Randomized Adversarial Imitation Learning for Autonomous Driving

MyungJae Shin , Joongheon Kim
Chung-Ang University, Seoul, Republic of Korea
{mjshin.cau, joongheon}@gmail.com

Abstract

With the evolution of various advanced driver assistance system (ADAS) platforms, the design of autonomous driving system is becoming more complex and safety-critical. The autonomous driving system simultaneously activates multiple ADAS functions; and thus it is essential to coordinate various ADAS functions. This paper proposes a randomized adversarial imitation learning (RAIL) method that imitates the coordination of autonomous vehicle equipped with advanced sensors. The RAIL policies are trained through derivative-free optimization for the decision maker that coordinates the proper ADAS functions, e.g., smart cruise control and lane keeping system. Especially, the proposed method is also able to deal with the LIDAR data and makes decisions in complex multi-lane highways and multi-agent environments.

1 Introduction

With the increasingly growing interests in autonomous driving, the various forms of advanced driver assistance system (ADAS) functions such as smart cruise control (SCC), lane keeping system (LKS) and collision-avoidance systems (CAS) have been developed with high potentials in the enhanced convenience of drivers for limited on-driving situations. Especially, in multi-lane highway environments, it is essential to form efficient long term assistance strategies while maintaining safety because the malfunctions in safety cause on-road accidents and road congestion. The various ADAS functions presented in modern autonomous driving have high interdependence; thus it has to be regarded as a single integrated system. Therefore the strategies that properly coordinate the ADAS functions are required.

A conventional system hierarchy of autonomous vehicle is as illustrated in Fig. 1. The low-level ADAS controllers are directly connected to the LIDAR sensors accessible in autonomous vehicle. The controllers determine the information needed to control the autonomous vehicle and transmit the determined operations to mechanical components. As a single integrated system, it is expected that multiple ADAS functions simultaneously cooperate to manage the systems

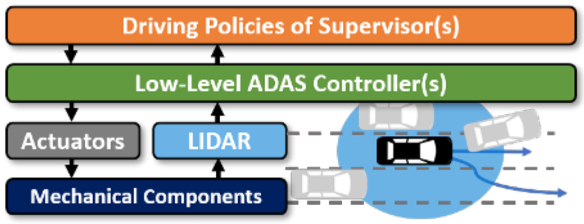

Driving Policies of Supervisor(s)
Low-Level ADAS Controller(s)
Actuators
LIDAR
Mechanical Components

Figure 1: Simplified learning hierarchy to control vehicle systems.

operation of the vehicle. Therefore, a supervisor that coordinates the low-level controllers needs to select appropriate ADAS functions when the vehicle acts in dynamic on-road environments [Korssen *et al.*, 2018]. The objective of the supervisor is to be a decision maker of the overall system during driving operation. The problem is that the driving policies of the supervisor should satisfy the robustness regardless of various traffic environments. Prior research results on autonomous driving consist of diverse approaches with rule-based driving policies. However, these policies have been difficult to cope with time-varying environments (i.e., huge observation spaces and action spaces) [Gipps, 1986; Ahmed, 1999]. Recently, the emergence of deep reinforcement learning (DRL), which utilizes powerful function approximations such as neural networks, allows the supervisor to obtain robust driving policies; it has made revolutionary progresses in the autonomous driving [Mnih *et al.*, 2015; Silver *et al.*, 2016; Sallab *et al.*, 2016; Hoel *et al.*, 2018; Mukadam *et al.*, 2017]. However, there are challenges with DRL when the driving policies try to learn the policies that maximize the expected rewards during operation. The criteria for what should be the reward function of autonomous driving is still in progress in many studies. Furthermore, since there are undesirable policies to maximize the expected rewards at the expense of violating the implicit rules of the environments, it is difficult to learn the robust and safe policies through DRL in autonomous driving [Pan *et al.*, 2018]. These problems motivate the researchers to adopt imitation learning (IL) to optimize the driving policy instead. The IL trains the driving policies based on the desired behavior demonstrations rather than the configuration of the specific reward functions as well as the IL can leverage domain knowledge. Based on the advantages of IL, it has been proved that the IL performs remarkably in the areas of robotics, navigation, autonomous vehicle, and etc [Pomerleau, 1991; Pomerleau, 1989; Pan *et al.*, 2018]. However, the main challenge faced by many researchers is the techniques that combine DRL and IL algorithms require too much data to achieve reasonable performance, and the corresponding famous example is generative adversarial imitation learning (GAIL) [Schulman *et al.*, 2015; Schulman *et al.*, 2017]. To address this issue, the algorithm models become complicated; and thus the models lead to reproducibility crisis. Furthermore, the models are sensitive to the implementation structure of the same algorithms and rewards from environments. For example, in GAIL, the discriminator of Generative Adversarial Networks (GAN) takes a role of the reward function. With the combination of discrimination and the complex DRL algorithms, e.g., TRPO and PPO, the GAIL trains the policies. As a result, the reconstruction results do not have always reasonable performance, and can stuck in suboptimal even with marginal differences. These problems make the difficulties in training robust autonomous driving policies; the trained policies have not yet been successfully deployed to autonomous vehicles [Henderson *et al.*, 2017b; Islam *et al.*, 2017]. Recently, augmented random search (ARS) that consists of the natural gradient policy algorithm is proposed [Rajeswaran *et al.*, 2017]. Because the ARS is a derivative-free simple linear policy optimization method, it is relatively easy to reconfigure the robust trained policy that shows reasonable performance. In this work, we present an IL-based method that combines the concepts of ARS and GAIL. For more details, random search based *randomized adversarial imitation learning (RAIL)* algorithm is proposed in this paper; and the RAIL algorithm trains policies using randomly generated matrices where the random matrices are used to search update directions that lead to optimal policies. This approach is advantageous in terms of computation (such as back-propagation) overhead reduction whereas DRL algorithms that use gradients to optimize weights. Furthermore, by leveraging expert demonstrations, our system can learn the driving policies of supervisor that achieves similar performance compared to the expert in terms of average speeds and lane changes. Through the data-intensive performance evaluation, it is demonstrated that the proposed RAIL algorithm can train the autonomous driving decision maker as desired.

Our proposed RAIL method shows that the random search in the space of policy parameters can be adapted to IL for autonomous driving policies. For more details, our contributions are as follows: (i) self-driving mechanism is proposed inspired by IL. Our method can successfully imitate expert demonstrations; and the corresponding static and linear policies can achieve similar speeds with many lane changes and overtakes. (ii) previous IL methods are based on conventional RL methods which show complicate configurations to control autonomous driving. However, RAIL has simplicity based on derivative-free random search. (iii) this method has not been previously applied to learn the robust driving policies in autonomous driving.

The rests of this paper are as follows: Sec. 2 and Sec. 3 describe the previous work and background knowledge. Sec. 4 defines our problem, i.e., training policies for autonomous driving. Sec. 5 designs the RAIL algorithm. Sec. 6 shows the experiment results by expert demonstrations in highway autonomous vehicle control. Sec. 7 concludes this paper.

2 Related Work

Imitation Learning (IL)

The IL methods are divided into two categories, i.e., behavioral cloning (BC) and inverse reinforcement learning (IRL). The BC is considered as the simplest IL method. To restore expert policy, it works by collecting training data from the expert drivers behaviors, and then uses it to directly learn the corresponding policy. If the policy deviates from trajectories that is trained in the training procedure, the agent tends to be fragile. This is because behavior cloning tries to reduce the 1-step deviation error of training data, not to reduce the error of entire trajectories. Prerequisites for reasonable policy restoration is a sufficient number of expert driving demonstrations. On the other hand, IRL has an intermediate procedures to estimate and recover the hidden reward function which explains the expert demonstration [Ziebart *et al.*, 2008; Finn *et al.*, 2016b]. Since IRL has to optimize the policy as well as the reward function, IRL generally implies significant computational costs. In [Finn *et al.*, 2016a] an [Ho and Ermon, 2016], the theoretical and practical considerations of connections between IRL and adversarial network is studied. GAIL framework learns a policy that can imitate expert demonstration using the discriminator network, which bypasses the reward function optimization.

Simplest Model-Free RL

The simplest model-free RL method that can solve standard benchmarks of RL has been studied under the two different directions: linear policies via natural policy gradients [Rajeswaran *et al.*, 2017] and a derivative-free policy optimization [Salimans *et al.*, 2017]. [Rajeswaran *et al.*, 2017] shows that complicated structures of policies are not needed to solve continuous control problems. The authors train linear policies via natural policy gradients. The trained policies obtain competitive performance on the complex continuous environments. In[Salimans *et al.*, 2017], the authors showed that evolution strategies (ES) offers less data efficiency than traditional RL, but offers many advantages. Especially, a derivative-free optimization allows ES to be more efficiently in distributed learning. Furthermore, the trained policies tend to be more diverse compared to policies trained by traditional RL methods. In [Mania *et al.*, 2018], the connection between [Salimans *et al.*, 2017] and [Rajeswaran *et al.*, 2017] is studied to obtain the simplest model-free RL method yet, a derivative-free optimization for training linear policies. The proposed simple random search method showed state-of-art sample efficiency compared to competing methods in MuJoCo locomotion benchmarks.

3 Background

Markov Decision Process (MDP)

MDP is formalized by $(S, A, p(s), p(s'|s,a), r(s,a,s'), \gamma)$ where S, A, $p(s)$, $p(s'|s,a)$, $r(s,a,s')$, and γ stand for set of states, set of actions, initial state distribution, environmental dynamic represented as conditional state distribution, reward functions, and discount factor, respectively. The environment interactions between a subject and its environment is unbounded in the continuing tasks; and thus the returns are defined as $R_t = \sum_{i=t}^{\infty} \gamma^{i-t} r(s_i, a_i, s_{i+1})$. The objective of MDP is to find a policy that maximizes the expected returns.

Generative Adversarial IL (GAIL)

GAIL is used for reward function in this paper. Based on GAN, the GAIL trains a binary classifier, $D(s, a)$, referred to as the discriminator, to distinguish between transitions sampled from an expert demonstration and those generated by the trained policies. With GAIL, an agent is able to learn a policy that imitates expert demonstrations using the adversarial network. The objective of GAIL is defined as follows:

$$\underset{\theta}{\operatorname{argmin}} \underset{\phi}{\operatorname{argmax}} \{\mathbb{E}_{\pi_\theta} [\log \mathcal{D}_\phi(s, a)] + \mathbb{E}_{\pi_E} [\log(1 - \mathcal{D}_\phi(s, a))] - \lambda H(\pi_\theta)\}. \quad (1)$$

where π_θ, π_E are the policy which is parameterized by θ and an expert policy. In (1), $H(\pi_\theta) \triangleq \mathbb{E}_\pi [-\log \pi(a|s)]$ is entropy regularization. $\mathcal{D}_\phi(s, a) \to [0, 1]$ is the discriminator parameterized by ϕ [Ho and Ermon, 2016]. In GAIL, the policy is instead provided a reward for confusing the discriminator, which is then maximized via some on-policy RL optimization schemes. The $\mathcal{D}_\phi$ takes the role of a reward function; and thus it gives learning signal to the policy [Ho and Ermon, 2016; Guo *et al.*, 2018; Henderson *et al.*, 2017a].

Augmented Random Search (ARS)

ARS is a model-free RL algorithm. Based on random search in the parameter spaces of policies, ARS uses the method of finite differences to adjust its weights and learn the way how the policy performs its given tasks [Matyas, 1965; Mania *et al.*, 2018]. Through the random search in the parameter spaces, the algorithm can conduct a derivative-free optimization with noises [Matyas, 1965; Mania *et al.*, 2018]. To update the weights effectively, ARS selects update directions uniformly and updates the policies along with the selected direction. For updating the parameterized policy π_θ, the update direction is as $\frac{r(\pi_{\theta-\nu\delta}) - r(\pi_{\theta+\nu\delta})}{\nu}$ where δ is a zero mean Gaussian vector, ν is a positive real number which represents the standard deviation of exploration noise, and $r(\pi_\theta \pm \nu\delta)$ means the rewards from environments when the parameter of policies is $\pi_\theta \pm \nu\delta$. Let θ_t be the weight of policy at t-th training iteration. N denotes that the number of sampled directions per iteration. The update step is configured as:

$$\theta_{t+1} = \theta_t + \frac{\alpha}{N} \sum\nolimits_{i=1}^{N} [r(\pi_{\theta+\nu\delta_i}) - r(\pi_{\theta-\nu\delta_i})] \, \delta_i. \quad (2)$$

However, the problem of random search in the parameter spaces of policies is large variations in terms of the rewards $r(\pi_\theta \pm \nu\delta)$ which are observed during training procedure. The variations make the updated policies to be perturbed through the update steps [Mania *et al.*, 2018]. To address the large variation issue, the standard deviation σ_R of the rewards which is collected at each iteration is used to adjust the size of the update steps in ARS. Based on the adaptive step size, ARS shows better performance compared to DRL algorithms (i.e., PPO, TRPO, etc.) in specific environments.

4 Problem Definition

Motivation

By coordinating the ADAS functions in the limited situations such as highways, the autonomous driving can be realized. To coordinate the ADAS functions for autonomous driving, the supervisor determines the appropriate ADAS functions based on the nearby situations. However, the complete states of the environment are not known to the autonomous vehicle supervisor. The supervisor receives an observation that is conditioned on the current state of the system. The host vehicle interacts with the environment including surrounding vehicles and lanes; and thus it uses partially observable local information. Therefore, we need to model the observation of agent as an (O, A, T, R, γ) tuple representing a partially observable Markov decision process with continuous observations and actions for autonomous driving. Similar to MDP in Section 3, there are the set of partial observation states denoted by O, instead of S. In this paper, LIDAR data is regarded as the observation by vehicles.

In this paper, a finite state space $\mathcal{O} \in \mathbb{R}^n$ and a finite action space $\mathcal{A} \in \mathbb{R}^p$ are considered. The goal of IL for autonomous driving is to learn a policy $\pi_\theta \in \Pi : \mathcal{O} \times \mathcal{A} \to \mathbb{R}^p$ which imitates expert demonstration from GAN $\mathcal{D}_\phi(s, a) \to [0, 1]$ where $\theta \in \mathbb{R}^n$ are the policy parameters and $\phi \in \mathbb{R}^{n+p}$ are the discriminator parameters [Ho and Ermon, 2016].

The State Space

For sensor model, we use a vector observation that consists of LIDAR sensor data. In particular, N beams are spread evenly over the field of view $[\omega_{min}, \omega_{max}]$. The LIDAR sensor detects around the vehicle. Each sensor data has a maximum range of r_{max}. The sensor returns the distance between the first obstacle it encounters and the host vehicle, or r_{max} if no obstacle is detected. Then, the observation is described as $O = (o_1, \ldots, o_N)$. Furthermore, based on the distance information, the relative speed of the obstacle and the host vehicle can be calculated. Here, the relative speed observation is described as $V_r = (v_1, \ldots, v_N)$.

The Action Space

The policy is considered as a high-level decision maker which determines optimal actions based on observation on the highway. We assume that the autonomous vehicle utilizes the ADAS functions; and thus the determined actions of driving policy activate each ADAS function. The driving policy is defined in a discrete action space. The high level decisions can be break down into the following 5 actions as follows: (1) maintain current status, (2) accelerate speed for a constant amount $vel_{cur} + vel_{acc}$, (3) decelerate speed for a constant amount $vel_{cur} - vel_{dec}$, (4) make a left lane change, (5) make a right lane change. The actions expect that the vehicle is adjusted with autonomous emergency braking (AEB) and adaptive cruise control (ACC) [Mukadam *et al.*, 2017; Min and Kim, 2018; Hoel *et al.*, 2018].

The Reward Function

In the GAIL framework, the reward from adversarial network is defined $r_{\pi_\theta}(s, a) = -\log(1 - \mathcal{D}_\phi(s, a))$ or $r_{\pi_\theta}(s, a) = \log(\mathcal{D}_\phi(s, a))$ [Ho and Ermon, 2016]. The former type of the reward is used to encourage agent to train survival policies

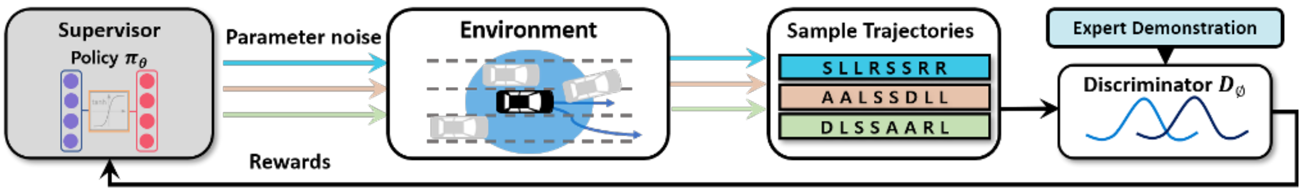


Figure 2: Structure of RAIL.

through a survival bonus in the form of positive reward based on their lifetime. The latter is often used to train policies with a per step negative reward, when a reward function consists of a negative constant for the state and action. However, in this case, it is hard to learn the survival policies [Kostrikov *et al.*, 2019]. The prior knowledge of environmental objectives is important, but the environment-dependent reward function is undesirable when the agent requires interactions with a training environment in order to imitate an expert policy. Therefore, we defined the reward function as follows: $\log(\mathcal{D}_\phi(s,a)) - \log(1-\mathcal{D}_\phi(s,a))$.

5 Randomized Adversarial IL (RAIL)

The approach in this paper, named *randomized adversarial imitation learning (RAIL)*, adopts IL through adversarial network paradigm (i.e., GAIL). The main concept of RAIL is enhance an conventional algorithm called ARS and GAIL [Ho and Ermon, 2016; Mania *et al.*, 2018]. RAIL aims to train the driving policy π_θ to imitate expert driver's demonstration. This section describes the details of RAIL and makes a connection between GAIL and a derivative-free optimization.

In Fig. 2, the overall structure of RAIL is described. The supervisor of host vehicle is considered as an agent which has policy π_θ. From the environment (i.e., multi-lane highway), the host vehicle receives the observation. Then, the random noise matrices of small values can be generated. The noise matrices are added to or subtracted from the policy parameters θ. As a result, several different temporary policies are produced. The agent interacts with the environments multiple times based on the generated noisy policies and the results are collected as sample trajectories. Based on the samples, the main policy π_θ is trained to control the autonomous driving successfully with fully utilizing ADAS functions which guarantee safety. In the training process, the policy π_θ attempts to fool a discriminator $\mathcal{D}_\phi$ into believing the sample trajectories of the agent come from expert demonstrations. The $\mathcal{D}_\phi$ tries to distinguish between the distribution of trajectories which are sampled by the policies π_θ and the expert trajectories $\mathcal{T}_E$. The trajectories consist of state-action pair (s,a). The discriminator takes the role of the reward module in RAIL, as shown in Fig. 2; and thus the policy π_θ is trained against the discriminator. Therefore, the performance of the discriminator has a significant impact on convergence and agent.

As shown in Fig.2, the discriminator is trained based on sample trajectories and expert demonstration. However, in training procedure, since the policy π_θ is updated every iteration, the distribution of the sample trajectories changes. As a result, the training of the discriminator is not stabilized; and thus it gives the inaccurate reward signal to the policy. Consequently, the policy can be perturbed during update step [Guo *et al.*, 2018]. In RAIL, the loss function of least square GAN (LS-GAN) is used to train a discriminator $\mathcal{D}_\phi$ [Mao *et al.*, 2017], and the objective function of the discriminator is as follows:

$$\begin{aligned}\underset{\phi}{\operatorname{argmin}}\, L_{LS}(\mathcal{D}) = \frac{1}{2}\mathbb{E}_{\pi_E}\left[(\mathcal{D}_\phi(s,a)-b)^2\right] + \\ \frac{1}{2}\mathbb{E}_{\pi_\theta}\left[(\mathcal{D}_\phi(s,a)-a)^2\right]\end{aligned} \tag{3}$$

where a and b are the discriminator labels for the sampled trajectories from the policy π_θ and the expert trajectories.

In this paper, least-squares loss function is used to train the discriminator. When the loss function of original GAN Eq.1 is used, sampled trajectories which are far from the expert trajectories but on the correct side of the decision boundary are almost not penalized by sigmoid cross-entropy loss. In a contrast, the least-squares loss function (3) penalizes the sampled trajectories which are far from the expert trajectories on either side of decision boundary [Mao *et al.*, 2017]. Therefore, the stability of training is improved; and it leads the discriminator to give accurate reward signals to the update step. In LS-GAN, a and b have relationship $b - a = 2$ for (3) to be Pearson $\mathcal{X}^2$ divergence [Mao *et al.*, 2017]. However, we use $a = 0$ and $b = 1$ as the target discriminator labels. The results of the discriminator $\mathcal{D}_\phi$ are in the range of 0 to 1 (experimentally determined). In RAIL, the discriminator is interpreted as a reward function for policy optimization. Forementioned in Sec.4, the form of reward signal is as follows:

$$r_{\pi_\theta}(s,a) = \log(\mathcal{D}_\phi(s,a)) - \log(1-\mathcal{D}_\phi(s,a)) \tag{4}$$

This means that if the trajectories sampled from the policy π_θ is similar to expert trajectories, the policy π_θ gets higher reward $r_{\pi_\theta}(s,a)$. The policy π_θ is updated to maximize the discounted sum of rewards given by the discriminator rather than the reward from the environment as shown in Fig. 2. The objective of RAIL can be described as $\underset{\theta}{\operatorname{argmax}}\, \mathbb{E}_{(s,a)\sim\pi_\theta}[r(s,a)]$, and then, it is as follows by (4):

$$\underset{\theta}{\operatorname{argmax}}\, \mathbb{E}_{(s,a)\sim\pi_\theta}\left[\log(\mathcal{D}_\phi(s,a)) - \log(1-\mathcal{D}_\phi(s,a))\right] \tag{5}$$

where this (5) represents the connection between adversarial IL and randomized parameter space search in RAIL.

Algorithm

As mentioned, RAIL is related to ARS which is model-free reinforcement algorithm. Thus, RAIL utilizes parameter space exploration for a derivative-free policy optimization. The parameters of π_θ are denoted by θ. The π_θ consists of π_θ^i,

Algorithm 1: RAIL

```
Hyperparameters: $\alpha$ step size, $N$ number of sampled directions per iteration, $\delta^i$ and $\delta^o$ Gaussian
                 vectors from zero mean and $\nu$ a positive real number standard deviation of the
                 exploration noise, $h$ hidden layer size
Initialize      : $\theta^i$, $\theta^o$ from behavior cloning,
                  $\mu_0 = \mathbf{0} \in \mathbb{R}^n$, and $\sum_0 = \mathbf{I}_n \in \mathbb{R}^{n\times n}$
1  while $t \leq$ Episode Length do
2      i.i.d. Random Sampling with
         $\delta_t = \{\delta_1, \delta_2, ..., \delta_N; \delta_k^i \in \mathbb{R}^{h\times n}, \delta_k^o \in \mathbb{R}^{p\times h}\}$
3      Collect $2N$ rollouts and corresponding rewards using the
         $2N$ noisy policies for $k \in \{1, 2, \ldots, N\}$.
4          $\pi_{t,(k),+} = (\theta_t + \nu\delta_k)\text{diag}(\sum_t)^{-1/2}(s - \mu_t)$
5          $\pi_{t,(k),-} = (\theta_t - \nu\delta_k)\text{diag}(\sum_t)^{-1/2}(s - \mu_t)$
6      Update discriminator parameter $\phi_t$ :
7      $\nabla_{\phi_t} L_{LS} = \frac{1}{2}\mathbb{E}_{\pi_E}\left[(\nabla_{\phi_t}\mathcal{D}_{\phi_t}(s,a) - b)^2\right]$
8                  $+\frac{1}{2}\mathbb{E}_{\pi_\theta}\left[(\nabla_{\phi_t}\mathcal{D}_{\phi_t}(s,a) - a)^2\right]$
9      Update the policy parameter $\theta_t$ :
10     $\theta_{t+1} = \theta_t + \frac{\alpha}{N\sigma_R}\sum_{i=1}^{N}\left[r(\pi_{t,(k),+}) - r(\pi_{t,(k),-})\right]\delta_{(k)}$
11     where trajectories $T$ sampled from $\pi_{(t,(k),\pm)}$
12     $r(\pi_{t,(k),\pm}) =$
         $\mathbb{E}_{(s,a)\sim\pi_{t,(k),\pm}}\left[\log(\mathcal{D}_\phi(s,a)) - \log(1 - \mathcal{D}_\phi(s,a))\right]$
13     Set $\mu_{t+1}$, $\sum_{t+1}$ to be the mean and covariance of the
         states encountered from the start of training.
14     $t = t + 1$
15 end
```

π_θ^o, and activation function. The $\pi_{\theta^i}^i$ is the input layer of π_θ where $\theta^i \in \mathbb{R}^{n\times h}$ are the parameters of the input layer. In addition, $\pi_{\theta^o}^o$ is the output layer where $\theta^o \in \mathbb{R}^{h\times p}$. The noises δ^i and δ^o of parameter space for exploration are $n \times h$ and $h \times p$ matrices where they are sampled from zero mean and ν standard deviation Gaussian distribution. In this paper, let θ be a set of θ^i and θ^o. δ means a set of δ^i and δ^o.

The pseudo-code of RAIL is represented in Algorithm 1. The policy parameters θ^i and θ^o are initialized from behavior cloning. In training procedure, the noises $\delta^{\mathbf{i}} and \delta^{\mathbf{o}}$ which mean the search directions in parameter space of policy are chosen randomly for each iteration (line 2). Each set of selected N noises makes two policies in the current policy π_θ. We collect $2N$ rollouts and rewards from N noisy policies $\pi_{t,k,\pm} = \theta_t \pm \nu\delta_k$ (line 3-6). The high dimensional complex problems have multiple state components with various ranges; and thus it makes the policies to result in large changes in the actions when the same sized changes are not equally influence state components. Therefore, the state normalization is used in RAIL (line 4-5,14); and it allows policy $\pi_{t,i,\pm}$ to have equal influence for the changes of state components when there are state components with various ranges [Mania *et al.*, 2018; Salimans *et al.*, 2017; Nagabandi *et al.*, 2018]. The discriminator $\mathcal{D}_\phi$ gives the reward signal to update step. However, since the trajectories for the training of the discriminator can only be obtained from current policies π_{θ_t}, a discriminator is trained whenever the policy θ_t is updated. The discriminator $\mathcal{D}_\phi$ finds the parameter ϕ which minimizes the objective function (3) (line 7-9).

Average	RAIL (Stacked)	RAIL (Linear)	Expert
Speed [km/h]	70.38	65.00	68.83
# Overtake	45.04	40.03	44.48
# Lane change	15.01	13.05	14.04
Longitudinal	2719.38	2495.57	2642.11
Lateral	-122.98	-175.6	-132.52

Table 1: Performance (Avg. 16 Episodes, 40 Trajectories)

By using the reward signals from the discriminator, the policy weight is updated in the direction of $+\delta$ or $-\delta$ based on the result of $r(\pi_{t,(k),+}) - r(\pi_{t,(k),-})$ (line 10-13). The state normalization is based on the information of the states encountered during training; and thus μ and $\sum$ are updated (line 14).

6 Experiments

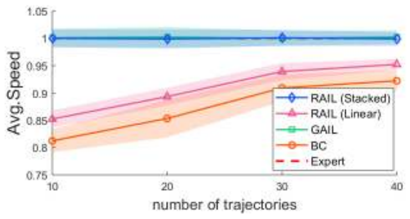
In this section, we compare the performance between RAIL and baselines. Furthermore, in order to assess the performance gaps between the single-layer and multi-layer policies trained by RAIL, the single-layer and two-layer (i.e., multi-layer) policies was implemented.

Simulator

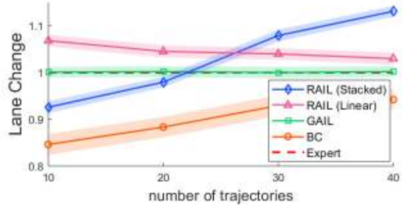
The simulated road environment is a highway driving roadway composed of five lanes. Other vehicles are generated in the center of the random lanes within a certain distance to the host vehicle. In addition, it is assumed that other vehicles do not collide with others while randomly changes the lanes. Aforementioned in Sec. 4, the observation is based on LIDAR sensor result. We assume LIDAR sensor detects a range of 360 degrees with one ray per 15 degree. The ray returns the distance between the first obstacle it encounters and the host vehicle. If there are no obstacles, it returns the maximum sensing range. We make the expert demonstration using PPO with specific action controls. The results present the average of 16 experiment results. In the experiments, the trained weights by BC are used to fast convergence in GAIL and RAIL. This simulation is inspired by [Min and Kim, 2018]. We implemented the RAIL simulator based on Unity.

Results

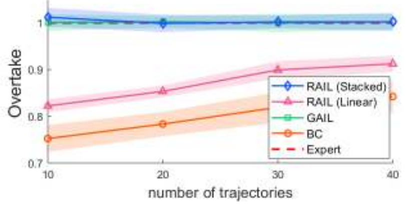
The purpose of experiments in Fig. 3 is to show the sample efficiency. In order to assess the efficiency, average speed, number of lane changes, number of overtakes, longitudinal reward, and lateral reward were considered as shown in Fig. 3 and Fig. 4. In Table 1, it can be seen that the two-layer policy resulted in the highest values of average speed and average overtaking statistics where the values are $70.38\,\text{km/h}$ and 45.04, respectively. This is because the trained policies sometimes can achieve higher performance than experts since GAIL-based frameworks perform policy optimization based on the interaction with the environment. On the other hand, the performance of single-layer policy shows 90% performance compared to expert. This is because the single-layer is not enough to handle high dimensional observations properly. Aforementioned, BC tries to minimize 1-step deviation error along the expert demonstration. As a result, the single-layer policy shows undesirable performance due to distribution mismatch between training and testing. In Fig. 4, a longitudinal reward is used to analyze the environmental reward.



(a) Normalized Speed.

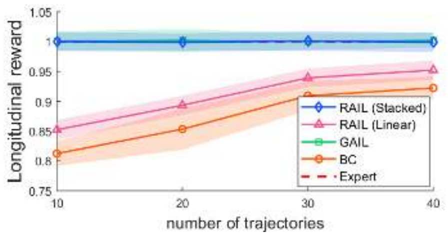


(b) Normalized Lane change.

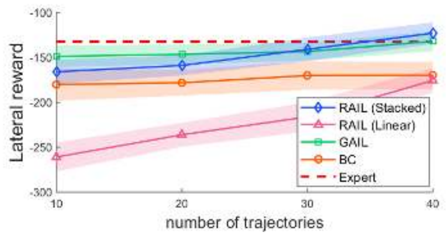


(c) Normalized Overtake.

Figure 3: The performance of trained policy according to the set number of expert trajectories (Average of 5 episodes).



(a) Normalized Longitudinal Rewards.



(b) Lateral Rewards.

Figure 4: The rewards of trained policy according to the set number of expert trajectories (Average of 5 episodes).

The longitudinal reward is proportional to the speed; and thus the normalized result shows the same result as the average speed as shown in Fig 3a. In order to assess sensitivity to action decisions, a lateral reward was used. Until the lane change is completed, the host vehicle can change the decision according to the observation. Because the lateral reward occurs continuously during lane change, the frequent changes during the operation lead to reward reduction. In Fig. 4b, the two-layer policy obtains a large lateral reward in the last case. However, the two-layer policy shows more lane changes than the expert. This is because the two-layer policy is less likely to change the decision during the operation. On the other hand, the single-layer policy shows the frequent lane changes than the expert. The single-layer policy obtains the smallest lateral reward. This is because the single-layer policy changes its decision frequently. BC shows the least number of lane changes. However, the trained policy obtains larger reward than the single-layer policy trained by RAIL. The number of lane change is considerably smaller than the single-layer policy; and thus it leads to the trained policy obtains larger lateral reward than the single-layer policy. The experiment of Fig. 3c was conducted to measure appropriate decisions to imitate the expert demonstration. In order to achieve the similar number of overtakes as the expert, the lane change point and decision should be similar to the expert during the simulation. In Fig. 3c, the two-layer policy shows the desired performance compared to expert. This result is related to the tendency (i.e., meaningless lane change and decision change) which is shown in Fig. 3b and Fig. 4b. Furthermore, the decision points and actions are similar to the expert. However, the single-layer policy shows a lower number of overtakes than the expert because the average speed is low as well as makes inappropriate lane change decisions based on observation.

In summary, we verified that the proposed RAIL improves the average speed and reduces the number of unnecessary lane changes rather than BC. This means that the RAIL trains driving policies in the correct directions. The experimental results show that the two layer policy achieves desired performance similar to driving experts.

7 Conclusion

This paper proposes randomized adversarial imitation learning (RAIL) for effect autonomous driving policy training which utilizes ADAS functions to guarantee the safety of vehicles. The RAIL is not only a derivative-free but also the simplest model-free algorithm. Through the proposed algorithm, the policies that successfully drive autonomous vehicles are trained via derivative free optimization. During the training procedure, the simple update step makes the algorithm to be facile; and thus it makes the reconstruction results which get reasonable performance easily. By comparing the performance of the proposed model with complex deep reinforcement learning based methods, we demonstrate that the proposed RAIL trains the policies that achieve desired performance during autonomous driving. This results can be a breakthrough to the common belief that random search in the parameter space of policies can not be competitive in terms of performance. The evaluation shows the possibility that autonomous vehicles can be controlled by the RAIL policies.

Acknowledgments

This research was supported by IITP (MSIT 2018-0-00170, Virtual Presence in Moving Objects through 5G) and (MSIP 2017-0-00068, A Development of Driving Decision Engine for Autonomous Driving using Driving Experience Information). J. Kim is a corresponding author.

References

[Ahmed, 1999] Kazi I. Ahmed. *Modeling drivers' acceleration and lane changing behavior*. PhD thesis, MIT, 1999.

[Finn *et al.*, 2016a] Chelsea Finn, Paul Christiano, Pieter Abbeel, and Sergey Levine. A connection between generative adversarial networks, inverse reinforcement learning, and energy-based models. In *NIPS Workshop on Adversarial Training*, 2016.

[Finn *et al.*, 2016b] Chelsea Finn, Sergey Levine, and Pieter Abbeel. Guided cost learning: Deep inverse optimal control via policy optimization. In *ICML*, 2016.

[Gipps, 1986] Peter G Gipps. A model for the structure of lane-changing decisions. *Transportation Research Part B: Methodological*, 1986.

[Guo *et al.*, 2018] Yijie Guo, Junhyuk Oh, Satinder Singh, and Honglak Lee. Generative adversarial self-imitation learning. *arXiv preprint arXiv:1812.00950*, 2018.

[Henderson *et al.*, 2017a] Peter Henderson, Wei-Di Chang, Pierre-Luc Bacon, David Meger, Joelle Pineau, and Doina Precup. OptionGAN: Learning joint reward-policy options using generative adversarial inverse reinforcement learning. *arXiv preprint arXiv:1709.06683*, 2017.

[Henderson *et al.*, 2017b] Peter Henderson, Riashat Islam, Philip Bachman, Joelle Pineau, Doina Precup, and David Meger. Deep reinforcement learning that matters. *arXiv preprint arXiv:1709.06560*, 2017.

[Ho and Ermon, 2016] Jonathan Ho and Stefano Ermon. Generative adversarial imitation learning. In *NIPS*, 2016.

[Hoel *et al.*, 2018] Carl-Johan Hoel, Krister Wolff, and Leo Laine. Automated speed and lane change decision making using deep reinforcement learning. *arXiv preprint arXiv:1803.10056*, 2018.

[Islam *et al.*, 2017] Riashat Islam, Peter Henderson, Maziar Gomrokchi, and Doina Precup. Reproducibility of benchmarked deep reinforcement learning tasks for continuous control. *arXiv preprint arXiv:1708.04133*, 2017.

[Korssen *et al.*, 2018] Tim Korssen, Victor Dolk, Joanna van de Mortel-Fronczak, Michel Reniers, and Maurice Heemels. Systematic model-based design and implementation of supervisors for advanced driver assistance systems. *IEEE TITS*, 2018.

[Kostrikov *et al.*, 2019] Ilya Kostrikov, Kumar Krishna Agrawal, Debidatta Dwibedi, Sergey Levine, and Jonathan Tompson. Discriminator-actor-critic: Addressing sample inefficiency and reward bias in adversarial imitation learning. In *ICLR*, 2019.

[Mania *et al.*, 2018] Horia Mania, Aurelia Guy, and Benjamin Recht. Simple random search provides a competitive approach to reinforcement learning. *arXiv preprint arXiv:1803.07055*, 2018.

[Mao *et al.*, 2017] Xudong Mao, Qing Li, Haoran Xie, Raymond Lau, Zhen Wang, and Stephen Paul Smolley. Least squares generative adversarial networks. In *ICCV*, 2017.

[Matyas, 1965] J Matyas. Random optimization. *Automation and Remote Control*, 26(2):246–253, 1965.

[Min and Kim, 2018] Kyushik Min and Hayoung Kim. Deep Q-learning based high level driving policy determination. In *IV*, 2018.

[Mnih *et al.*, 2015] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 2015.

[Mukadam *et al.*, 2017] Mustafa Mukadam, Akansel Cosgun, Alireza Nakhaei, and Kikuo Fujimura. Tactical decision making for lane changing with deep reinforcement learning. In *NIPS Workshop MLITS*, 2017.

[Nagabandi *et al.*, 2018] Anusha Nagabandi, Gregory Kahn, Ronald Fearing, and Sergey Levine. Neural network dynamics for model-based deep reinforcement learning with model-free fine-tuning. In *ICRA*, 2018.

[Pan *et al.*, 2018] Yunpeng Pan, Ching-An Cheng, Kamil Saigol, Keuntaek Lee, Xinyan Yan, Evangelos Theodorou, and Byron Boots. Agile autonomous driving using end-to-end deep imitation learning. In *RSS*, 2018.

[Pomerleau, 1989] Dean Pomerleau. Alvinn: An autonomous land vehicle in a neural network. In *NIPS*, 1989.

[Pomerleau, 1991] Dean Pomerleau. Rapidly adapting artificial neural networks for autonomous navigation. In *NIPS*, 1991.

[Rajeswaran *et al.*, 2017] Aravind Rajeswaran, Kendall Lowrey, Emanuel V Todorov, and Sham Kakade. Towards generalization and simplicity in continuous control. In *NIPS*, 2017.

[Salimans *et al.*, 2017] Tim Salimans, Jonathan Ho, Xi Chen, Szymon Sidor, and Ilya Sutskever. Evolution strategies as a scalable alternative to reinforcement learning. *arXiv preprint arXiv:1703.03864*, 2017.

[Sallab *et al.*, 2016] Ahmad El Sallab, Mohammed Abdou, Etienne Perot, and Senthil Yogamani. End-to-end deep reinforcement learning for lane keeping assist. *arXiv preprint arXiv:1612.04340*, 2016.

[Schulman *et al.*, 2015] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *ICML*, 2015.

[Schulman *et al.*, 2017] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[Silver *et al.*, 2016] David Silver, Aja Huang, Chris Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of Go with deep neural networks and tree search. *Nature*, 2016.

[Ziebart *et al.*, 2008] Brian D Ziebart, Andrew L Maas, J Andrew Bagnell, and Anind K Dey. Maximum entropy inverse reinforcement learning. In *AAAI*, 2008.