, and P_g is the generator distribution.

IV. Mixed Demonstration Generative Adversarial Imitation Learning

The *constraint critic* in CONGAN [12] evaluates a relative distance of its output between the user-selected images and the user-rejected images. Using this property of the *constraint critic* in IRL, a policy can be trained to imitate an expert agent rather than an agent that acts in an unwanted way, called a negative agent. We define an objective function with a term, called a *constraint reward* function (CoR), as below.

$$\begin{aligned}\min_{\pi\in\Pi}\max_{D}E_{\pi}[\log(D(s,a))-\eta\mathrm{CoR}(s,s_E,s_N)]+\\ E_{\pi_E}[\log(1-D(s,a))]-\lambda H(\pi),\end{aligned} \tag{5}$$

where s_E is a set of states of expert demonstrations, s_N is a set of states in negative demonstrations, and η and λ control the proportion of CoR and H, respectively. In this equation, the CoR term is added to GAIL's objective function. CoR estimates a relative distance of a state between the expert states set and the negative states set, so we define the term similar to CONGAN's *constraint critic* as below.

$$\begin{aligned}\mathrm{CoR}(s,s_E,s_N)&=\frac{(1+\frac{d_{s_E}}{\alpha})^{-\frac{\alpha+1}{2}}}{(1+\frac{d_{s_E}}{\alpha})^{-\frac{\alpha+1}{2}}+(1+\frac{d_{s_N}}{\alpha})^{-\frac{\alpha+1}{2}}},\\ d_{s_E}&=\sqrt{\frac{1}{|s_E|}\sum_{s_e\in s_E}(\|s-s_e\|_2)^2},\\ d_{s_N}&=\sqrt{\frac{1}{|s_N|}\sum_{s_n\in s_N}(\|s-s_n\|_2)^2},\end{aligned} \tag{6}$$

where $\|\cdot\|_2$ is l_2 norm, and α is a hyperparameter used to control the sharpness of CoR. To show the algorithm can be computed like GAIL [7], it is necessary to show the existence and uniqueness of the saddle point of the proposed objective function. This property can be proved using the occupancy measure of policy.

Theorem 1: Equation (5) has the unique saddle point.

Proof: There is one-to-one correspondence between policy and occupancy measure because of [7], so let π's occupancy measure is ρ_{π} and π_E's occupancy measure is ρ_{π_E}. Then, replace the discriminator with a cost function regularizer ψ in the objective function.

$$\min_{\pi\in\Pi}\psi^*(\rho_{\pi}-\rho_{\pi_E})-\eta E_{\pi}[\mathrm{CoR}(s,s_E,s_N)]-\lambda H(\pi) \tag{7}$$

Using a property of the cost function regularizer's conjugate function, replace π with ρ_{π} in the expression.

$$\begin{aligned}\min_{\pi\in\Pi}\max_{D}\sum_{s,a}\rho_{\pi}(s,a)(c(s,a)-\eta\mathrm{CoR}(s,s_E,s_N))\\ -\sum_{s,a}\rho_{\pi_E}(s,a)c(s,a)-\lambda\bar{H}(\rho_{\pi})-\psi(c),\end{aligned} \tag{8}$$

Algorithm 1 MixGAIL

Input: Expert demonstrations $\tau_E \sim \pi_E$, negative demonstrations $\tau_N \sim \pi_N$, expert states set $s_E = \{s_e | \forall (s_e, a_e) \in \tau_E\}$, negative states set $s_N = \{s_n | \forall (s_n, a_n) \in \tau_N\}$, initial policy and discriminator parameters θ_0, ω_0, weighting parameter of CoR $\eta \geq 0$, η's decay rate ϵ

1: **for** $i = 0, 1, 2, \ldots$ **do**
2: Sample demonstrations $\tau_i \sim \pi_{\theta_i}$
3: Update the discriminator parameters from ω_i to $\omega_{i+1}$with the gradient

$$\hat{E}_{\tau_i}[\nabla_\omega \log(D_\omega(s,a))] + \hat{E}_{\tau_E}[\nabla_\omega \log(1 - D_\omega(s,a))]$$

4: Take a policy step from θ_i to θ_{i+1}, using the TRPO rule with cost function $\log(D_{\omega_{i+1}}(s,a)) - \eta \mathrm{CoR}(s, s_E, s_N)$. Specifically, take a KL-constrained natural gradient step with

$$\hat{E}_{\tau_i}[\nabla_\theta \log \pi_\theta(a|s) A_{\pi_\theta}(s,a)] - \lambda \nabla_\theta H(\pi_\theta)$$

where

$$\begin{aligned} A_{\pi_\theta}(s,a) = &- \log D_{\omega_{i+1}}(s,a) + \gamma V_{D\pi_\theta}(s') \\ &- V_{D\pi_\theta}(s) + \eta(\mathrm{CoR}(s, s_E, s_N) \\ &+ V_{C\pi_\theta}(s') - V_{C\pi_\theta}(s)), \end{aligned}$$

and s' is the next state after state s.
5: update $\eta \leftarrow \epsilon\eta$, when $\Delta V_D > \Delta V_C$
6: **end for**

where c is a cost function, which is negative of reward function, and the causal entropy of the occupancy measure $\bar{H}$ is defined in [7] as following:

$$\bar{H}(\rho) = -\sum_{s,a} \rho(s,a) \log(\rho(s,a) / \sum_{a'} \rho(s,a')). \quad (9)$$

Let $L(\rho_\pi, c) = \sum_{s,a} \rho_\pi(s,a)(c(s,a) - \eta \mathrm{CoR}(s, s_E, s_N)) - \sum_{s,a} \rho_{\pi_E}(s,a) c(s,a) - \lambda \bar{H}(\rho_\pi) - \psi(c)$. $c(s,a)$ and $\mathrm{CoR}(s, s_E, s_N)$ are independent of π, so the term $\sum_{s,a} \rho_\pi(s,a)(c(s,a) - \eta \mathrm{CoR}(s, s_E, s_N))$ is linear combination of ρ_π, and $\bar{H}$ is strictly concave. Therefore, $L(\rho_\pi, c)$ is strictly convex for ρ_π. Also, $L(\rho_\pi, c)$ is concave for c because of definition of the cost function regularizer in [7]. From the convexity of $L(\rho_\pi, c)$, the proposed objective function has the unique saddle point. ∎

Theorem 1 implies that we can alternatively update the discriminator and the policy. Thus, we can optimize the proposed objective function through the GAIL framework. If an agent's state is closer to the expert states set than the negative one, CoR's output is bigger than 0.5. On the contrary, if agent's state is closer to the negative states, CoR's output is smaller than 0.5. Thus, the proposed objective function pull the training agent toward the expert agent and push away from the negative agent. Unlike GAIL, the policy is trained through rewards not only made by the discriminator, but also made by CoR, so the CoR term is used as a bonus function. Then, the reward function of a policy is defined as below.

$$r(s,a) = -\log(D(s,a)) + \eta \mathrm{CoR}(s, s_E, s_N). \quad (10)$$

In perspective of a reward function, CoR helps to make a more sophisticated reward function than only using discriminator's reward at the initial training phase because the discriminator is not trained perfectly at that time. For example, if a driver meets a corner for the first time and do not slow down, the discriminator cannot give a low reward because the driver did not explore the corner, but CoR can give a low reward due to the existence of demonstrations crashing in corners at a high speed in negative demonstrations. Therefore, the proposed objective function updates the policy quickly at the initial time.

The weighting parameter η of CoR needs to be scaled over training time because if the discriminator is getting more accurate, the large proportion of CoR term in the objective function obstructs to give a policy well-restored rewards. Therefore, η is decayed as training is progressed. If η is modified, the value function's target value is also fluctuated. For the stable update, we aligned the policy's value network into a discriminator's value network (V_D) and a *constraint reward*'s value network (V_C) as below.

$$\begin{aligned} V_\pi(s) &= V_{D\pi}(s) + \eta V_{C\pi}(s), \\ V_{D\pi}(s) &= E_\pi \left[\sum_{k=t}^{T} -\gamma^{k-1} \log D(s_k, a_k) \middle| s_t = s \right], \\ V_{C\pi}(s) &= E_\pi \left[\sum_{k=t}^{T} \gamma^{k-1} \mathrm{CoR}(s_k, s_E, s_N) \middle| s_t = s \right], \end{aligned} \quad (11)$$

where γ is a discount factor.

Here, we propose two ways to decay η. The first one is to decay η exponentially per training step, which is used for the RC car experiment. The other is to conditionally decay η when the discriminator's reward is more supportive to the training policy than *constraint reward*, which is used for the TORCS experiment. Each reward's contribution to training can be measured by the change of the reward's value. The policy's objective can be expressed as $\max_\pi V_\pi(s_0) = \max_\pi V_{D\pi}(s_0) + \eta V_{C\pi}(s_0)$, where s_0 is an initial state. Let π becomes π' after a training step. Then, the changes in reward values can be computed as follows:

$$\Delta V_C = V_{C\pi'}(s_0) - V_{C\pi}(s_0), \quad (12)$$

$$\Delta V_D = V_{D\pi}'(s_0) - V_{D\pi}(s_0). \quad (13)$$

The second way to scale η can be interpreted as decaying it when ΔV_D is bigger than ΔV_C. If the change of the initial state's value is large after taking a training step, it can be inferred that the policy is becoming similar to the optimal policy. On the other hand, if the change of the initial state's value is insignificant, it can be inferred that the reward function cannot distinguish the policy after training and the policy before training. Therefore, if ΔV_D is larger than ΔV_C, it can be referred that the *constraint reward* has less influence on training than the discriminator's reward.

For training, we approximate discriminator D and policy π. D and π are replaced by D_ω and π_θ, respectively, where ω and θ are weights. The training process is explained in Algorithm 1.

V. Experimental Setup

A. Simulation Experiment Setup

We test our algorithm on a racing car simulator, TORCS [21] (see Figure 1). To validate the effectiveness of negative demonstrations, the proposed method is compared with GAIL [7] which employs only expert demonstrations.

This environment has a three dimensional action space representing steering, throttle and brake, and has a 29 dimensional observation space, which consists of the velocity vector, 19 range finder sensors, a deviation from the center

Fig. 1: Scenes of negative demonstrations of TORCS experiments. The car crashes to the lane side or skids along the lane.

Fig. 2: The RC car used for hardware experiment.

of the lane, the angle between the car and the track direction, four wheels' rotational speed, and rotation per minute (RPM) of the engine. The objective is to cross the finish line as fast as possible without colliding with walls. To measure the performance of the trained agent and compare it to expert performance, we use the underlying reward function of TORCS. Note that the underlying reward is only used for evaluation and not used during training.

1) Collecting Expert and Negative Demonstrations: To collect expert demonstrations, we train the expert policy by using PPO [22]. The negative demonstrations were collected from an under-trained PPO policy. We provide 48,798 expert state-action pairs for GAIL. On the other hand, we provide the proposed algorithm 24,588 expert state-action pairs, which is the half of state-action pairs provided to GAIL, and 24,339 negative state-action pairs.

2) Comparison with Gradient Affecting Version: Under the GAIL framework, the expert demonstrations affect the discriminator through a gradient step, so it is a natural idea to make the negative demonstrations affect the gradient of the objective function. Nevertheless, MixGAIL uses negative demonstrations to influence the reward function of the policy. To show why MixGAIL does not affect the gradient, we introduce a method, called gradient affecting MixGAIL, and compare this method with the proposed MixGAIL.

The conjugate of the cost function regularizer is used to estimate the distance between the expert and the trained policy in GAIL. Gradient affecting MixGAIL uses the regularizer of the cost function to separate the training agent and the negative agent. Its objective function is expressed as below.

$$\min_{\pi \in \Pi} \psi^*(\rho_\pi - \rho_{\pi_E}) - \eta\psi^*(\rho_\pi - \rho_{\pi_N}) - \lambda H(\pi), \qquad (14)$$

where π_N is the negative agent's policy.

Then we replace the cost function regularizer with the discriminator.

$$\begin{aligned} &\min_{\pi \in \Pi} \max_{D} E_\pi[\log D(s,a)] + E_{\pi_E}[\log(1 - D(s,a))] \\ &- \eta(E_\pi[\log D(s,a)] + E_{\pi_N}[\log(1 - D(s,a))]) - \lambda H(\pi) \end{aligned} \qquad (15)$$

In the equation (15), the discriminator related terms are added to GAIL's objective function. It means that the negative demonstrations affect the gradient of discriminator directly during the training phase. The discriminator's gradient step and the policy's gradient step is taken alternately during the training phase. η is exponentially decayed after the policy's gradient step. The policy's gradient is the same as Algorthim 1, and the discriminator's gradient is shown as below.

$$\begin{aligned} &\hat{E}_\tau[\nabla_\omega \log D_\omega(s,a)] + \hat{E}_{\tau_E}[\nabla_\omega \log(1 - D_\omega(s,a))] - \\ &\eta(\hat{E}_\tau[\nabla_\omega \log D_\omega(s,a)] + \hat{E}_{\tau_N}[\nabla_\omega \log(1 - D_\omega(s,a))]), \end{aligned} \qquad (16)$$

where τ is the demonstrations sampled from the policy π, τ_E and τ_N are the expert and negative demonstrations, respectively, and the discriminator is approximated by ω.

B. RC Car Experiment Setup

Our environment's input state is the LiDAR sensor (20 state values) and the action is one dimension (steering of a RC car). If the RC car collides with walls before it reaches end of the circuit, or reach the end of the circuit without collision, the trial is ended. We pretrained both GAIL and MixGAIL with behavior cloning to initialize and save time for training. The RC car used for the training is shown in figure 2

We compare our algorithm with GAIL [7] to show the faster convergence of the proposed algorithm. GAIL and MixGAIL received the same number of demonstrations. However, the half of demonstrations given to MixGAIL are expert demonstrations, and the other half are negative demonstrations.

The performance measure of this experiment is based on how much the RC car has moved along the given path. We numbered each temporary wall, and the number of nearest wall to the stopped position is our performance measure.

1) Collecting Expert and Negative demonstrations: In this experiment, we manually collected our expert demonstrations by controlling the RC car ourself. These demonstrations are composed of 4,510 state-action pairs. The negative demonstrations are sampled from a policy from under-trained GAIL. These demonstrations consist of demonstrations colliding with walls before reaching the goal.

The number of state-action pairs that consists negative demonstration is 2,255, which is the same as the number of state-action pair of expert demonstration input.

VI. Results and Analysis

A. TORCS Results

We tested GAIL and MixGAIL nine times using different random seeds and plotted a graph of rewards per training step with averages and standard deviations (see Figure 3).

1) Analysis: The success and fail is determined by 95% of the expert data's performance. MixGAIL reaches 95% of expert data's performance 27.6% faster than original GAIL. Also, it shows 0.1% to 0.2% of better performance than the original GAIL. The result is shown in Figure 3, and the numerical result is shown in Table I.

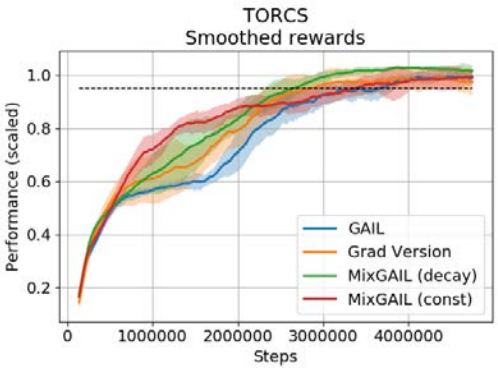

Fig. 3: Scaled performance of GAIL and MixGAIL on TORCS. Bold lines are average value over nine trials. Dotted line is 95% of expert data's performance.

	GAIL	MG (decay)	MG (const)	Gradient
H.T.(steps)	3,659,974	2,652,659	3,377,976	2,783,082
Max reward	0.995	1.029	0.994	0.991
Final reward	0.989	1.015	0.993	0.975

TABLE I: The numerical analysis of TORCS simulation of proposed method and comparisons. MG (decay) indicates the η decayed version of MixGAIL, MG (const) indicates the η constant version of MixGAIL, and Gradient indicates the Gradient affecting verion of MixGAIL. H.T. indicates hitting time.

As shown in Figure 3, the proposed MixGAIL has faster update speed per step, and can learn complex problems faster than GAIL [7]. Even though the size of the training sets is the same, the proposed algorithm takes advantage of negative demonstrations, which guide our algorithm to overcome the bottleneck of training by inhibiting the agent from visiting hazardous state-action regions.

2) Effect of Hyperparameters in Training: The proposed method consistently obtain promising performances with various α. This result shows that α does not affect to the performance.

For η, we experimented about how will η decaying affect the learning process. As shown in Figure 3, leaving η constant have faster learning speed initially. However, as training proceeds, the learning curve is not smooth and gets caught up by η decaying version. We speculate that using the additional CoR rewards makes the agent confused when the agent's performance becomes similar to the expert's one. Consequently, we empirically show that scheduling η can achieve both better performance and faster convergence than fixing η as a constant.

3) Comparison to Gradient Affecting MixGAIL: Figure 3 shows that there is a bottleneck also in the gradient affecting MixGAIL's learning curve.

MixGAIL avoids hazardous state-actions pairs as shown in the learning curve (Figure 3) since the constraint reward function restrains the agent from visiting hazardous state-action pairs in advance. However, gradient affecting MixGAIL only uses the discriminator when judging where the hazardous state-action pairs are, causing exploration of the hazardous zone during training the discriminator. The exploration causes the bottleneck in the learning curve.

Additionally, another problem is that training of gradient affecting MixGAIL does not converge well when the initial value of η is too big. The gradient (15) can be interpreted as the sum of two terms, a gradient by negative demonstrations and a gradient by expert demonstrations. If, however, expert and negative demonstrations have common state-action pairs, the two types of gradients conflict with each other, making the training process unstable. Since η adjusts the reflection rate of each gradient in training, the higher the value of η, the more likely it is that gradient affecting MixGAIL diverges. On the other hand, because the convergence of MixGAIL is not affected by η, η's value can be adjusted without restrictions in MixGAIL.

	GAIL	8:2	7:3	5:5
H.T. (steps)	3,401,211	2,759,172	2,752,169	2,746,975
Max reward	1.009	1.007	1.010	1.031
Final reward	1.000	0.993	1.005	1.023

TABLE II: The numerical analysis of TORCS simulation on different data ratios. Each column indicates GAIL, MixGAIL with expert and negative data ratio of 8:2, 7:3, 5:5. H.T. indicates hitting time.

	GAIL	MixGAIL
H.T. (steps)	$>$ 72,816	72,816
Max reward	0.807	0.970
Final reward	0.804	0.969

TABLE III: The numeral analysis of hardware RC car experiment of GAIL and MixGAIL. H.T. indicates hitting time.

4) Different data ratio: Rather than using only same number of expert and negative data, we changed the ratio of expert and negative data. While the total number of expert and negative state-action pairs are the same, we changed the data ratio into 7:3 and 8:2. We tested each example 5 times using different random seeds. The results are shown in Table II. The results shows that as the data ratio becomes closer to 5:5, the results gets more promising.

B. RC Car Experiment Results

1) Analysis: The success and fail is determined by 95% of the expert data's performance. MixGAIL reached 95% of expert driver performance, while GAIL could not reach it. Also, it showed 20% better performance than the original GAIL. The result is organized in Table III. As shown in Figure 6, not only is MixGAIL faster in learning speed, but also has better final performance in the RC car experiment. The proposed method shows more than 20% of better result than the original GAIL [7], even though the number of given data is the same. Therefore, we can conclude MixGAIL processes data more efficiently than GAIL.

The results of the proposed method show less variance than GAIL. We speculate that this is due to exploration of the agent. As the training proceeds, the original GAIL explores regions that are not visited by given demonstrations, mostly unsafe regions. By visiting unsafe regions, GAIL results in a large variance. However, in the proposed MixGAIL, the *constraint reward* term gives information about unsafe regions and restricts the agent from visiting those regions. This allows our agent to safely explore the environment, making the variance small.

Even though MixGAIL performs better than GAIL, there are limitations in MixGAIL. As the size of the training set increases, the computation and memory required to compute *constraint reward* increases linearly. For further work, we plan to develop an efficient method to compute the constraint reward.

Fig. 4: Snapshots of a demonstration of GAIL after finishing the training.

Fig. 5: Snapshots of a demonstration of MixGAIL after finishing the training.

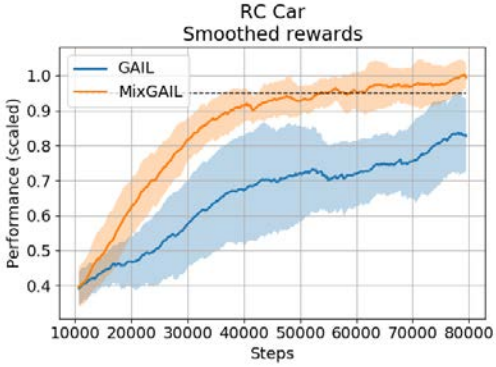

Fig. 6: Scaled performance of GAIL and MixGAIL on RC Car hardware platform. Bold lines are average value over 4 trials. Dotted line is 95% of expert's performance.

VII. Conclusion

In this paper, we have proposed a method of using both expert demonstrations and negative demonstrations in imitation learning. To mitigate the overlapping state-action demonstrations of expert and negative demonstrations, we have added a *constraint reward* term to original GAIL and called it MixGAIL. MixGAIL showed faster and better solution than the original GAIL. For the TORCS environment, MixGAIL is 27.6% faster to learn to run around the track without collision. Also, for the real-world RC car, our algorithm showed faster learning speed and better result than the original GAIL. The proposed algorithm could be applied to multiple other environments, which is left as a future work.

References

[1] A. Haji, P. Shah, and S. Bijoor, "Self driving RC car using behavioral cloning," *CoRR*, vol. abs/1910.06734, 2019.

[2] M. Cho, "A study on the obstacle recognition for autonomous driving RC car using lidar and thermal infrared camera," in *Eleventh International Conference on Ubiquitous and Future Networks*, 2019, pp. 544–546.

[3] D. Li, D. Zhao, Q. Zhang, and Y. Chen, "Reinforcement learning and deep learning based lateral control for autonomous driving [application notes]," *IEEE Comp. Int. Mag.*, vol. 14, no. 2, pp. 83–98, 2019.

[4] P. Klose and R. Mester, "Simulated autonomous driving in a realistic driving environment using deep reinforcement learning and a deterministic finite state machine," in *Proc. of the 2nd International Conference on Applications of Intelligent Systems*, 2019, pp. 30:1–30:6.

[5] N. Naus and J. Jeuring, "Building a generic feedback system for rule-based problems," in *Trends in Functional Programming - 17th International Conference*, 2016, pp. 172–191.

[6] W. Zhao, M. A. Reinthal, D. D. Espy, and X. Luo, "Rule-based human motion tracking for rehabilitation exercises: Realtime assessment, feedback, and guidance," *IEEE Access*, vol. 5, pp. 21 382–21 394, 2017.

[7] J. Ho and S. Ermon, "Generative adversarial imitation learning," in *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016*, pp. 4565–4573.

[8] Z. Wang, J. Merel, S. E. Reed, N. de Freitas, G. Wayne, and N. Heess, "Robust imitation of diverse behaviors," in *Proc. of the Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems*, 2017, pp. 5320–5329.

[9] R. P. Bhattacharyya, D. J. Phillips, B. Wulfe, J. Morton, A. Kuefler, and M. J. Kochenderfer, "Multi-agent imitation learning for driving simulation," in *Proc. of the International Conference on Intelligent Robots and Systems*, 2018, pp. 1534–1539.

[10] H. Xiao, M. Herman, J. Wagner, S. Ziesche, J. Etesami, and T. H. Linh, "Wasserstein adversarial imitation learning," *CoRR*, vol. abs/1906.08113, 2019.

[11] K. Lee, S. Choi, and S. Oh, "Maximum causal tsallis entropy imitation learning," in *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, 3-8 December 2018, Montréal, Canada*, 2018, pp. 4408–4418.

[12] E. Heim, "Constrained generative adversarial networks for interactive image generation," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 10 753–10 761.

[13] K. Shiarlis, J. V. Messias, and S. Whiteson, "Inverse reinforcement learning from failure," in *Proc. of the International Conference on Autonomous Agents & Multiagent Systems*, 2016, pp. 1060–1068.

[14] P. Abbeel and A. Y. Ng, "Apprenticeship learning via inverse reinforcement learning," in *Proc. of the Twenty-First International Conference on Machine Learning*, 2004, pp. 1–8.

[15] S. Choi, K. Lee, and S. Oh, "Robust learning from demonstration using leveraged Gaussian processes and sparse-constrained optimization," in *IEEE International Conference on Robotics and Automation*, 2016, pp. 470–475.

[16] ——, "Scalable robust learning from demonstration with leveraged deep neural networks," in *Proc. of the International Conference on Intelligent Robots and Systems*, 2017, pp. 3926–3931.

[17] K. Lee, S. Choi, and S. Oh, "Inverse reinforcement learning with leveraged Gaussian processes," in *Proc. of the International Conference on Intelligent Robots and Systems*, 2016, pp. 3907–3912.

[18] S. Choi, K. Lee, and S. Oh, "Robust learning from demonstrations with mixed qualities using leveraged Gaussian processes," *IEEE Trans. Robotics*, vol. 35, no. 3, pp. 564–576, 2019.

[19] K. Lee, Y. Choi, and S. Oh, "Inverse optimal control from demonstrations with mixed qualities," in *2020 17th International Conference on Ubiquitous Robots (UR)*, 2020, pp. 581–586.

[20] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein generative adversarial networks," in *Proc. of the 34th International Conference on Machine Learning*, 2017, pp. 214–223.

[21] D. Loiacono, L. Cardamone, and P. L. Lanzi, "Simulated car racing championship: Competition software manual," *CoRR*, vol. abs/1304.1672, 2013.

[22] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *CoRR*, vol. abs/1707.06347, 2017.