Learning from Imperfect Demonstrations via Adversarial Confidence Transfer

Zhangjie Cao*[1], Zihan Wang*[2], Dorsa Sadigh[1,2]

***Abstract*— Existing learning from demonstration algorithms usually assume access to expert demonstrations. However, this assumption is limiting in many real-world applications since the collected demonstrations may be suboptimal or even consist of failure cases. We therefore study the problem of learning from imperfect demonstrations by learning a confidence predictor. Specifically, we rely on demonstrations along with their confidence values from a different *correspondent* environment (source environment) to learn a confidence predictor for the environment we aim to learn a policy in (target environment—*where we only have unlabeled demonstrations.*) We learn a common latent space through adversarial distribution matching of multi-length partial trajectories to enable the transfer of confidence across source and target environments. The learned confidence reweights the demonstrations to enable learning more from informative demonstrations and discarding the irrelevant ones. Our experiments in three simulated environments and a real robot reaching task demonstrate that our approach learns a policy with the highest expected return. We show the videos of the real robot arm experiments on our website.**

I. Introduction

Standard imitation learning algorithms assume that collected demonstrations are provided by experts, who can always achieve successful outcomes [1], [2], [3]. However, this assumption is usually violated in real-world applications. Expert demonstrations require domain expertise and thus are expensive to obtain. On the other hand, we often have easy access to a plethora of imperfect demonstrations ranging from expert behavior to noise. This is evident when collecting human demonstrations on a robot arm since controlling manipulators with high degrees of freedom can be challenging for human users [4], or a bounded rational demonstrator may have difficulty providing optimal demonstrations [5].

Prior work on learning from imperfect demonstrations can be roughly categorized into confidence-based and ranking-based methods: Confidence-based methods learn a confidence over demonstrations to re-weight them [6], while ranking-based methods recover the reward function [7], [8], [9]. However, all these works require some level of annotation, e.g., confidence or rankings of the demonstrations. For complex environments, large-scale annotations are required to cover the whole state-action space, which is labor-intensive and expensive to obtain. Furthermore, the annotations can often contain a significant amount of noise when collected through crowdsourcing platforms.

Emails: caozj@cs.stanford.edu, wangzih@stanford.edu, dorsa@cs.stanford.edu

[1]Computer Science, Stanford University, CA, USA

[2]Electrical Engineering, Stanford University, CA, USA

* means Equal Contribution. Author ordering determined by coin flip over a Google Hangout.

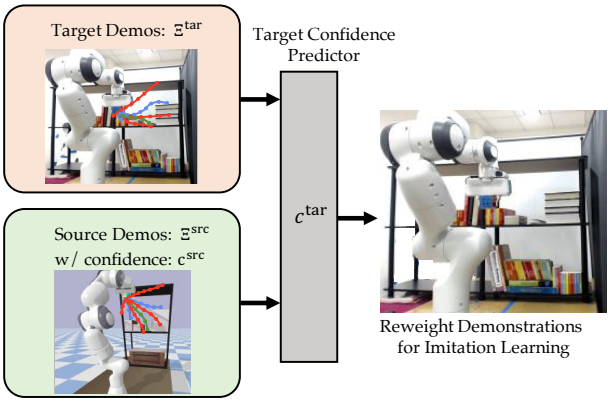


Fig. 1: Overview of the problem setting and our proposed target confidence predictor.

In this paper, we focus on confidence-based methods. As shown in Fig. 1, our insight is that instead of using manually annotated confidence over demonstrations, we leverage confidence annotations in a different but related environment—the *source* environment. The source environment may have a different state-action space from the *target* environment—the environment we are operating in. The source environment and the target environment should have a correspondence mapping between state-action pairs [10]. We assume that collecting offline demonstrations along with their confidence annotations is easy in the source environment. As an example, the source and target environments can be a simulated robot and a real robot respectively or two different robot arms with different degrees of freedom. Our key idea is to leverage the correspondence between state-action pairs in the source and target environments to learn a confidence measure without the restrictive assumption of access to confidence annotations of target demonstrations.

This problem setting introduces a new challenge: The source and target environments have different state-action spaces. That means the dimensions of the states and actions and even the semantic meaning of each dimension can be different between the two environments, so we cannot directly apply the source confidence predictor to the target state-action pairs. We instead need to transfer the source confidence predictor to the target environment.

We propose an approach that leverages the assumption of the correspondence between state-action pairs in the source and target environments leading to the existence of a common latent space, where one can learn a shared confidence predictor for both environments. Our approach aligns both the feature-level distributions and the confidence-level distributions of

partial trajectories with different lengths across domains by a domain adversarial loss. We design a multi-length partial trajectory matching, which preserves temporal relationships across consecutive states and actions, and enables an accurate matching with the corresponding trajectory in the target environment.

Our contribution is a new confidence-based imitation learning algorithm that learns from imperfect demonstrations. Instead of relying on manual annotations in our environment, we leverage confidence-annotated demonstrations in a related environment—where collecting this data is easier—to learn a confidence predictor for our environment of interest. We experiment with several simulation environments and a real robot reaching task. Our results suggest that our approach achieves both more accurate confidence prediction and higher performance of the imitation policy than other methods.

II. Related Work

Standard Imitation Learning. Imitation learning aims to learn a policy by imitating the behavior of expert demonstrations, where mainstream methods can be categorized into behavior cloning (BC), inverse reinforcement learning (IRL), and generative adversarial imitation learning (GAIL). BC directly learns the policy mapping state to action by supervised learning over state-action pair data in the demonstrations [11], [12], [13]. IRL first recovers the reward function from demonstrations and learns a policy using reinforcement learning based on the learned reward function [14], [1], [2]. GAIL aligns the occupancy measure between the policy and the demonstrations with adversarial learning [3], [15], [16]. These standard imitation learning methods usually assume that all of the demonstrations are from experts who successfully finish the task. Our problem goes beyond this assumption by allowing for learning from potentially suboptimal or even malicious demonstrations.

Imitation Learning from Imperfect Demonstrations. Imitation learning from imperfect demonstrations aims to recover a well-performing policy from demonstrations that can range from random noise to optimal. Ranking-based and confidence-based methods are the two core approaches to address this problem. Ranking-based methods recover the reward function from provided ground-truth [7], [17], self-generated [8], [9], or interactively collected rankings [18]. In our setting, we can only generate rankings and learn the reward in the source environment, which cannot be applied to the target environment with a different state space. Target rankings can be self-generated by injecting noise in the target demonstrations as in [8], but that would require the self-generated demonstrations to be ranked worse than the original ones. This may not hold in our setting with the potential of nearly random demonstrations in our dataset.

Confidence-based methods typically learn the confidence predictor from confidence annotations [6] but the annotations are assumed to be on target demonstrations. We go beyond this assumption in our setting, since the confidence predictors are from a different environment. Cao et al. learn the target confidence predictor from the reward of demonstrations in the source environment but still assume the two environments share the same state space [19]. We target a more general setting, where the source and target environments have different state-action spaces and dynamics.

Correspondence of Dynamics. To learn the correspondence [20] between different dynamics, there are mainly two categories of methods: learning a translation mapping [21], or a common latent space [22] for states and actions. To learn a translation mapping, Tzeng et al. [23] weakly align pairs of images with paired image observations. Ammar el al. [24] utilize unsupervised manifold alignment to find correspondence between states of demonstrations. Kim et al. [25] and Zhang et al. [10] recently translated states and actions across environments by learning a unidirectional mapping of states and a cycle of forward mapping and inverse mapping between actions. Learning a translation mapping is often much more challenging than simply finding a common latent space. A common latent space is sufficient for the transfer of a confidence predictor across different dynamics.

To learn a common latent space, domain randomization methods align the different dynamics by randomizing the dynamics with hand-crafted augmentations [26], [27], [28], but they require the differences between the two dynamics to be covered by the augmentations. Gupta et al. [29] require pairs of states from different dynamics to learn the latent space. The assumptions of paired data or covering augmentations are often restrictive in practice. We remove these assumptions and only assume the existence of correspondence and learn the invariant representation space using multi-length partial trajectory matching.

III. Problem Statement

Let the source and target environments be represented by two deterministic Markov decision processes (MDP), M^{src} and M^{tar}, where $\mathcal{M}^\bullet = \langle \mathcal{S}^\bullet, \mathcal{A}^\bullet, p^\bullet, \mathcal{R}^\bullet, \rho_0^\bullet, \gamma^\bullet \rangle, \bullet \in \{\text{src}, \text{tar}\}$. $\bullet = \text{src}$ indicates the source domain, while $\bullet = \text{tar}$ indicates the target domain. $\mathcal{S}^\bullet$ is the state space, $\mathcal{A}^\bullet$ is the action space, $p^\bullet : \mathcal{S}^\bullet \times \mathcal{A}^\bullet \times \to \mathcal{S}^\bullet$ is the transition function, $\rho_0^\bullet$ is the initial state distribution, $\mathcal{R}^\bullet : \mathcal{S}^\bullet \times \mathcal{A}^\bullet \to \mathbb{R}$ is the reward function, and $\gamma^\bullet$ is the discount factor. A policy for M^{tar} is $\pi^{\text{tar}} : \mathcal{S}^{\text{tar}} \times \mathcal{A}^{\text{tar}} \to [0, 1]$ defining a probability distribution over the target action space in a given state. To evaluate a policy π^{tar}, we use the expected return, which is defined as $\eta_{\pi^{\text{tar}}} = \mathbb{E}_{s_0 \sim \rho_0^{\text{tar}}, \pi^{\text{tar}}} [\sum_{t=0}^{\infty} (\gamma^{\text{tar}})^t \mathcal{R}^{\text{tar}}(s_t, a_t, s_{t+1})]$, where t indicates the time step. A trajectory $\xi^\bullet$ is drawn from either the source or target environments, and is a sequence of state-action pairs: $\xi^\bullet = \{s_0^\bullet, a_0^\bullet, s, \dots, a_{T-1}^\bullet, s_T^\bullet\}$.

Problem Definition. In our problem setting, we are given a set of imperfect target demonstrations $\Xi^{\text{tar}} = \{\xi_1^{\text{tar}}, \xi_2^{\text{tar}}, \dots\}$ in the target environment M^{tar}, and a set of imperfect source demonstrations $\Xi^{\text{src}} = \{\xi_1^{\text{src}}, \xi_2^{\text{src}}, \dots\}$ along with a measure of confidence c^{src} in the source environment $\mathcal{M}^{\text{src}}$, where the confidence function $c^{\text{src}} : \mathcal{S}^{\text{src}} \times \mathcal{A}^{\text{src}} \to \mathbb{R}$ evaluates how useful the input state-action pairs are and if they are available or easy to access. Our goal is to find a well-performing policy π^{tar} for the target environment. Note that we go beyond standard imitation learning by allowing these demonstrations

in both environments to range from useless or even harmful demonstrations to nearly optimal or optimal ones. At the high level, our approach is to learn a confidence function $c^{\text{tar}} : \mathcal{S}^{\text{tar}} \times \mathcal{A}^{\text{tar}} \to \mathbb{R}$, which assigns a value to how confident we are in the optimality of the action a_t^{tar} in a particular state s_t^{tar}. We can then reweight the target state-action pairs and conduct standard imitation learning on the reweighted state-action pairs to learn a policy. To learn c^{tar}, we would like to learn a confidence predictor f^{tar} for the target environment by leveraging knowledge of the confidence function c^{src} from the source environment.

Assumptions. To transfer the confidence predictor from the source to the target environment, we assume that the source and target environments are *correspondent*, which means that there exists a relational mapping between the state-action pairs of the source agent and the target agent [10]. Specifically, there exists a relation on states: $\Phi : \mathcal{S}^{\text{src}} \times \mathcal{S}^{\text{tar}}$. There also exists a mapping from source state-action pairs to the target action space, $H_1 : \mathcal{S}^{\text{src}} \times \mathcal{A}^{\text{src}} \to \mathcal{A}^{\text{tar}}$, and another mapping from target state-action pairs to the source action space, $H_2 : \mathcal{S}^{\text{tar}} \times \mathcal{A}^{\text{tar}} \to \mathcal{A}^{\text{src}}$. The mappings ϕ, H_1, and H_2 satisfy the following: if $(s^{\text{src}}, s^{\text{tar}}) \in \Phi$, then $\forall a^{\text{src}} \in \mathcal{A}^{\text{src}}, (s^{\text{src}}_{\text{next}}, s^{\text{tar}}_{H_1}) \in \Phi$ and $\forall a^{\text{tar}} \in \mathcal{A}^{\text{tar}}, (s^{\text{tar}}_{\text{next}}, s^{\text{src}}_{H_2}) \in \Phi$. Here, $s^{\text{src}}_{\text{next}}$ is the successor state from s^{src} when taking the action a^{src}, and $s^{\text{tar}}_{H_1}$ is the successor state from state s^{tar} when taking the action $H_1(s^{\text{src}}, a^{\text{src}})$, and $s^{\text{tar}}_{\text{next}}$ and $s^{\text{src}}_{H_2}$ are defined similarly. The assumption builds a connection between the source and target agents. Without this assumption, the two environments may have no relations to each other, and there will not be any reason for knowledge transfer across them. This assumption is often satisfied in many real-world applications. For instance, we are interested in two particular settings: (1) The source environment being a simulation of a real robot that would define the target environment. For example, the source environment can be a simulated robotic arm picking an apple from a bowl in simulation, while the target environment can be a real robotic arm picking an apple from a bowl. (2) The source and target environments are different robots performing the same task in a similar context. For example, the source environment can be a simpler 3-DoF robot arm reaching the center of a bowl without colliding with obstacles, while the target robot can be a 7-DoF robot performing the same task. We note that the correspondence assumption is less restrictive compared to assumptions in related prior work. For instance, correspondence is implied, when assuming the same dynamics as in [6], [8].

IV. Adversarial Confidence Transfer

We aim to leverage the confidence annotation in the source environment and transfer the confidence knowledge to the target environment. Our key insight is to map the source and target state-action pairs to a common latent space $\mathcal{Z}$. We enforce the latent features for correspondent state-action pairs in the source and target environments, z^{src} and z^{tar}, to be the same. Here, $(s^{\text{src}}, a^{\text{src}})$ and $(s^{\text{tar}}, a^{\text{tar}})$ satisfy $(s^{\text{src}}, s^{\text{tar}}) \in \Phi$ and $(s^{\text{src}}_{\text{next}}, s^{\text{tar}}_{\text{next}}) \in \Phi$. We then build a shared decoder to predict the confidence from the latent features.

Since the source and target environments have different state-action spaces, we learn two different encoders $E^{\text{src}} : \mathcal{S}^{\text{src}} \times \mathcal{A}^{\text{src}} \to \mathcal{Z}$ and $E^{\text{tar}} : \mathcal{S}^{\text{tar}} \times \mathcal{A}^{\text{tar}} \to \mathcal{Z}$ to map the source and target state-action pairs into a common latent space $\mathcal{Z}$. We then learn a shared decoder $F : \mathcal{Z} \to \mathbb{R}$ to predict the confidence from latent features with confidence-labeled demonstrations in the source environment. The target confidence predictor can be computed by $F \circ E^{\text{tar}}$. Now, the challenge is how to learn the encoders to ensure that the common latent space satisfies the above requirements.

A. Multi-length Partial Trajectory Matching

To map correspondent state-action pairs to a similar latent feature, we develop a distribution matching objective to align the latent distribution of source and target state-action pairs. We can directly align the distribution of state-action latent features, i.e., matching $E^{\text{src}}(s_t^{\text{src}}, a_t^{\text{src}})$ and $E^{\text{tar}}(s_t^{\text{tar}}, a_t^{\text{tar}})$. However, this does not capture the temporal dependency between state-action pairs over a trajectory. To address this issue, we incorporate a prior that captures this temporal dependency when learning the encoders. Imagine a setting where a simulated robot and a real robot are both making a burger. The state-action pairs corresponding to placing the ingredients (e.g., the patty, lettuce, and tomatoes) are always after placing the bottom bun. If we only match the state-action pairs one at a time between the simulated and the real robot, placing the bottom bun may match placing any of the ingredients or even the top bun, but if we also match the consecutive steps of the state-action pairs, placing the bottom bun in both environments will more likely capture the information that the bottom bun comes before the rest of the ingredients. So, as shown in Fig. 2 (right), we propose a multi-length partial trajectory alignment that emphasizes learning the temporal relationship between state-action pairs. Specifically, we match the latent feature distribution of length k, $(k = 1, 2, \ldots, K)$ partial trajectories, which preserves the temporal relationship between consecutive states and actions, and make the latent features of correspondent state-action pairs more likely to be aligned.

B. Feature-Level and Confidence-Level Matching

We develop our learning algorithm that aligns the latent distribution of state-action pairs. We first train the source encoder E^{src} and the shared decoder F with source confidence-labeled state-action pairs using a regression loss:

$$\mathcal{L}_{\text{reg}} = \mathbb{E}_{(s^{\text{src}}, a^{\text{src}}) \sim \xi^{\text{src}}, \xi^{\text{src}} \sim \Xi^{\text{src}}} \left[\| F(E^{\text{src}}(s^{\text{src}}, a^{\text{src}})) - c^{\text{src}}(s^{\text{src}}, a^{\text{src}}) \| \right]. \tag{1}$$

After learning E^{src} and F, we fix the parameters of E^{src}, and F and use the latent feature space of E^{src} as the common feature space. We then only need to make the target encoder E^{tar} encode the target state-action pairs into this shared feature space and match the source latent feature distribution.

We learn the latent feature distribution alignment using a generative adversarial network. Since we only want the partial trajectories of the same length to be matched, for each length of partial trajectories k, we adopt a discriminator D_k and a loss $\mathcal{L}^k_{\text{fea}}$ to match the latent distribution of length-k

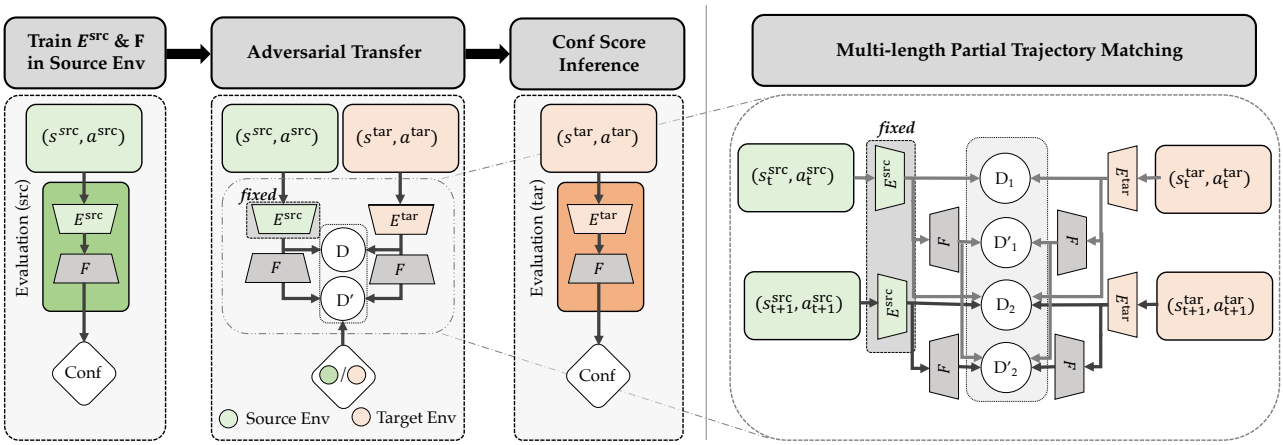


Fig. 2: (left) The overview of the learning process consisting of three stages: In the first stage, we train the source encoder E^{src} and the shared decoder with source confidence-labeled demonstrations $s^{\text{src}}, a^{\text{src}}$. In the second stage, we align the latent distributions of source and target state-action pairs with a multi-length partial trajectory matching, where we explain the second stage in detail in the right figure. In the third stage, we use the learned E^{tar} chained with F as the target confidence predictor. (right) Multi-length partial trajectory matching: we align the feature-level distributions (output of encoders E^{src} and E^{tar}) of different lengths' partial trajectories with different discriminators $D_1, D_2, \ldots, D_n$ and align the confidence-level distributions (output of the shared decoder F) of different lengths' partial trajectories with different discriminators $D'_1, D'_2, \ldots, D'_n$.

partial trajectories. Specifically, the discriminator D_k aims to discriminate the latent features of source and target length-k partial trajectories, while the target encoder E^{tar} is trained to confuse the discriminator. The loss for the discriminator D_k can be derived as a binary classification loss:

$$\begin{aligned}\mathcal{L}^k_{\text{fea}} = &- \mathbb{E}_{(s_1,a_1,\ldots,s_k,a_k)\sim\xi,(\xi,)\sim\Xi^{\text{src}}} \\ &\left[\log\left(D_k(\text{Cat}\left(E^{\text{src}}(s_1,a_1),\ldots,E^{\text{src}}(s_k,a_k)\right)\right)\right] \\ &- \mathbb{E}_{(s_1,a_1,\cdots,s_k,a_k)\sim\xi,\xi\sim\Xi^{\text{tar}}} \\ &\left[\log\left(1-D_k(\text{Cat}\left(E^{\text{tar}}(s_1,a_1),\ldots,E^{\text{tar}}(s_k,a_k)\right)\right)\right].\end{aligned} \tag{2}$$

We concatenate the encoded features of multiple consecutive state-action pairs as the latent feature for the length-k partial trajectories. The discriminator D_k is trained to minimize the discriminator loss $\mathcal{L}^k_{\text{fea}}$ and the target encoder E^{tar} is trained to maximize the discriminator loss.

However, the above losses only match the latent feature distributions of state-action pairs, which is confidence-agnostic. So there may still exist mismatches of state-action pairs with different confidence values. As shown in Fig. 2 (right), to address the issue, we introduce a confidence-level distribution matching to match the predicted confidence of source and target state-action pairs, which further guides the latent features to be matched in a confidence-aware way. Specifically, we use another discriminator D'_k for match the length-k confidence sequence of length-k partial trajectories using a similar binary classification loss:

$$\begin{aligned}\mathcal{L}^k_{\text{con}} = &- \mathbb{E}_{(s_1,a_1,\cdots,s_k,a_k)\sim\xi,(\xi,)\sim\Xi^{\text{src}}} \\ &\left[\log\left(D'_k(\text{Cat}\left(F(E^{\text{src}}(s_1,a_1)),\ldots,F(E^{\text{src}}(s_k,a_k))\right)\right)\right] \\ &- \mathbb{E}_{(s_1,a_1,\ldots,s_k,a_k)\sim\xi,\xi\sim\Xi^{\text{tar}}} \\ &\left[\log\left(1-D'_k(\text{Cat}\left(F(E^{\text{tar}}(s_1,a_1)),\ldots,F(E^{\text{tar}}(s_k,a_k))\right)\right)\right].\end{aligned} \tag{3}$$

Similarly, D'_k is trained to minimize the discriminator loss $\mathcal{L}^k_{\text{con}}$ and E^{tar} is trained to maximize the discriminator loss.

$$\min_{E^{\text{src}},F} \mathcal{L}_{\text{reg}} \tag{4}$$

$$\min_{D_k,D'_k} \mathcal{L}^k_{\text{fea}} + \mathcal{L}^k_{\text{con}} (k \in [1,K]) \tag{5}$$

$$\max_{E^{\text{tar}}} \sum_{k=1}^{K} \left(\mathcal{L}^k_{\text{fea}} + \lambda \mathcal{L}^k_{\text{con}}\right) \tag{6}$$

To optimize the whole network, we first train the source encoder and the shared decoder with the source confidence-labeled data as shown in Eqn. (4) (the first stage in Fig. 2 (left)). We then iteratively train the feature-level and confidence-level discriminators and the target encoder as shown in Eqn. (5) and Eqn. (6), where λ is the trade-off between $\mathcal{L}_{\text{fea}}$ and $\mathcal{L}_{\text{con}}$ (the second stage in Fig. 2 (left)). After convergence, we chain the target encoder E^{tar} and the confidence decoder F to form the target confidence predictor as shown under **Conf. Score Inference** in Fig. 2.

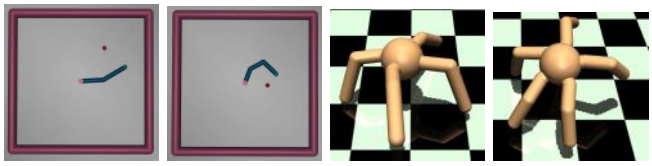
We can then predict the confidence of each state-action pair (s,a) in the target environment as $F(E^{\text{tar}}(s,a))$. With the confidence of each state-action pair, when conducting imitation learning on the target demonstrations, we use the confidence to re-weight each state-action pair in the imitation loss as [6]. Then the imitation learning policy can learn more from useful demonstrations with higher confidence.

Relation to Existing Works. Our method relaxes assumptions in prior works [6], [19] and targets a more realistic problem. However, we argue that our work is compatible with prior works. The common latent feature space in our work can be regarded as a new space of state-action pairs for the source and target environments, where the correspondence between state-action pairs is preserved, especially w.r.t. confidence. With the common latent feature space, we can easily extend prior works to the setting of different state spaces by conducting the algorithm, that is originally performed on states, on the latent feature. Our contribution is not only providing a new approach to learning from imperfect demonstrations but also expanding the usage of algorithms from prior work by introducing a new space to operate in.

V. Experimental Results

We conduct experiments on two MuJoCo environments, a simulated and a real Franka Panda arm. We release the code here. We compare our method (**Ours**) with baseline methods (**GAIL** [3]), Dynamics Cycle-Consistency (**DCC**) [10], and variants of our method by gradually adding modules: starting from GAIL without confidence weighting, we first add feature-level matching as **Ours-Feature**. Then we add the confidence-level matching as **Ours-Confidence**. Adding multi-length partial trajectory matching to Ours-Confidence derives Ours. Such design can separately demonstrate the efficacy of our each module. We show the results of using the ground-truth confidence to reweight the target demonstrations (**Oracle**). We explain how we implement the baselines on our website.

For the Mujoco environments, we evaluate the average return of 100 rollouts. For both the simulated robot and sim-to-real environments, we evaluate the average return of 500 rollouts. In all the experiments, we compute the results over 10 runs and report the mean and the standard deviation. We compute the p-values using the student's t-test. The details on the process of demonstration generation for all environments are shown on our website.



(a) 1-joint (b) 2-joint (c) 4-leg (d) 5-leg

Fig. 3: Illustration of source and target MuJoCo environments. The left two figures are Reacher and the right two are Ant.

MuJoCo Environment. We create 4 different MuJoCo environments (see Fig. 3): 1-joint and 2-joint reacher, 4-leg and 5-leg ant. 1-joint reacher and 4-leg ant are the source environments, and the 2-joint reacher and 5-leg ant are the target environments. The task for the reacher is to reach the red point from its initial configuration. The task for the ant is to move towards the right horizontally as fast as possible. We train the optimal policy using the RL algorithm TRPO [30]. For the Reacher environment, we select the random policy, a partially-trained policy, and the optimal policy to collect demonstrations with mixed confidence. For the Ant environment, we select the random policy, three partially-trained policies, and the optimal policy to collect demonstrations with mixed confidence.

The expected return for these experiments is shown in Table I. **Ours** outperforms **GAIL** and **DCC**, and are comparable with the **Oracle** in both environments. Also, **Ours** outperforms **Ours-Confidence**, which demonstrates the efficacy of multi-step partial trajectory matching. **Ours-Confidence** outperforms **Ours-Feature** and **Ours-Feature** outperforms **GAIL**, which demonstrate the efficacy of confidence-level and feature-level matching respectively. For 2-joint reacher and 5-leg ant, the highest p-values comparing with baselines (between **Ours** and **DCC**) are 0.147 and 0.031 (statistically significant) respectively.

	Reacher	Ant
GAIL	-47.13±5.67	507.85±338.80
DCC	-41.11±5.35	1218.59±231.74
Ours-Feature	-43.97±4.68	1059.02±280.95
Ours-Confidence	-41.82±6.80	1263.04±343.02
Ours	**-39.53**±4.32	**1556.43**±159.12
Oracle	-28.73±3.97	1622.55±179.43

TABLE I: The expected return for MuJoCo environments.

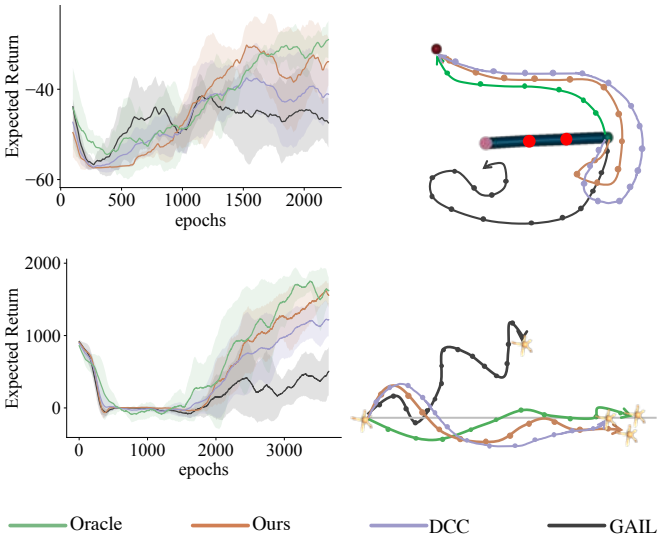


Fig. 4: The up row and the bottom row are the expected return and sample trajectories for 2-joint reacher and 5-leg ant respectively. The light grey line in the Ant environment is the positive x-axis, which is the direction the ant supposes to move.

Figure 4(a) and 4(c) show the learning curves of the algorithms. We observe that all methods have similar convergence rates. Figure 4(b) and 4(d) show a rollout generated by each of the policies. The rollout from the oracle policy reaches the goal in the most effective manner following the shortest trajectory length. **Ours** is the next closest to the **Oracle**, while the **GAIL** baseline fails at reaching the goal position at times.

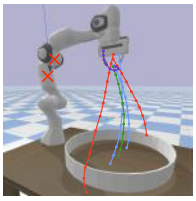


Fig. 5: Illustration of different trajectories and the 5 DoF robot arm.

Method	OT		OSC	
	Return	Success	Return	Success
GAIL	-902.2±152.2	42.8±11.9	-1065.8±105.8	21.2±8.6
DCC	-495.6±46.5	76.4±3.4	-723.7±121.4	49.5±15.2
Ours-Feature	-1198.4±80.4	31.2±7.5	-868.1±265.7	37.6±18.3
Ours-Confidence	-781.9±12.7	59.6±0.8	-821.2±121.4	42.0±11.9
Ours	**-154.8**±12.8	**92.8**±1.0	**-702.2**±104.4	**52.4**±7.9
Oracle	-55.6±2.3	100.0±0.0	-613.9±159.4	59.6±13.2

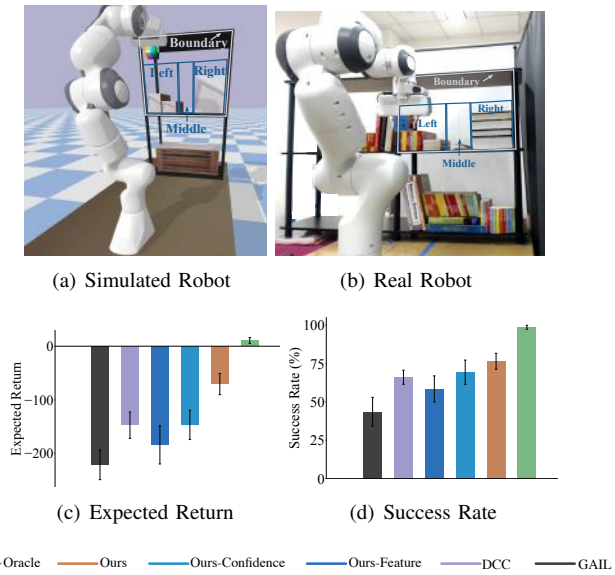
TABLE II: The expected return and the success rate among 500 trials (%) of GAIL, DCC, Ours-Single, Ours w/o $\mathcal{L}_{\text{con}}$, Ours, and Oracle for the two setups OR and OSC in the simulated robot experiments.

Simulated Robot. As shown in Fig. 5, we create a task to move the end-effector of a 5 DoF Panda Franka Robot arm toward the center of a plate without colliding with the boundaries. The reward function consists of the negative L2 distance to the center; collisions with the wall or failing to reach the table within the time limit; and positive reward when the end-effector reaches the plate.

The source robot arm has 7 DoF and the target has 5 DoF, where the joints marked by red in Fig. 5 are disabled. We show different kinds of trajectories: (green–optimal trajectory

reaching the center of the plate), (blue–suboptimal trajectories reaching the plate but not the center), (red–failure cases colliding with the wall), (violet–another type of failure case failing to reach the plate within the time limit). We create two settings: (1) (OT) consists of the optimal (green) and out-of-time (violet) trajectories; and (2) (OSC) consists of optimal (green), suboptimal (blue), and collision (red) trajectories. In the OT setting, we use the joint and end-effector position as the state, which serves as an easy setup since the end-effector position is shared between the 7-DoF and the 5-DoF. In the OSC setting, we use the joint position, velocity and torque as the state, which is difficult to align. For both settings, we use the 3D end-effector position difference as the action space.

Expected return and the success rate are shown in Table II. **Ours** outperforms **GAIL** and **DCC** with a large margin, which indicates that the learned target confidence predictor can assign higher confidence to useful demonstrations than **DCC** and **GAIL**. The order of the performance from high to low is **Ours**, **Ours-Confidence**, **Ours-Feature** and **GAIL**, which demonstrates the efficacy of our multi-length partial trajectory matching, confidence-level matching and feature-level matching respectively. The highest p-values comparing with baselines (between **Ours** and **DCC**) are 5.64×10^{-12} for the success rate and 1.25×10^{-13} for the expected return in the OR setting, and 0.006 for the success rate and 0.031 for the expected return in the OSC setting, demonstrating statistical significance.



(a) Simulated Robot (b) Real Robot

(c) Expected Return (d) Success Rate

Fig. 6: (a-b) Illustration of the simulated robot and the real robot arm environments. Different colors indicate different areas to place the objects. (c-d) Expected return and success rate.

Sim-to-Real Environment. In the sim-to-real environment, we use a simulated and a real Franka Panda Arm as the source and the target respectively (both with 7-DoF). The task is to move the cube to the upper layer of the shelf (see Fig. 6). On the shelf, there is a large stack of books on the right and a small stack on the left. The middle area is empty. We assign positive rewards to success of placing the cube on the shelf, with rewards 200, 300, and 100 for placing on the left, the middle and the right respectively. We penalize time by giving a -1 reward for each time step. The average number of steps for all demonstrations is about 300. If the arm fails to put the cube within 1000 steps, it receives reward 0. We use the joint position, velocity and the end effector position as the state space and joint forces as the action space.

Fig. 6(c) and 6(d) show **Ours** outperforms baselines **DCC** and **GAIL**. The performance from high to low is: **Ours**, **Ours-Confidence**, **Ours-Feature**, and **GAIL**, which demonstrates that multi-length partial trajectory matching, feature-level and confidence-level matching are all necessary to learn a common latent space and an accurate target confidence predictor for the real robot. The p-value between **Ours** and the highest baseline, **DCC**, is 0.0004 for the expected return and 0.0052 for the success rate, demonstrating statistical significance.

Varying Composition of Demonstrations. We conduct experiments by varying the composition of source demonstrations but fix the target demonstration set in the two Mujoco environments to test the robustness of confidence predictor under different demonstration setup. We do not change the target demonstrations since that will also influence the imitation learning performance and we cannot test the efficacy of the confidence predictor. On the column 'Demonstration', we show the number of demonstrations that are collected from different policies from random to optimal in Reacher (3 policies) and Ant (5 policies). As shown in Table III, the first row shows the demonstration setting of the experiments mentioned in Table I. From top to bottom, the optimality of source and target demonstrations deviates more and more. We observe that the performance drops with larger deviation but does not drop too much even on the last row. This demonstrates that the proposed approach can work stably even with confidence distribution shift between demonstrations.

Reacher		**Ant**	
Demonstration	Expected Return	Demonstration	Expected Return
94/5/1	**-36.69**±5.65	48/49/97/5/1	**1525.03**±176.91
50/49/1	-37.38±10.11	48/73/73/5/1	1489.85±268.22
5/94/1	-37.96±6.61	48/97/49/5/1	1436.78±176.91
5/48/47	-37.61±5.50	72/73/49/5/1	1351.44±115.59
5/1/94	-39.49±2.56	97/48/49/5/1	1345.96±299.08

TABLE III: Expected return with respect to varying compositions of the source demonstrations for Reacher and Ant. The numbers in the demonstration column indicate the number of the demonstrations collected from random to optimal policies (3 policies for Reacher and 5 policies for Ant).

VI. Conclusion

Summary. We propose an algorithm to learn from imperfect demonstrations, where the demonstrations can be suboptimal or even fail. We learn a confidence predictor by leveraging confidence labels and demonstrations in a different but correspondent environment. We show the policy learned by our method outperforms other baselines in various environments.
Limitations and Future Work. We require the source and target environments are correspondent, which restricts the applications of our method. In the future, we plan to relax this assumption by defining some correspondence between confidence instead of the strict definition on dynamics.

REFERENCES

[1] P. Abbeel and A. Y. Ng, "Apprenticeship learning via inverse reinforcement learning," in *ICML*, 2004, pp. 1–8.

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

[3] J. Ho and S. Ermon, "Generative adversarial imitation learning," in *NeurIPS*, vol. 29, 2016.

[4] D. P. Losey, K. Srinivasan, A. Mandlekar, A. Garg, and D. Sadigh, "Controlling assistive robots with learned latent actions," in *ICRA*, 2020.

[5] H. A. Simon, *Models of bounded rationality: Empirically grounded economic reason*. MIT press, 1997.

[6] Y.-H. Wu, N. Charoenphakdee, H. Bao, V. Tangkaratt, and M. Sugiyama, "Imitation learning from imperfect demonstration," in *ICML*, 2019.

[7] D. Brown, W. Goo, P. Nagarajan, and S. Niekum, "Extrapolating beyond suboptimal demonstrations via inverse reinforcement learning from observations," in *International Conference on Machine Learning*. PMLR, 2019, pp. 783–792.

[8] D. S. Brown, W. Goo, and S. Niekum, "Better-than-demonstrator imitation learning via automatically-ranked demonstrations," in *Conference on Robot Learning*. PMLR, 2020, pp. 330–359.

[9] L. Chen, R. Paleja, and M. Gombolay, "Learning from suboptimal demonstration via self-supervised reward regression," *arXiv preprint arXiv:2010.11723*, 2020.

[10] Q. Zhang, T. Xiao, A. A. Efros, L. Pinto, and X. Wang, "Learning cross-domain correspondence for control with dynamics cycle-consistency," in *International Conference on Learning Representations*, 2021. [Online]. Available: https://openreview.net/forum?id=QIRlze3I6hX

[11] M. Bain and C. Sammut, "A framework for behavioural cloning." in *Machine Intelligence 15*, 1995.

[12] S. Ross and D. Bagnell, "Efficient reductions for imitation learning," in *AISTATS*, 2010.

[13] S. Ross, G. Gordon, and D. Bagnell, "A reduction of imitation learning and structured prediction to no-regret online learning," in *AISTATS*, 2011.

[14] A. Y. Ng, S. J. Russell *et al.*, "Algorithms for inverse reinforcement learning." in *ICML*, 2000, pp. 663–670.

[15] J. Fu, K. Luo, and S. Levine, "Learning robust rewards with adverserial inverse reinforcement learning," in *ICLR*, 2018.

[16] H. Xiao, M. Herman, J. Wagner, S. Ziesche, J. Etesami, and T. H. Linh, "Wasserstein adversarial imitation learning," *arXiv preprint arXiv:1906.08113*, 2019.

[17] D. S. Brown and S. Niekum, "Deep bayesian reward learning from preferences," *arXiv preprint arXiv:1912.04472*, 2019.

[18] E. Novoseller, Y. Wei, Y. Sui, Y. Yue, and J. Burdick, "Dueling posterior sampling for preference-based reinforcement learning," in *Conference on Uncertainty in Artificial Intelligence*. PMLR, 2020, pp. 1029–1038.

[19] Z. Cao and D. Sadigh, "Learning from imperfect demonstrations from agents with varying dynamics," *IEEE Robotics and Automation Letters (RA-L)*, 2021.

[20] C. L. Nehaniv, K. Dautenhahn *et al.*, "The correspondence problem," *Imitation in animals and artifacts*, vol. 41, 2002.

[21] M. E. Taylor, P. Stone, and Y. Liu, "Transfer learning via intertask mappings for temporal difference learning." *Journal of Machine Learning Research*, vol. 8, no. 9, 2007.

[22] F. Sadeghi and S. Levine, "Cad2rl: Real single-image flight without a single real image," *arXiv preprint arXiv:1611.04201*, 2016.

[23] E. Tzeng, C. Devin, J. Hoffman, C. Finn, P. Abbeel, S. Levine, K. Saenko, and T. Darrell, "Adapting deep visuomotor representations with weak pairwise constraints," in *Algorithmic Foundations of Robotics XII*. Springer, 2020, pp. 688–703.

[24] H. B. Ammar, E. Eaton, P. Ruvolo, and M. Taylor, "Unsupervised cross-domain transfer in policy gradient reinforcement learning via manifold alignment," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 29, no. 1, 2015.

[25] K. H. Kim, Y. Gu, J. Song, S. Zhao, and S. Ermon, "Cross domain imitation learning," 2019.

[26] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel, "Domain randomization for transferring deep neural networks from simulation to the real world," in *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)*. IEEE, 2017, pp. 23–30.

[27] 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)*. IEEE, 2018, pp. 3803–3810.

[28] S. Zakharov, W. Kehl, and S. Ilic, "Deceptionnet: Network-driven domain randomization," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 532–541.

[29] 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.

[30] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, "Trust region policy optimization," in *ICML*, 2015.