2019 IEEE Second International Conference on Artificial Intelligence and Knowledge Engineering (AIKE)

Adversarial Imitation Learning between Agents with Different Numbers of State Dimensions*

Taketo Yoshida and Yasuo Kuniyoshi
School of Information Science and Technology, The University of Tokyo,
Japan.
{yoshida, kuniyosh}@isi.imi.i.u-tokyo.ac.jp

***Abstract*—Designing a reward is difficult in reinforcement learning because there is often a tradeoff between the two roles played by the reward, one being to make an agent behave as desired and the other being to allow an agent to learn a policy easily. Meanwhile, adversarial imitation learning can eliminate the design of a reward by estimating the reward that the expert is maximizing from trajectories of the expert. However, previous methods of adversarial imitation learning can be applied only when the number of state dimensions is the same for the agent and expert. We here define the new problem of adversarial imitation learning between agents having different numbers of state dimensions and propose a method of solving the problem. Our method makes the numbers of dimensions equal by inserting an arbitrary constant into the smaller state so that the feature extractor can be shared between agents. In addition, we verify whether domain adaptation can extract a feature independent of the number of dimensions. We find that the imitation rate of our method is at least 80% in a simulation of a reaching task while rates of previous methods are no more than 55%. We also find that our method is robust against the value to be inserted into the state, the dimension into which the value is inserted, and the numbers of joints of the agent and expert.**

***Index Terms*—imitation learning, reinforcement learning, inverse reinforcement learning**

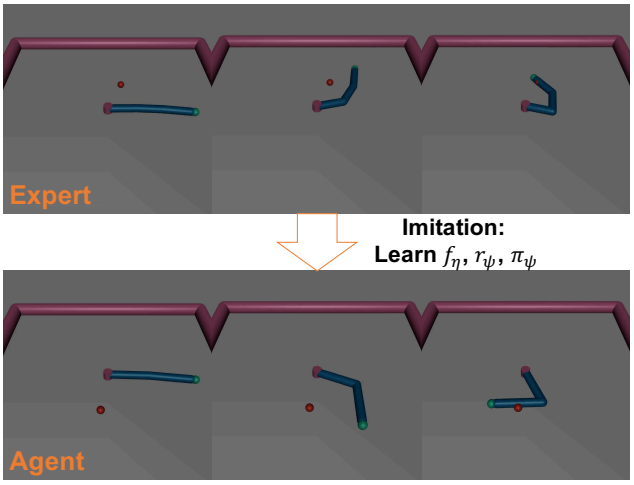

Fig. 1. The target agent of the two-joint reacher imitates the source expert of the three-joint reacher using our method.

I. INTRODUCTION

Impressive advances have been made in deep reinforcement learning. Deep reinforcement learning can solve problems that cannot be resolved by existing rule-based algorithms and control methods in many fields, such as [1], video games [2], and robotic control [3]. Deep reinforcement learning combines the strengths of deep learning, which is a machine learning method capable of automatically extracting features, and reinforcement learning, which is a control method that learns the optimal control policy automatically by trial and error within the environment.

However, it is difficult to design the reward for deep reinforcement learning. A reward function has two roles. One is to facilitate the learning of policy and the other is to make the behaviors of the learned policy consistent with the designer's intended behavior. There is often a trade-off between these two roles. To facilitate the learning of a policy, when a dense reward is designed and subgoals are set, the agent often sticks to only the subgoals and falls into a local minimum [4] [5]. Meanwhile, the agent cannot learn when a sparse reward is designed to make the behavior of the learned policy intentional; e.g., DQN [6] could not solve the game Montezuma's Revenge although it was able to solve almost all other Atari games. The reward function therefore needs to be carefully designed and many previous studies have tuned many coefficients finely and let the agent solve tasks [5] [7] [8] [9]. To reduce the labor of designing the reward function, it is powerful to estimate the reward function from the expert's trajectory using inverse reinforcement learning (IRL) [10] [11] [12] [13] [14]. By alternately performing reinforcement learning and inverse reinforcement learning, adversarial imitation learning (AIL), such as generative adversarial imitation learning (GAIL) [15] and adversarial inverse reinforcement learning (AIRL) [16], can imitate expert trajectories robustly.

The imitation of an expert with a different body shape is an effective means of understanding tasks and learning how to solve tasks; e.g., human infants imitate adults by watching adults performing tasks [17]. Reward design becomes unnecessary if agents learn a reward function from human demonstrations and learn the policy. However, conventional AIL methods are adaptable only when the numbers of state dimensions of the experts to be imitated and the agent of the

This work was supported in parts by JSPS KAKENHI Grant JP18H04108, and a donation from NVIDIA corporation.

978-1-7281-1488-0/19/$31.00 ©2019 IEEE
DOI 10.1109/AIKE.2019.00040

imitation principal are exactly the same. It is impossible to imitate the trajectory of experts with a different number of joints using these methods.

The present paper formulates the problem of imitating an expert with a different number of state dimensions. Because agents are not taught what tasks are, it is necessary to extract feature quantities useful for tasks from the trajectory of the expert. The agent therefore needs to learn three functions, namely the feature extractor, reward, and policy. We present a novel approach of sharing feature extractors between experts and agents through state shaping, inserting arbitrary constants into missing state dimensions. We additionally present another method using domain adaptation to eliminate the effect for the difference in the number of state dimensions in addition to state shaping.

The main contributions of our work are the formulation of imitation learning in the case that the expert has a different number of state dimensions and an algorithm that can be used to learn alternately the feature extractors that extract useful features for tasks, reward functions, and policy. To evaluate the performance of our imitation learning method, we simulate a reaching task where the expert has more dimensions than the agent.

Our results show that the imitation rate of our method using only state shaping with one sharing feature extractor for agents is 80% or more in the simulation of a reaching task while the rates of previous methods with separate feature extractors for each agent is no more than 55%. Our results also show that our method is robust against the value to be inserted into the state, the dimension into which the value is inserted and the number of joints of agent and expert.

II. RELATED WORK

In reinforcement learning, it is difficult to design the reward when trying to let the agent solve a complicated task. In research on the manipulation of a robot hand [18], the value of the reward and the conditions under which the reward can be obtained are finely adjusted. To remove the need for such efforts, inverse reinforcement learning automatically learns a reward function from expert trajectories. Adopting IRL, a reward can be automatically learned from expert trajectories [10] [11] [12] [13]. Finn et al. proposed a method called guided cost learning (GCL), which is one of the methods of IRL, and showed that a robot can learn the cost and policy of placing dishes from demonstrations. GCL is based on the guided policy search [19] [20], which is one of the methods of model-based reinforcement learning [21] [22] [23] [24] [25] [26], and is thus is a model-based approach of IRL. Meanwhile, model-free approaches of IRL, such as GAIL [15] and AIRL [16], have recently been studied for imitation learning. These methods train a reward function and a policy in an adversarial manner like Generative Adversarial Networks [27]. Agents can achieve the same performance level as experts even in continuous and high-dimensional state action spaces. AIRL can learn a reward function from an expert trajectory even with a different state transition probability, such as in the case of different lengths of legs, but is not applicable to an expert trajectory with different state dimensions in contrast with our method.

Human expert demonstrations whose number of state dimensions are different from that of agents have been used in the field of reinforcement learning. Studies have used human expert data to initialize a policy by supervised learning and started reinforcement learning from the initialized policy [1]. The policy initialized by human expert data performs well in the range of states included in expert data but not in states not included in expert data. Reinforcement learning is used to ensure a policy performs well in those states. In research on learning the complex dexterous manipulation of a robot hand [5], the loss of a policy has two parts, that of supervised learning by human demonstration and that of reinforcement learning by rewards. However, to supervise policy through expert data or human demonstration, the action space and state space must be the same for the environment in which an agent interacts and the environment in which the experts' data are collected. Other studies used human data to design a reward function [7] [9]. Stable [7] or dynamic [9] whole-body motions can be realized when human motion capture data are used in designing rewards. However, these studies, carefully tuned coefficients of rewards through much trial and error. In contrast, we use expert data in automatically learning a reward function.

Imitation learning from visual inputs has long been studied [28] [29] [30] [31] [32] [33], especially in the field of mobile robots [34] [35] [36] [37]. Yu et al. proposed a method that can quickly imitate the moves of human demonstrations [38] through model agnostic meta learning [39] for fast adaptation. However, the cited studies used not IRL but a method of supervised learning called behavioral cloning. Behavioral cloning cannot solve a complicated problem whose state action spaces are large because agents cannot understand how to act in states not included in expert trajectories, which are common in the problem of large state action spaces [37]. In contrast, we use IRL to learn a reward and policy by interacting in an environment.

Skill transfer in reinforcement learning is similar to our work. Gupta et al. proposed a method to transfer knowledge about how to solve a task from a source agent with a different body shape [40]. They used autoencoders [41] to extract features invariant between source and target agents. Autoencoders are trained using trajectories from both source and target tasks in proxy tasks that have a feature effective for solving a task that is in common with a target task. However, their methods are not imitation learning but reinforcement learning because an agent can receive an external reward designed by humans as well as an internal reward of similarity to a source agent. Additionally, we do not assume a proxy task.

We assume the problem that the source agent imitates the expert or a target agent and that source and target agents have different body shapes. This difference can be viewed as a domain shift. A domain shift can be solved in two main ways, one being domain randomization and the other being

domain adaptation. Domain randomization is a technique that generalizes a model to a class of domain to randomize specific factors of inputs. In reinforcement learning, domain randomization is often used to solve the reality gap of a simulator [42] [43] [8] [44] [45] [46] [47]. This is a method of dealing with the problem that high performance is not achieved when the policy learned by the simulator is tested in the real world using the difference between the simulator and real world. The method learns a policy of responding to any parameter in a class by randomizing parameters, such as the friction, control delay, and observation noise, in various environments. Domain adaptation is a technique that allows a model to quickly adapt to a target domain [48] [49] [50] [51] [52]. Bousmalis et al. proposed a method of rapidly adapting the policy of a grasping robot learned in a simulator to the real world [53]. This problem is called unsupervised domain adaptation (UDA) when labels of the source domain can be obtained but those of the target domain cannot [54] [55] [56] [57] [58]. In our setting, there are experts of a target agent but no experts of a source agent. There are target domain data and labels of the only agent class and no data or labels of the expert class in our setting, while there are target domain data of all classes and no labels of such data in UDA. Ganin et al. proposed domain adversarial neural networks (DANNs) to make the intermediate representation of the neural network invariant to the domain [56]. This is a method of adversarial training between a classifier branching off from the intermediate layer and classifying the domains and a feature extractor consisting of layers up to the intermediate layer. We investigate the method using UDA.

III. ADVERSARIAL IMITATION LEARNING

In GAIL, there is a neural network representing the policy π_θ and the classifier D_ψ, and the following objective function is optimized in a minimax game.

$$\begin{aligned}&\text{argmin}_\theta \text{argmax}_\psi L(\theta,\psi) = \\ &E_{\pi_E}[logD_\psi(s,a)] + E_{\pi_\theta}[log(1-D_\psi(s,a))] - \lambda\mathcal{H}(\pi_\theta)\end{aligned} \tag{1}$$

The policy π_θ is learned to maximize the cumulative reward, assuming that the immediate reward function $r(s,a)$ is $-log(1-D_\psi(s,a))$ Learn policies using on-policy methods such as trust region policy optimization (TRPO) [59] and proximal policy optimization (PPO) [60]. Meanwhile, $D_\psi(s,a)$ is a classifier that outputs the probability that the pair of the state s and action a is an expert. At the end of optimization, $D_\psi(s,a)$ can be interpreted as implicitly restoring the reward function inside the expression of the discriminator. In GAIL, the reward function is not explicitly learned but learned behind in the classifier. It is therefore difficult to newly learn a policy using the learned reward function.

GAN-GCL assumes the distribution of expert trajectories as a Boltzmann distribution and approximates the energy function $f_\psi(\tau)$ with a neural network [61]. The density ratio of the trajectory distribution generated by the agent's policy$\pi(\tau)$ and the trajectory distribution of the expert $f_\psi(\tau)/Z$ is then used as a discriminator. Here Z is a partition function.

$$D_\psi(\tau) = \frac{exp(f_\psi(\tau))}{exp(f_\psi(\tau)) + \pi(\tau)} \tag{2}$$

In AIRL, pairs of states s and actions a instead of trajectoriesτ are used as inputs of the energy function. $f_\psi(s,a)$ is represented by functions, $r_\psi(s,a)$ and $V_\psi^{shaping}(s)$. $r_\psi(s,a)$ is a reward function while $V_\psi^{shaping}(s)$ is a value function called the shaping value function.

$$D_{\phi,\psi}(s_t,a_t,s_{t+1}) = \frac{exp(f_{\theta,\psi}(s_t,a_t,s_{t+1}))}{exp(f_{\theta,\psi}(s_t,a_t,s_{t+1})) + \pi(a_t|s_t)} \tag{3}$$

$$\begin{aligned}&f_{\theta,\psi}(s_t,a_t,s_{t+1}) = \\ &r_\psi(s_t,a_t) + \gamma V_\psi^{shaping}(s_{t+1}) - V_\psi^{shaping}(s_t)\end{aligned} \tag{4}$$

AIL, such as GAIL and AIRL, can be interpreted as bringing the occupancy measure$\rho_\pi(s,a) = \pi(a|s)\sum_t = 0^\infty \gamma^t P(s_t = s|\pi)$ closer to that of the expert on the criterion of Jensen–Shannon divergence like (5). Here, the occupancy measure represents the distributions of pairs of states and actions when rolling out a policy.

$$\text{argmin}_\theta \text{max}_\psi L(\theta,\psi) = \text{argmin}_\theta D_{JS}(\rho_{\pi_\theta}(s,a)||\rho_t(s,a)) \tag{5}$$

The optimal discriminator, $D_\psi^*(s)$ or $D_{\psi,\phi}^*(s,a,s')$, measures the Jensen–Shannon divergence of the occupancy measure, and the optimal policy minimizes the divergence.

IV. PROBLEM SETTING

Let's assume that trajectories of a source agent (SA) $D_{SA} = \{\{\left(s_t^{SA}, a_t^{SA}\right); t = 0,...,T-1\}_i; i = 0,...N-1\}$, those of the expert or source agent (source expert, SE) $D_{SE} = \{\{\left(s_t^{SE}, a_t^{SE}\right); t = 0,...,T-1\}_i; i = 0,...N-1\}$, and those of a target agent (TA) $D_{TA} = \{\{\left(s_t^{TA}, a_t^{TA}\right); t = 0,...,T-1\}_i; i = 0,...N-1\}$ can be obtained, where T is the last time step of trajectories and N is the number of trajectories. Additionally, the number of state dimensions of the source agent is not that of the target agent; i.e., $\text{dim}(s^{SA}) \neq \text{dim}(s^{TA})$. Let's assume that a true reward r^* takes a feature z_{task} that is effective for a task as the argument and depends on only states and not actions. For example, z_{task} is the velocity vector of the center of mass in the horizontal direction in locomotion tasks or relative position vector of the fingertip and the target object in reaching tasks. While in (5) the occupancy measures for the same dimensions are made closer by learning π_θ and $r_\psi(s)$ or D_ψ, there is the difference in the state dimensions in our problem such that we cannot get closer to the occupancy measure in the same space and we need not get closer to the occupancy measure of the dimension irrelevant to the task. Therefore, the source feature extractor $f_{\eta^S}(s^{task})$ that extracts z_{task} from s^{SA} or s^{SE} and the target feature extractor $f_{\eta^S}(s^{task})$ that extracts z_{task} from s^{TA} have to be learned. Then, in the space of z_{task}, the occupancy

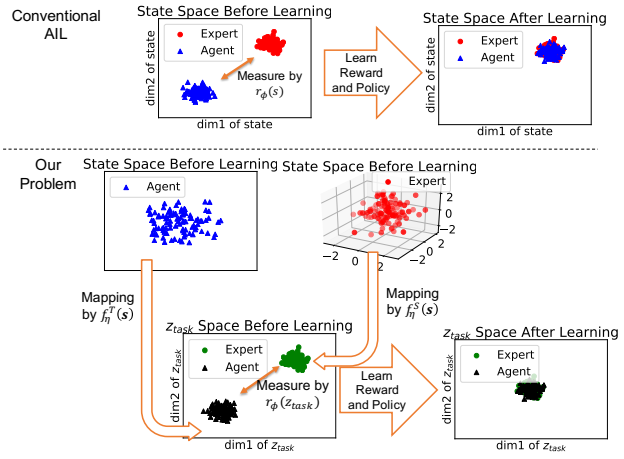

Fig. 2. The problem in conventional AIL is to match the occupancy measure in a state space between an agent and an expert by learning the reward function and agent's policy (above). Meanwhile, our problem is to match the occupancy measure in the space of features effective for a task by learning a reward function, agent's policy, and feature extractor.

measure has to be made closer between the source agent and the source expert by learning the reward function $r_\psi(s)$ and the source agent's policy π_{SA} and the target agent's policy π_{TA}. We can then change (5) like (7).

$$\text{argmin}_\theta D_{JS}(\rho_{\pi_\theta}(z^{SA}_{task})||\rho_t(z^{SE}_{task})) \tag{6}$$

$$\text{argmin}_\theta D_{JS}(\rho_{\pi_\theta}(f_{\eta^S}(s^{task}))||\rho_t(z^{SE}_{task})) \tag{7}$$

We present a conceptual comparison of the setting of our problem with the setting of the conventional AIL in Fig. 2.

V. METHOD

We learn the discriminator that discriminates between states of the source agent s^{SA} and states of the source expert s^{SE}. Additionally, π_{SA} is learned for the reward of $-log(1-D)$, and π_{TA} is learned for the reward of $-log(1-D)$ in the case of GAIL and r_ϕ in the case of AIRL.

Feature extractors can be learned here in two ways; i.e., by providing separate feature extractors for source and target agents and by learning the same feature extractor for source and target agents. The first approach can be realized adopting Gupta et al.'s method adopting an auto encoder [40] or canonical correlation analysis. When using the method of Gupta et al., the feature extractor is learned with a loss due to the addition of the squared difference between the feature quantity of the source agent and that of the target agent to the reconstruction loss. In the case of using CCA, the projection direction is learned so as to maximize the correlation between feature quantities z_{SA} and z_{TA} obtained via the linear projection of s_{SA} and s_{TA}. However, it is expected that for methods having separate feature extractors, because it is rare that the values of z_{task} are the same for pairs of data when trying to maximize the similarity, the information of z_{task} is removed as extracted features.

We therefore propose another method of sharing feature extractors between source and target agents. The proposed method comprises state shaping and unsupervised domain adaptation.

A. State Shaping

To input states with different dimension numbers to a common classifier, it is necessary to shape the state to match the number of dimensions.

We propose state shaping, which matches the number of state dimensions by inserting an arbitrary constant v into the missing state dimensions. When agent 1 has L joints, agent 2 has M joints, and the representation of the state is composed of s_{joint} dependent on the joint number and s_{common} not dependent on the joint number, state shaping transforms the state as follows.

if

$$s_1 = [s_{joint_1}, s_{joint_2}, \ldots, s_{joint_L}, s_{common}]$$

$$s_2 = [s_{joint_1}, s_{joint_2}, \ldots, s_{joint_M}, s_{common}],$$

where $L > M$,

then

$$s^{'}_2 = [s_{joint_1}, s_{joint_2}, \ldots, \underbrace{v, \ldots}_{\times(L-M)}, s_{joint_M}, s_{common}] \tag{8}$$

B. UDA

After aligning the number of dimensions by state shaping, we next propose using UDA to extract features independent of the number of state dimensions. We use MMD [62] and DANN [56] as representative UDA methods. In MMD, the feature extractor is trained so as to maximize similarity between the intermediate feature of neural networks of the source agent and neural networks of the target agent. The similarity is calculated using a kernel; we choose a Gaussian kernel with a width of 1. The loss when AIRL is adopted for imitation learning and DANN is used for domain adaptation is shown below.

$$\begin{aligned} &L_{AIRL+StateShaping+DANN}\,(\theta, \phi, \psi, \omega, \eta) \\ &= \mathbb{E}_{(s^{SA}_t, s^{SA}_{t+1})\sim D^{SA}}\left[log\left(1 - D_{\phi,\psi}(f_\eta(s^{SA}_t), f_\eta(s^{SA}_{t+1}))\right)\right] \\ &+ \mathbb{E}_{(s^{SE}_t, s^{SE}_{t+1})\sim D^{SE}}\left[logD_{\phi,\psi}(f_\eta(s^{SE}_t), f_\eta(s^{SE}_{t+1}))\right] \\ &+ \mathbb{E}_{s^{SA}\sim D^{SA}}\left[log\left(1 - D_\omega(R(f_\eta(s^{SA})))\right)\right] \\ &+ \mathbb{E}_{s^{TA}\sim D^{TA}}\left[logD_\omega(R(f_\eta(s^{TA})))\right] \end{aligned} \tag{9}$$

D_ω represents the domain classifier that classifies the states in terms of whether the state came from the source agent or target agent. R represents the gradient reversal layer (GRL) proposed in [56], which is the identity function in a forward pass (see (10)) but inverts the sign in a backward pass (see (11)).

$$R(x) = x \tag{10}$$

$$\frac{dR}{dx} = -I \tag{11}$$

Using the GRL, a gradient with the same absolute value but opposite sign to the domain classifier flows to the feature

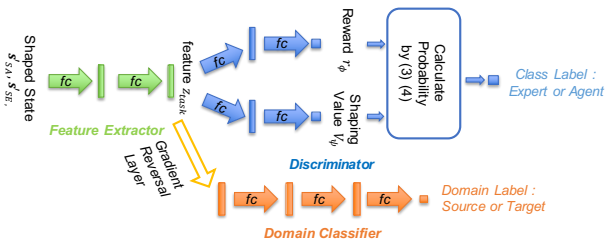

Fig. 3. Our neural network architecture. First, feature extractors extract features from states. The extracted features are used for inputs of both the discriminator, which discriminates whether the states come from the expert or agent, and the agent classifier, which classifies whether from source agent or target agent. Features input to the agent classifier pass through the GRL.

extractor. The feature extractor is therefore trained to extract a feature that is invariant between source and target agents while discriminating between the source agent and source expert. It is expected that the feature learned by this way can discriminate the target agent and target expert though the target expert does not actually exist. λ in (9) represents the weight about udaWe take a value of 1 for λ.

We present an overview of neural networks of the proposed method using AIRL and DANN in Fig. 3. Our neural networks for r_ϕ, $V_\psi^{shaping}$, and f_η have two fully connected layers with 32 output dimensions. All layers are followed by ReLU nonlinearities.

To produce source-expert trajectories D_{SE}, we train a policy using PPO for the policy update and generalized advantage estimation [63] for the reduction of variance. We then rollout the learned policy 100 times. D_{SE} therefore consists of 100 trajectories of the source expert. Whether we use GAIL or AIRL, we also train the policy of the source agent π_{theta}^{SA} and π_{theta}^{TA} by PPO and GAE, and we use 20 trajectories as on-policy data per update. We use a value of 0.995 as the discount factor in PPO and a value of 0.97 as the tradeoff value between bias and variance in GAE in all experiments.

VI. RESULTS

Experiments are conducted to answer three questions. (1) Can we train an agent to imitate an expert having a different shape? (2) How do methods of separating extracted features perform and does our method perform better? (3) Is our method robust against a value inserted in a state in state shaping, the dimension into which the value is inserted and the number of joints of agent and expert. We answer these questions by simulating a reaching task.

A. EXPERIMENTAL SETTING

We consider the Reacher-v1, 2 dimensional simulated reaching task in the MuJoCo physics engine [64]. We use the two-joint reacher as the target agent and three-joint reacher as the source agent and source expert. States of the source agent and source expert are expressed by (12), where x_f and x_e respectively denote the x coordinates of the fingertip and target object.

$$[cos\theta_1, cos\theta_2, cos\theta_3, sin\theta_1, sin\theta_2, sin\theta_3, \dot{\theta_1}, \dot{\theta_2}, \dot{\theta_3}, x_f, y_f, z_f, x_t, y_t, z_t] \tag{12}$$

The state of the target agent is expressed by (13).

$$[cos\theta_1, cos\theta_2, sin\theta_1, sin\theta_2, \dot{\theta_1}, \dot{\theta_2}, \quad x_f, y_f, z_f, x_t, y_t, z_t] \tag{13}$$

Then, in the case of inserting v into the dimension corresponding to the third joint in state shaping, we deform the state of the target agent into dimensions that correspond to joint 3 as in (14).

$$[cos\theta_1, cos\theta_2, v, sin\theta_1, sin\theta_2, v, \dot{\theta_1}, \dot{\theta_2}, v, x_f, y_f, z_f, x_t, y_t, z_t] \tag{14}$$

In the reacher simulation, a reward is defined using the norm of the two-dimensional relative position vector; i.e., $r = -||[x_f - x_t, y_f - y_t]||^2$. We evaluate the performance using the imitation ratio. The imitation ratio is calculated using (15), where R denotes the mean of cumulative reward of 20 episodes obtained by rolling out the learned policy. The learned policy is obtained by running each method until policy interacts with the environment in 7e7 time steps for methods using GAIL or 1.2e8 time steps for methods using AIRL.

$$\text{imitation ratio} = \frac{R_{method} - R_{random}}{R_{expert} - R_{random}} \tag{15}$$

B. Imitation from three joints to two joints

Table I presents the cumulative reward and success rate of the proposed and previous methods. The first and second rows respectively give the imitation ratios of the expert and random agent, which acts out random actions. We first focus on the third to seventh rows. The third to fifth rows give the results of methods with separate feature extractors for each agent while the fifth to seventh rows give the results of our method with one feature extractor shared between agents through state shaping. The imitation rates of methods with separate feature extractors are at most 55% while those of our method are at least 80%. It is thought that methods with separate feature extractors cannot maximize the similarity of a pair of data with the same z_{task} while a method that shares a feature extractor can use the same calculation for two agents. We next focus on the last four rows. UDA has little effect except for the method using AIRL for imitation learning and state shaping and DANN for the feature extractors (AIRL+StateShaping+DANN). Using DANN made the performance worse. It is thought that there is a difference in the distribution of z_{task} between the source agent and target agent, and DANN therefore removes the information of z_{task}.

Furthermore, we investigate why our methods with state shaping do not achieve an imitation ratio of 100%. We introduce the attention vector to visualize which dimensions the network of the feature extractor attaches importance to. We

TABLE I
CUMULATIVE REWARD AND SUCCESS RATE OF THE PROPOSED AND PREVIOUS METHODS

Method	Cumulative Reward	Imitation Rate (%)
Expert	-4.29	100
Random Agent	-43.8	0
GAIL+CCA	-29.20	37.0
AIRL+CCA	-22.46	54.0
AE [Gupta+]	-22.91	52.9
GAIL+StateShaping	-11.8	81.0
AIRL+StateShaping	-11.07	82.8
GAIL+StateShaping+DANN	-12.3	80.0
GAIL+StateShaping+MMD	-8.57	89.1
AIRL+StateShaping+DANN	-40.7	7.85
AIRL+StateShaping+MMD	-10.20	85.0

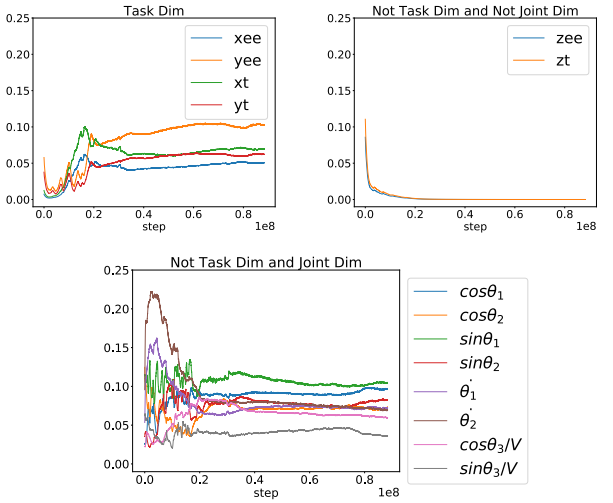

Fig. 4. Graphs showing how the values of the attention vector change as the feature extractor is learned.

define the attention vector av as below, where b denotes the learnable parameter of the score of a dimensions importance.

$$b = [b_1, \ldots, b_L] \tag{16}$$

$$av = softmax(b) = [\frac{b_1}{\sum_{k=1,\ldots,L} exp(b_k)}, \ldots, \frac{b_1}{\sum_{k=1,\ldots,L} exp(b_k)}] \tag{17}$$

We calculate the Hadamard product of the attention vector and states and then input the product to the feature extractor. We show how the value of the attention vector changes in Fig. 4. The attention given to dimensions important to the task, such as x_e, y_e, x_t, and y_t, increases as the feature extractor is learned while that given to dimensions not important to the task and not dependent on joints, such as z_e and z_t, decreases. However, the attention given to dimensions that are not important and dependent does not decrease, and it is thought that the imitation ratio does not reach 100% because the information of this class of dimensions remains in the extracted feature.

TABLE II
ROBUSTNESS AGAINST THE DIMENSION IN WHICH A VALUE IS INSERTED.

Dimension inserted into	Cumulative Reward	Imitation Rate (%)
Joint 1	-11.07	82.8
Joint 2	-12.9	78.2
Joint 3	-12.8	78.5
Random	-13.3	77.2

TABLE III
ROBUSTNESS AGAINST THE VALUE INSERTED.

Value inserted	Cumulative Reward	Imitation Rate (%)
0	-11.07	82.8
Random	-12.1	80.2

C. Robustness of State Shaping

We investigate the robustness of state shaping against the dimension into which the value is inserted into, the value inserted, the number of experts, and the number of joints of the source expert and target agent. First, Table II presents the robustness of state shaping against the place the value is inserted into. Joints 1, 2, and 3 respectively refer to the value v being inserted into the dimensions relating to θ_1, θ_2, and θ_3. The term "Random" refers to the value v being inserted into a dimension relating to a randomly chosen joint. Results show that our method using state shaping is robust against the dimension in which the value is inserted.

Second, we show the robustness against the value inserted in Table III. A value of "0" means that v is zero. The terms "Fix Random" and Joint 3 respectively refer to the value v being inserted into the dimensions relating to θ_1, θ_2, and θ_3. The term "Random" refers to the value v being sampled from a uniform distribution whose maximum value is 1 and minimum value is zero. Results show that our method using state shaping is robust against the value inserted.

Finally, we show the robustness against the numbers of joints of the source expert and target agent in Table IV. Similar imitation ratios are obtained except when there are four source experts and two target agents, for which the imitation ratio is -2.22. The state shaping is thus largely robust against the number of joints.

TABLE IV
ROBUSTNESS AGAINST THE NUMBERS OF JOINTS OF THE SOURCE EXPERT OR AGENT AND TARGET AGENT.

Num. of joints of expert	Num. of joints of agent	Imitation Rate (%)
3	2	82.8
4	2	-2.22
4	3	91.2
5	2	87.1
5	4	92.3

VII. CONCLUSION

We defined the problem of AIL for agents with different numbers of dimensions. We then proposed a method of solving the problem by sharing the feature extractor between the

source expert or agent and the target agent. We additionally proposed a method of using UDA to reduce the effect of the difference in the number of state dimensions. We showed that our method has an imitation rate that is more than 25% superior to the rate of methods with separate feature extractors. We also showed that domain adaptation has no effect on performance or can even worsen the performance. We introduced the attention vector for the visualization of which state dimensions the feature extractor attaches importance to. Analysis with the attention vector revealed that the method with state shaping fails to remove information not important to the task and dependent on the joint. We also showed that state shaping is robust against the value to be inserted into the state, the dimension in which the value is inserted, and the numbers of joints of the source expert or agent and target agent.

An imitation ratio of 100% was not achieved in the present study. Improvements in performance will thus be sought in future work. To this end, data augmentation of the source expert is thought to be effective because it makes it easy to remove unnecessary information from the feature; e.g., it may be effective to source expert's trajectories solving the same task for different state transition probabilities. Having achieved a higher imitation ratio, it will be important to consider different tasks and environments as our work focused only on the reaching task.

ACKNOWLEDGMENT

We thank Kazutoshi Tanaka and Izumi Karino for valuable comments on experiments and methods.

REFERENCES

[1] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot *et al.*, "Mastering the game of go with deep neural networks and tree search," *nature*, vol. 529, no. 7587, pp. 484–489, 2016.

[2] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," *arXiv preprint arXiv:1312.5602*, 2013.

[3] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.

[4] Jack Clark and Dario Amodei, "Faulty Reward Functions in the Wild," https://blog.openai.com/faulty-reward-functions/.

[5] A. Rajeswaran, V. Kumar, A. Gupta, G. Vezzani, J. Schulman, E. Todorov, and S. Levine, "Learning Complex Dexterous Manipulation with Deep Reinforcement Learning and Demonstrations," in *Proceedings of Robotics: Science and Systems (RSS)*, Pittsburgh, Pennsylvania, June 2018.

[6] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, p. 529, 2015.

[7] X. B. Peng, G. Berseth, K. Yin, and M. van de Panne, "Deeploco: Dynamic locomotion skills using hierarchical deep reinforcement learning," *ACM Transactions on Graphics (Proc. SIGGRAPH 2017)*, vol. 36, no. 4, 2017.

[8] J. Hwangbo, J. Lee, A. Dosovitskiy, D. Bellicoso, V. Tsounis, V. Koltun, and M. Hutter, "Learning agile and dynamic motor skills for legged robots," *Science Robotics*, vol. 4, no. 26, p. eaau5872, 2019.

[9] X. B. Peng, P. Abbeel, S. Levine, and M. van de Panne, "Deepmimic: Example-guided deep reinforcement learning of physics-based character skills," *ACM Transactions on Graphics (TOG)*, vol. 37, no. 4, p. 143, 2018.

[10] A. Y. Ng, S. J. Russell *et al.*, "Algorithms for inverse reinforcement learning."

[11] P. Abbeel and A. Y. Ng, "Apprenticeship learning via inverse reinforcement learning," in *Proceedings of the Twenty-first International Conference on Machine Learning*, ser. ICML '04. New York, NY, USA: ACM, 2004, pp. 1–. [Online]. Available: http://doi.acm.org/10.1145/1015330.1015430

[12] S. Russell, "Learning agents for uncertain environments," in *Proceedings of the eleventh annual conference on Computational learning theory*. ACM, 1998, pp. 101–103.

[13] B. D. Ziebart, A. L. Maas, J. A. Bagnell, and A. K. Dey, "Maximum entropy inverse reinforcement learning." 2008.

[14] C. Finn, S. Levine, and P. Abbeel, "Guided cost learning: Deep inverse optimal control via policy optimization," in *International Conference on Machine Learning*, 2016, pp. 49–58.

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

[16] J. Fu, K. Luo, and S. Levine, "Learning robust rewards with adversarial inverse reinforcement learning," *arXiv preprint arXiv:1710.11248*, 2017.

[17] A. Meltzoff, "Born to learn: What infants learn from watching us," *Role Early Exp. Infant Dev.*, 01 1999.

[18] I. Popov, N. Heess, T. Lillicrap, R. Hafner, G. Barth-Maron, M. Vecerik, T. Lampe, Y. Tassa, T. Erez, and M. Riedmiller, "Data-efficient deep reinforcement learning for dexterous manipulation," *arXiv preprint arXiv:1704.03073*, 2017.

[19] S. Levine and V. Koltun, "Guided policy search," in *International Conference on Machine Learning*, 2013, pp. 1–9.

[20] S. Levine, N. Wagener, and P. Abbeel, "Learning contact-rich manipulation skills with guided policy search," in *2015 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2015, pp. 156–163.

[21] M. Deisenroth and C. E. Rasmussen, "Pilco: A model-based and data-efficient approach to policy search," in *Proceedings of the 28th International Conference on machine learning (ICML-11)*, 2011, pp. 465–472.

[22] M. P. Deisenroth, "Learning to control a low-cost manipulator using data-efficient reinforcement learning."

[23] N. R. Ke, A. Singh, A. Touati, A. Goyal, Y. Bengio, D. Parikh, and D. Batra, "Modeling the long term future in model-based reinforcement learning," in *International Conference on Learning Representations*, 2019. [Online]. Available: https://openreview.net/forum?id=SkgQBn0cF7

[24] L. Kaiser, M. Babaeizadeh, P. Milos, B. Osinski, R. H. Campbell, K. Czechowski, D. Erhan, C. Finn, P. Kozakowski, S. Levine, R. Sepassi, G. Tucker, and H. Michalewski, "Model-based reinforcement learning for atari," 2019.

[25] R. S. Sutton, "Dyna, an integrated architecture for learning, planning, and reacting," *ACM SIGART Bulletin*, vol. 2, no. 4, pp. 160–163, 1991.

[26] S. Chiappa, S. Racaniere, D. Wierstra, and S. Mohamed, "Recurrent environment simulators," *arXiv preprint arXiv:1704.02254*, 2017.

[27] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in neural information processing systems*, 2014, pp. 2672–2680.

[28] Y. Kuniyoshi, M. Inaba, and H. Inoue, "Learning by watching: Extracting reusable task knowledge from visual observation of human performance," *IEEE transactions on robotics and automation*, vol. 10, no. 6, pp. 799–822, 1994.

[29] H. Miyamoto, S. Schaal, F. Gandolfo, H. Gomi, Y. Koike, R. Osu, E. Nakano, Y. Wada, and M. Kawato, "A kendama learning robot based on bi-directional theory," *Neural networks*, vol. 9, no. 8, pp. 1281–1302, 1996.

[30] J. Demiris, S. Rougeaux, G. M. Hayes, L. Berthouze, and Y. Kuniyoshi, "Deferred imitation of human head movements by an active stereo vision head," in *Proceedings 6th IEEE International Workshop on Robot and Human Communication. RO-MAN'97 SENDAI*, Sep. 1997, pp. 88–93.

[31] C. G. Atkeson and S. Schaal, "Learning tasks from a single demonstration," in *Proceedings of International Conference on Robotics and Automation*, vol. 2. IEEE, 1997, pp. 1706–1712.

[32] C. Finn, T. Yu, T. Zhang, P. Abbeel, and S. Levine, "One-shot visual imitation learning via meta-learning," *arXiv preprint arXiv:1709.04905*, 2017.

[33] Y. Duan, M. Andrychowicz, B. Stadie, O. Jonathan Ho, J. Schneider, I. Sutskever, P. Abbeel, and W. Zaremba, "One-shot imitation learning," in *Advances in Neural Information Processing Systems 30*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds. Curran Associates, Inc., 2017, pp. 1087–1098. [Online]. Available: http://papers.nips.cc/paper/6709-one-shot-imitation-learning.pdf

[34] J. Zhang and K. Cho, "Query-efficient imitation learning for end-to-end autonomous driving," *arXiv preprint arXiv:1605.06450*, 2016.

[35] M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang *et al.*, "End to end learning for self-driving cars," *arXiv preprint arXiv:1604.07316*, 2016.

[36] A. Giusti, J. Guzzi, D. C. Cireşan, F.-L. He, J. P. Rodríguez, F. Fontana, M. Faessler, C. Forster, J. Schmidhuber, G. Di Caro *et al.*, "A machine learning approach to visual perception of forest trails for mobile robots," *IEEE Robotics and Automation Letters*, vol. 1, no. 2, pp. 661–667, 2016.

[37] D. A. Pomerleau, "Alvinn: An autonomous land vehicle in a neural network," in *Advances in neural information processing systems*, 1989, pp. 305–313.

[38] T. Yu, C. Finn, A. Xie, S. Dasari, T. Zhang, P. Abbeel, and S. Levine, "One-shot imitation from observing humans via domain-adaptive meta-learning," *arXiv preprint arXiv:1802.01557*, 2018.

[39] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," *arXiv preprint arXiv:1703.03400*, 2017.

[40] A. Gupta, C. Devin, Y. Liu, P. Abbeel, and S. Levine, "Learning invariant feature spaces to transfer skills with reinforcement learning," *arXiv preprint arXiv:1703.02949*, 2017.

[41] P. Baldi, "Autoencoders, unsupervised learning, and deep architectures," in *Proceedings of ICML workshop on unsupervised and transfer learning*, 2012, pp. 37–49.

[42] O. M. Andrychowicz, B. Baker, M. Chociej, R. Józefowicz, B. McGrew, J. W. Pachocki, A. Petron, M. Plappert, G. Powell, A. Ray, J. Schneider, S. Sidor, J. Tobin, P. Welinder, L. Weng, and W. Zaremba, "Learning dexterous in-hand manipulation," 2018.

[43] J. Tan, T. Zhang, E. Coumans, A. Iscen, Y. Bai, D. Hafner, S. Bohez, and V. Vanhoucke, "Sim-to-real: Learning agile locomotion for quadruped robots," *arXiv preprint arXiv:1804.10332*, 2018.

[44] X. B. Peng, M. Andrychowicz, W. Zaremba, and P. Abbeel, "Sim-to-real transfer of robotic control with dynamics randomization," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, May 2018, pp. 1–8.

[45] J. Matas, S. James, and A. J. Davison, "Sim-to-real reinforcement learning for deformable object manipulation," *arXiv preprint arXiv:1806.07851*, 2018.

[46] F. Muratore, F. Treede, M. Gienger, and J. Peters, "Domain randomization for simulation-based policy optimization with transferability assessment," in *Conference on Robot Learning*, 2018, pp. 700–713.

[47] S. James, P. Wohlhart, M. Kalakrishnan, D. Kalashnikov, A. Irpan, J. Ibarz, S. Levine, R. Hadsell, and K. Bousmalis, "Sim-to-real via sim-to-sim: Data-efficient robotic grasping via randomized-to-canonical adaptation networks," 2018.

[48] X. Glorot, A. Bordes, and Y. Bengio, "Domain adaptation for large-scale sentiment classification: A deep learning approach," in *Proceedings of the 28th international conference on machine learning (ICML-11)*, 2011, pp. 513–520.

[49] M. Ghifary, W. B. Kleijn, and M. Zhang, "Domain adaptive neural networks for object recognition," in *Pacific Rim international conference on artificial intelligence*. Springer, 2014, pp. 898–904.

[50] B. Gong, K. Grauman, and F. Sha, "Connecting the dots with landmarks: Discriminatively learning domain-invariant features for unsupervised domain adaptation," in *International Conference on Machine Learning*, 2013, pp. 222–230.

[51] S. J. Pan, I. W. Tsang, J. T. Kwok, and Q. Yang, "Domain adaptation via transfer component analysis," *IEEE Transactions on Neural Networks*, vol. 22, no. 2, pp. 199–210, 2011.

[52] M. Long, J. Wang, G. Ding, J. Sun, and P. S. Yu, "Transfer feature learning with joint distribution adaptation," in *Proceedings of the IEEE international conference on computer vision*, 2013, pp. 2200–2207.

[53] K. Bousmalis, A. Irpan, P. Wohlhart, Y. Bai, M. Kelcey, M. Kalakrishnan, L. Downs, J. Ibarz, P. Pastor, K. Konolige *et al.*, "Using simulation and domain adaptation to improve efficiency of deep robotic grasping," *arXiv preprint arXiv:1709.07857*, 2017.

[54] M. Baktashmotlagh, M. T. Harandi, B. C. Lovell, and M. Salzmann, "Unsupervised domain adaptation by domain invariant projection," in *Proceedings of the IEEE International Conference on Computer Vision*, 2013, pp. 769–776.

[55] M. Long, Y. Cao, J. Wang, and M. I. Jordan, "Learning transferable features with deep adaptation networks," *arXiv preprint arXiv:1502.02791*, 2015.

[56] Y. Ganin, E. Ustinova, H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, M. Marchand, and V. Lempitsky, "Domain-adversarial training of neural networks," *The Journal of Machine Learning Research*, vol. 17, no. 1, pp. 2096–2030, 2016.

[57] P. Haeusser, T. Frerix, A. Mordvintsev, and D. Cremers, "Associative domain adaptation."

[58] J. Hoffman, E. Tzeng, T. Park, J.-Y. Zhu, P. Isola, K. Saenko, A. A. Efros, and T. Darrell, "Cycada: Cycle-consistent adversarial domain adaptation," *arXiv preprint arXiv:1711.03213*, 2017.

[59] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, "Trust region policy optimization," in *International Conference on Machine Learning*, 2015, pp. 1889–1897.

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

[61] C. Finn, P. Christiano, P. Abbeel, and S. Levine, "A connection between generative adversarial networks, inverse reinforcement learning, and energy-based models," *arXiv preprint arXiv:1611.03852*, 2016.

[62] A. Gretton, K. M. Borgwardt, M. J. Rasch, B. Schölkopf, and A. Smola, "A kernel two-sample test," *Journal of Machine Learning Research*, vol. 13, no. Mar, pp. 723–773, 2012.

[63] J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel, "High-dimensional continuous control using generalized advantage estimation," *arXiv preprint arXiv:1506.02438*, 2015.

[64] E. Todorov, T. Erez, and Y. Tassa, "Mujoco: A physics engine for model-based control," in *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*. IEEE, 2012, pp. 5026–5033.