Multi-policy Grounding and Ensemble Policy Learning for Transfer Learning with Dynamics Mismatch

Hyun-Rok Lee[1,3] , **Ram Ananth Sreenivasan**[1] , **Yeonjeong Jeong**[2,4] , **Jongseong Jang**[2] , **Dongsub Shim**[2] and **Chi-Guhn Lee**[1]
[1]University of Toronto
[2]LG AI Research
[3]Inha University
[4]York University
hyunrok.lee@inha.ac.kr, ram.sreenivasan@mail.utoronto.ca, yjjeong@yorku.ca, {j.jang, dongsub.shim}@lgresearch.ai, cglee@mie.utoronto.ca

Abstract

We propose a new transfer learning algorithm between tasks with different dynamics. The proposed algorithm solves an Imitation from Observation problem (IfO) to ground the source environment to the target task before learning an optimal policy in the grounded environment. The learned policy is deployed in the target task without additional training. A particular feature of our algorithm is the employment of multiple rollout policies during training with a goal to ground the environment more globally; hence, it is named as Multi-Policy Grounding (MPG). The quality of final policy is further enhanced via ensemble policy learning. We demonstrate the superiority of the proposed algorithm analytically and numerically. Numerical studies show that the proposed multi-policy approach allows comparable grounding with single policy approach with a fraction of target samples, hence the algorithm is able to maintain the quality of obtained policy even as the number of interactions with the target environment becomes extremely small.

1 Introduction

Reinforcement learning (RL) learns a policy of high return through interactions with the environment. Despite its successes in various domains, a critical limitation of RL algorithms is that a large number of transition samples are required to obtain an effective solution. Thus, transfer learning in RL seeks to minimize interactions with the target environment through source knowledge [Taylor and Stone, 2009]. In this work, we address the specific transfer learning problem in which source and target tasks are different only with respect to their dynamics.

In general, knowledge obtained from source tasks with less dynamics differences to the target task offers better opportunities for a successful transfer across tasks with different dynamics [Wang *et al.*, 2020]. This has motivated a handful of methods modifying the dynamics of source task to reduce dynamics mismatch between source and target task [Hanna and Stone, 2017; Desai *et al.*, 2020], which aim to effectively ground the source environment through an action transformation. A recent work [Desai *et al.*, 2020] shows that the dynamics matching problem based on action transformation can be cast as an imitation from observation (IfO) problem. Based on this intuition, generative adversarial reinforced action transformation (GARAT) extends generative adversarial imitation learning (GAIL) to learn an action transformation policy that modifies dynamics of a source environment to minimize the discrepancy with respect to target dynamics. While GARAT can successfully reduce the dynamics mismatch between the source and the target environments, doing so is challenging with a limited number of samples, which especially being important when the target system is physical such as robot.

In this study, we propose to solve multiple IfO problems in a single grounding process for better grounding with much fewer samples. GARAT uses a single rollout policy to collect samples from the target environment. However, grounding with a single rollout policy can lead to a locally grounded environment resulting in dynamics matching only near trajectories of the rollout policy. We tackle this issue by learning an action transformation policy with multiple rollout policies, which we call multi-policy grounding (MPG). Through the use of multiple rollout policies, we expect the improvement of grounding quality.

Moreover, when learning a policy in grounded environments, we employ an ensemble method that uses multiple grounded environments. This is to avoid the possibility of obtaining a policy that works well only in a particular grounded environment. Hence, we learn a single agent policy that performs well across multiple grounded environments with a hope that the resulting policy generalizes well to the target task. The contribution of our work is three-fold as follows:

- We propose multi-policy grounding (MPG) algorithm to obtain generalizable action transformation policy with a small number of samples;
- We employ ensemble method to find more generalizable policy in the target task;

- We provide performance guarantee of the MPG.

Due to space limitations, we provide the supplementary material including appendices and our code at https://github.com/hyunrok/MPGE.

2 Related Work

In transfer RL for dynamics mismatch, many studies assume multiple source policies, usually from a library of multiple source tasks, are given a priori and propose a method to effectively choose or combine these policies. Option learning [Yang *et al.*, 2020b], policy function aggregation [Barekatain *et al.*, 2019], policy shaping [Plisnier *et al.*, 2019], and bayesian inference based mixture density network [Gimelfarb *et al.*, 2020] are different methods proposed to fully utilize multiple source policies. Another stream of research assumes the existence of hidden parameters that define transition dynamics of environment [Killian *et al.*, 2017; Perez *et al.*, 2020; Yang *et al.*, 2020a]. These studies exploit the advantage of hidden parameters that simplify the estimation of dynamics differences, hence facilitate transfer learning. Our work differs with these methods as we assume access to only one source task whose parameters may not be easy to modify or estimate (for example, a black-box simulator).

Modification on tasks itself is alternative approach for effective transfer. Potential-based reward shaping has been used for modifying reward function at target task through source knowledge while preventing possibility of negative transfer [Brys *et al.*, 2015; Gimelfarb *et al.*, 2020]. On the other hand, recent work set additional penalties/incentives in reward functions of source task to consider dynamics differences while policy learning [Eysenbach *et al.*, 2020].

Our work is more relevant to studies that modify the source dynamics. Policy robustness methods [Pinto *et al.*, 2017; Jakobi, 1997] are such examples that perturb the source task dynamics to train robust policy that perform well across a range of different dynamics. Simulation grounding in sim-to-real setting is relevant in terms of enhancing transferability by reducing discrepancy of dynamics [Chebotar *et al.*, 2019; Farchy *et al.*, 2013; Zhang *et al.*, 2020]. Especially, action transformation policy based approaches such as GAT [Hanna and Stone, 2017] and GARAT [Desai *et al.*, 2020] possess the advantage of modifying dynamics without direct manipulation of simulator. Our study also takes advantage of the action transformation based simulation grounding approaches that work for black-box simulators and propose a modification on top of the GARAT framework. We propose the idea of using multiple policies to ground the simulator in addition to using an ensemble of multiple simulated source domains inspired by previous work that show improved sample efficiency when using ensembles in RL [Rajeswaran *et al.*, 2017; Chua *et al.*, 2018; Kurutach *et al.*, 2018; Lee *et al.*, 2021].

3 Preliminaries

3.1 MDP and Dynamics Mismatch Setting

We formulate each RL task as a Markov Decision Process (MDP) denoted by the tuple $\langle S, A, R, P, \gamma \rangle$ where S represents the state space; A represents the action space; R represents the scalar reward function $S \times A \times S \mapsto \mathbb{R}$; $P(s'|s,a)$ represents the transition dynamics corresponding to taking action a in state s resulting in a next state s' and $\gamma \in [0,1)$ represents the discount factor. A policy $\pi \in \Pi$ is a mapping from S to a distribution over A that specifies the behavior of the RL agent. The objective of the agent is to find a policy that maximizes the expected return $J(\pi) = \mathbb{E}_{\pi,P}\Big[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t, s_{t+1})\Big]$ where s_0 is initial state, $a_t \sim \pi(\cdot|s_t)$ and $s_{t+1} \sim P(\cdot|s_t, a_t)$ for $t \geq 0$. The objective can also be rewritten in terms of the marginal transition distribution ρ_π as $J(\pi) = \frac{1}{1-\gamma}\sum_{s,a,s'} \rho_\pi(s,a,s') r(s,a,s')$. ρ_π is given by

$$\rho_\pi(s,a,s') = (1-\gamma)\pi(a|s)P(s'|s,a)\sum_{t=0}^{\infty}\gamma^t p(s_t = s|\pi, P) \tag{1}$$

where $p(s_t = s|\pi, P)$ is the probability of being at state s at time t under dynamics P and π. The current scenario of transfer learning with dynamics mismatch assumes that only the transition dynamics P varies across the different tasks.

3.2 Action Transformation based Grounding

The objective of grounding is matching the dynamics of source environment to that of the target task. Learning an action transformation policy π_g is useful for grounding when modifying the source dynamics P_{src} directly is difficult due to its black-box nature [Hanna and Stone, 2017]. The policy π_g is applied to the source dynamics P_{src} as an additional transition so that $P_g(s'|s,a) = \sum_{\tilde{a} \in A} P_{src}(s'|s,\tilde{a})\pi_g(\tilde{a}|s,a)$. Action transformation based grounding seeks to learn a π_g that minimizes the difference between grounded dynamics P_g and target task dynamics P_{trg}.

Generative Adversarial Reinforced Action Transformation (GARAT) [Desai *et al.*, 2020] learns action transformation policy by matching marginal transition distributions across the source and target tasks under a single rollout policy. This is done by defining an Imitation from Observation (IfO) problem on an augmented MDP for action transformation. State space of the augmented MDP is the joint state-action space, with elements $x = (s,a)$, of the original MDP and the actions correspond to the transformed action $\tilde{a}$. A policy function for this MDP is $\pi_g(\tilde{a}|s,a)$, and transition dynamics for the underlying rollout policy π is defined as $P(s',a'|s,a,\tilde{a}) = P_{src}(s'|s,\tilde{a})\pi(a'|s')$.

Transition samples (s,a,s') obtained with rollout policy π in target dynamics P_{trg} can be interpreted as demonstration data for IfO problem of the augmented MDP given by an optimal action transformation policy π_g^*. Based on this fact, GARAT leverages recent approaches from the IfO literature [Torabi *et al.*, 2018]. For a discriminative classifier $D(s,a,s') : S \times A \times S \mapsto (0,1)$, that distinguishes if the observed transition (s,a,s') is from the grounded (label 1) or target environment (label 0), the objective of GARAT is given by

$$J_{GARAT} = \min_{\pi_g \in \Pi_g} \max_{D} \mathbb{E}_{\pi, P_g}[\log D(s,a,s')] \\ + \mathbb{E}_{\pi, P_{trg}}[\log(1 - D(s,a,s'))], \quad (2)$$

where $\mathbb{E}_{\pi, P_{(\cdot)}}$ denotes expectation with respect to transition samples (s,a,s') collected by following π in dynamics $P_{(\cdot)}$. The policy π_g is learnt with reward function $-\log D(s,a,s')$ in the augmented MDP problem.

4 Multi-policy Grounding and Ensemble Policy Learning

Grounding-and-then-Policy Learning (GPL) approach aims to learn a policy $\pi_a \in \Pi$ that will be optimal in target task in two steps: (1) grounding the source environment and (2) policy learning in the grounded environment. In the first step, action transformation policy π_g is found as an optimal solution to the augmented MDP. This step uses a limited number of transition samples obtained with rollout policy at the target task. In the second step, optimal policy π_a is learned in the grounded environment defined through π_g without additional interactions with the target task.

We observed that the GPL suffered from two major issues: (1) grounding is done only locally around trajectories visited by the used single rollout policy, and (2) policy found to be optimal in the grounded environment catastrophically failed in the target environment. To address the issues, we extend the GPL in two ways: adoption of multiple policies as rollout policies (to address the first issue) and ensemble learning over multiple grounded environments (to address the second issue).

Figure 1 provides a schematic representation of the multi-policy grounding and ensemble policy learning method (MPGE). Note that grounding and policy learning are independent so various combinations should work such as multi-policy grounding and non-ensemble learning (MPG), single-policy grounding and non-ensemble learning (SPG), and alike.

Multi-policy grounding algorithm. The MPG algorithm extends GARAT [Desai *et al.*, 2020] by adopting K rollout policies. While a single grounding policy π_g is sought for, MPG considers the mixture of marginal transition distributions of K rollout policies. The objective function of MPG is defined by aggregating the performance of π_g in K MDP problems using a common discriminator across multiple rollout policies as follows:

$$J_{MPG} = \min_{\pi_g \in \Pi_g} \max_{D} \sum_{i=1}^{K} \big[\mathbb{E}_{\pi_i, P_g}[\log D(s,a,s')] \\ + \mathbb{E}_{\pi_i, P_{trg}}[\log(1 - D(s,a,s'))]\big] \quad (3)$$

where $D(s,a,s') : S \times A \times S \mapsto (0,1)$ distinguishes if the observed transition (s,a,s') is from the grounded or target environment when K rollout policies has been deployed. More intuitive interpretations of Equation (3) as a grounding objective is provided in Appendix A.

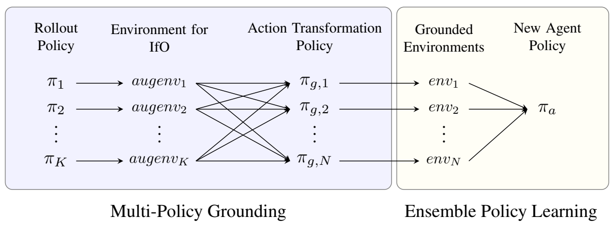

Figure 1: Overall framework of the proposed transfer RL framework

In case there exists $\bar{\pi}_g$ that can perfectly match the dynamics across the source and target tasks, MPG finds $\bar{\pi}_g$ provided unlimited target samples are available.

Proposition 1. *If there exists* $\bar{\pi}_g \in \Pi_g$ *such that* $P_g(s'|s,a) = P_{trg}(s'|s,a)\ \forall s,s' \in S, a \in A$, *then* J_{MPG} *is minimized by* $\bar{\pi}_g$.

Proposition 1 provides the performance guarantee that MPG will find an optimal grounding policy if exists. Proof is provided in Appendix B.

The MPG algorithm, given in Algorithm 1, finds a solution to K augmented MDPs defined by rollout policies $\{\pi_1, ..., \pi_K\}$ using a reward function $-\log D(s,a,s')$. While a single policy network for $\pi_g(\tilde{a}|s,a)$ is learned across K augmented MDPs, K value networks are updated for value functions $V_i^{\pi_g}(s,a)$ of i^{th} of K augmented MDPs. Algorithm 1 is presented assuming TRPO [Schulman *et al.*, 2015] as the RL algorithm for learning π_g, but any other actor-critic RL algorithms will do.

Algorithm 1 Multi-policy Grounding (MPG)

```
Input: Set of rollout policies $\{\pi_1, ..., \pi_K\}$, demonstration buffers $B_{i,trg}$ for $i \in \{1,..,K\}$
 1: Initialize action transformation policy $\pi_g(\tilde{a}|s,a)$, value network $V_i^{\pi_g}(s,a)$, and discriminators $D(s,a,s')$.
 2: Uniformly sample index $i \in \{1,...,K\}$, and initialize state $s$
 3: for $\lfloor$total time steps/batch size$\rfloor$ do
 4:     Initialize buffers for transition samples from grounded environments $B_{i,g} = \emptyset\ \forall i \in \{1,...,K\}$
 5:     for batch size do
 6:         if $s$ = terminal state then
 7:             Uniformly sample index $i \in \{1,...,K\}$, initialize state $s$
 8:         end if
 9:         $a \sim \pi_i(a|s)$, $s' \sim \pi_i(a|s)\pi_g(\tilde{a}|s,a)P_{src}(s'|s,\tilde{a})$
10:         Add $(s,a,s')$ to $B_{i,g}$
11:         $s \leftarrow s'$
12:     end for
13:     Update $D(s,a,s')$ to maximize
14:         $\sum_{i=1}^{K} \big[\mathbb{E}_{B_{i,g}}[\log D(s,a,s')]$
15:             $+\mathbb{E}_{B_{i,trg}}[\log(1 - D(s,a,s'))]\big]$
16:     Update $\pi_g(\tilde{a}|s,a)$ and $V_i^{\pi_g}(s,a)$ using TRPO rule based on reward $-\log D(s,a,s')$ and samples in $B_{i,g}$
17: end for
```

Diversification of rollout policy. As discussed earlier, a near perfect grounding given a rollout policy suffers from overfitting. This is due to the tendency that grounding is done locally along the trajectories generated by the rollout policy. As a result, dramatic degradation of policy performance can occur whenever optimal policy deviates from the trajectories of the rollout policies. Therefore, we focus on diversification of rollout policies. A naive approach can be to generate rollout policies using multiple random seeds. The random rollout policies, which are individually optimal in the source domain, may not cover the environment globally and hence still prone to overfitting. Therefore, we propose a sequential generation of rollout policies for more controlled diversification using entropy as a regularizer. Specifically, we add the state-specific entropy of previously generated rollout policies when a new rollout policy is sought for. The entropy regularization term acting as a bonus for a state s given l already-generated rollout policies is defined as $\frac{1}{l}\sum_{i=1}^{l} H\big(\pi_i(\cdot|s)\big)$. This bonus term is motivated by an observation that distribution over actions from a state away from optimal trajectory tends to remain uniform (i.e., high entropy). It makes an intuitive sense to avoid state transitions sufficiently explored by previously generated rollout policies.

Ensemble policy learning. After grounding the source dynamics P_{src} to P_g, we learn the agent policy π_a in the grounded environment using a RL algorithm. However, because we are using limited number of target samples for grounding, errors in action transformation are inevitable. This could lead to poor performance of π_a, optimal in the grounded environment, when it is actually used in the target task. To mitigate the performance degradation, we learn π_a in multiple grounded environments, hence called ensemble policy learning. The ensemble approach effectively regularizes the policy learning in grounded environment by preventing the learned policy from overfitting to a particular grounded environment. To this end, we learn π_g with N random seeds and same $B_{i,trg}\, \forall i \in \{1,..,K\}$ to form a collection of such policies: $\{\pi_{g,1},...,\pi_{g,N}\}$. During the policy learning, we uniformly sample an action transformation policy to define dynamics P_g at the beginning of each episode. That is, $P_g(s'|s,a) = \sum_{\tilde{a}\in A} P_{src}(s'|s,\tilde{a})\pi_{g,n}(\tilde{a}|s,a)$ for chosen $\pi_{g,n} \in \{\pi_{g,1},...,\pi_{g,N}\}$. As a result, we learn π_a whose performance is strong across N grounded environments. Now we have N as a hyper-parameter, and expect π_a to be more robust in the target task. Note that MPG and MPGE will not require additional computation over SPG and MPG, respectively. This is because MPS uses the same number of samples as MPG but simply adopts multiple policies for sampling, and because MPGE carries out the same number of training iterations for policy optimization but the training is done on multiple environments.

5 Experiments

In this section, we compare the proposed methods (**MPG** and **MPGE**) against the existing one (**SPG** or equivalently GARAT). In particular, we seek to answer the following questions by transferring an agent policy between environments with different transition dynamics:

1. Can the multi-policy grounding method lead to a better matching of the dynamics?
2. How does multiple rollout policies affect the final performance at the target environment?
3. Does the ensemble approach in policy learning provide any benefits?

Transfer scenarios. We validate the performance of the proposed methods in continuous control environments provided by OpenAI gym [Brockman *et al.*, 2016], namely InvertedPendulum, Hopper and HalfCheetah. For more interpretable results, we use transfer scenarios where the true action transformation is known. This is done by crippling one of the joints of the robots in target domain for Hopper and HalfCheetah to create the BrokenHopper and BrokenCheetah environments, respectively [Eysenbach *et al.*, 2020]. The fully functional robots are used as the source tasks. Alternatively, for the InvertedPendulum environment, the impact of the action on the robot is modified in the target task such that there is an additional positive "force" ϵ acting on the robot. These three transfer learning scenarios provide meaningful experiments to analyze the quality of the action transformation learnt. We also include a task where the true action transformation is not known to address more challenging mismatch between the source and the target tasks. Similar to [Desai *et al.*, 2020], the torso mass of the HalfCheetah is increased to create a HeavyCheetah for the target task. A full description of the tasks and transfer learning scenarios is provided in the Appendix C.

Experimental settings. We constructed two different types of rollout policy sets. Multi-seed (MS) set has 5 optimal policies for source task learned by the SAC algorithm with 5 random seeds. Entropy-regularized (ER) set has 5 source optimal policies that are generated sequential according to the diversification scheme explained earlier in Section 4. The entropy regularization term is added as $0.1 \times \frac{1}{l}\sum_{i=1}^{l} H\big(\pi_i(\cdot|s)\big)$ for l previously generated rollout policies. Rollout policies are trained for 2 million time steps in InvertedPendulum and 3 million time steps in the other environments. We also trained 5 target optimal policies at the target task with the same setting for comparison. Then, we collected transition samples from the target environment using these rollout policies, and validated the performance of the proposed grounding methods using six target sample sizes: 500, 1000, 5000, 10000, 50000, and 100000. We varied the number of rollout policies K as 1 (SPG), 3, and 5 in MPG. Note that we keep the total number of samples used across all rollout policies under control. When the total number of samples is 1000, for example, we collected 200 samples per rollout policy in case of MPG using 5 rollout policies. We used 5 action transformation policies obtained from MPG to define 5 grounded environments to MPGE.

Implementation of MPG algorithm is based on 'stable-baselines' library [Hill *et al.*, 2018] and codes released for GARAT [Desai *et al.*, 2020]. We learned π_g with TRPO algorithm over 5 million time steps in InvertedPendulum and 20 million time steps for the others, and rollout policy is uniformly selected at the start of each episode. While the action

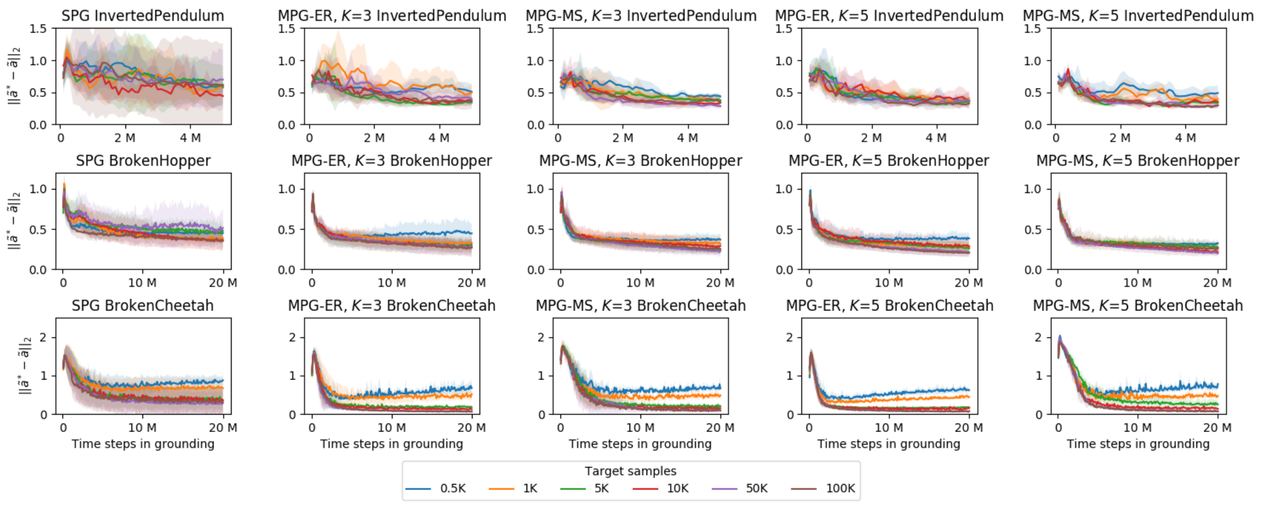

Figure 2: Average of action transformation errors while grounding. Shaded area represents standard deviation across five different random seeds.

transformation policy is updated every 2048 time steps, we divided the sample batch by 512 and updated the discriminator 4 times with normalized inputs. The discriminator consists of two hidden layers of 256 units each, with *tanh* as activation function, and a gradient penalty term is used as a regularizer. When learning the agent policy in the grounded environment, the source optimal policies obtained from the original source task are used as the initial policies, and the agent policies are trained with the same settings as when obtaining the rollout policy. Most hyperparameters for RL algorithms refer to 'RL-baselines zoo' [Raffin, 2018], except for setting the discount factor to 0 while learning π_g to obtain less dispersed action transformation. Detailed parameters are fully described in the Appendix C.

Analysis of action transformation error. We examine whether MPG matches the dynamics better than SPG in three environments (except HeavyCheetah). We compute action transformation error as the expectation of L2-norm of differences between learned action transformation $\tilde{a}$ and optimal action transformation $\tilde{a}^*$: $\mathbb{E}_{(s,a,\tilde{a})\sim\pi,\pi_g,P_{src}}[\|\tilde{a}^* - \tilde{a}\|_2]$. Figure 2 shows how the action transformation error changes during grounding. For MPG, we have two results according to rollout policy set used for grounding. **MPG-ER** and **MPG-MS** denote MPG algorithm using entropy-regularized set and multi-seed set, respectively. We plot the average of the action transformation error for 2048 transition samples obtained with a random policy π in the grounded environment for every 100000 time steps. Each algorithm has been evaluated five times. Note that grounding is done by π_g and the evaluation of grounding is done by π.

A larger number of rollout policies used during grounding should intuitively reduce the discrepancy of the marginal transition distributions between the grounded and the target environments. As expected, both MPG-ER and MPG-MS perform better than SPG over training steps using the same number of samples in terms of the average grounding error and the variance of grounding error as shown in Figure 2. On the other hand, more rollout policies would be less effective if they are not diversified enough, which is supported by an observation that the improvement is less when going from MPG $K = 3$ to $K = 5$ than going from $K = 1$ (i.e., SPG) to $K = 5$.

The advantage of MPG can be due to that MPG prevents local grounding owing to multiple rollout policies. We provide a further analysis on issues in grounding done by SPG in Appendix D. MPG-ER, which controls diversification of rollout policies, seems to perform better in BrokenCheetah, but the difference between MPG-MS and MPG-ER is less obvious in general. This suggests that grounding with multiple rollout policies could be overall beneficial but the marginal improvement is decreasing as the number of rollout policies grows. In the following experiments, we used five rollout policies.

Benefits of ensemble learning. Ensemble learning tries to improve the quality of policy π_a evaluated in the target task by optimizing π_a across multiple grounded environments. This way, the resulting policy is more generalizable as evident in Figure 3, where MPGE outperforms not only SPG (thanks to multiple rollout policy grounding) but also MPG (thanks to ensemble). Figure 3 shows that the performance gap of the three algorithms between the grounded and the target environment. Note that MPG-ER with 500 target samples is used in Figure 3 and we obtained similar results from MPG-MS. Interesting to observe that the ensemble (MPGE) makes more significant improvement than multi-policy grounding (MPG) does over SPG.

Performance at target task. Figure 4 shows the performance of the policy π_a, optimal in the grounded environment, in the target task as the sample size increases from 500 to 100000. Each algorithm has been evaluated 5 times with different random seeds, each rollout policy set for MPG-ER and MPG-MS contains 5 rollout policies, and 5 action transformation policies obtained from MPG to define 5 grounded environments to MPGE. We evaluated the performance of a pol-

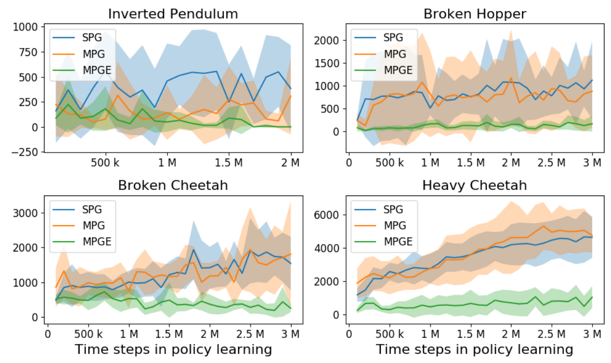

Figure 3: Differences of performance in grounded and target environment while learning π_a in grounded environment when using 500 target samples. Y-axis indicates the average of absolute performance gaps. Shaded area represents standard deviation across five different random seeds.

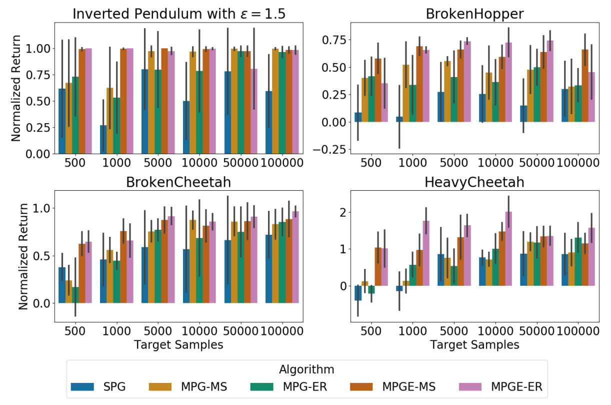

Figure 4: Comparison of performances at the target task. Bars indicate standard deviation of results across 5 random seeds.

icy with the average of returns in 50 episodes. We also normalized performance value by setting the performance when rollout policies are directly transferred to the target task as 0 (direct transfer) and the performance of policies learned in the target task as 1 (target optimal).

MPG-ER and MPG-MS generally outperform SPG with a few exceptions. This improvement can be due to more global grounding via multiple rollout policies. Performance of MPG without ensemble method (i.e., MPG-MS and MPG-ER) is decent as the target sample size varies from mere 500 to 100000. When ensemble is turned on, the performance improvement is quite noticeable. MPGE achieves more than 75% of the target optimal value with target transition sample size as few as 1000 and more with a single exception from BrokenHopper, in which MPGE shows at least 65% of the target optimal with sample size 1000 and above. In particular, for BrokenHopper and HeavyCheetah environments, MPGE was beneficial to find an agent policy that performs better on the target tasks compared to MPG. In HeavyCheetah, the ensemble methods outperform the optimal policy trained up to 3 million time steps in the target domain, which is why the normalized performance shown in Figure 4 is larger than 1.

6 Conclusions

In this work, we propose a transfer reinforcement learning method for dynamics mismatch setting by modifying the source dynamics through action transformation-based grounding. The existing methods have been found to suffer from dramatic degradation in target performance despite near perfect grounding, which motivated us to introduce multiple policy grounding (MPG). The target performance is further enhanced with ensemble policy training. Specifically, we re-write the objective function to allow multiple rollout policies yet a single discriminator to avoid overfitting. Ensemble method is to reduce the performance gap between the grounded and the target environments. Numerical experiments in four transfer scenarios based on continuous control task show that MPG algorithm and the ensemble method for policy learning could improve overall grounding quality and achieve better performance at target task.

While MPG and ensemble policy learning improves the stability of grounding algorithm and performance at target task, small errors in action transformation can still cause devastating performance collapse in more challenging scenarios. For more stable and effective grounding, we plan to extend the current approach in many directions. First, the MPG can be re-design so that it should reduce the discrepancy of trajectory distributions, rather than transition distribution, given our observation that globally grounded environment results in a better target performance. Second, more effective action transformation can be achieved based on the understanding of the dependency structure between state and action dimensions at the source task.

Acknowledgments

This research was supported by LG AI Research.

References

[Barekatain *et al.*, 2019] Mohammadamin Barekatain, Ryo Yonetani, and Masashi Hamaya. Multipolar: Multi-source policy aggregation for transfer reinforcement learning between diverse environmental dynamics. *ArXiv*, abs/1909.13111, 2019.

[Brockman *et al.*, 2016] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

[Brys *et al.*, 2015] Tim Brys, Anna Harutyunyan, Matthew E Taylor, and Ann Nowé. Policy transfer using reward shaping. In *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems*, pages 181–188, 2015.

[Chebotar *et al.*, 2019] Yevgen Chebotar, Ankur Handa, Viktor Makoviychuk, Miles Macklin, Jan Issac, Nathan Ratliff, and Dieter Fox. Closing the sim-to-real loop: Adapting simulation randomization with real world experience. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 8973–8979. IEEE, 2019.

[Chua *et al.*, 2018] Kurtland Chua, R. Calandra, Rowan McAllister, and Sergey Levine. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. In *NeurIPS*, 2018.

[Desai *et al.*, 2020] Siddharth Desai, Ishan Durugkar, Haresh Karnan, Garrett Warnell, Josiah Hanna, Peter Stone, and AI Sony. An imitation from observation approach to transfer learning with dynamics mismatch. *Advances in Neural Information Processing Systems*, 33, 2020.

[Eysenbach *et al.*, 2020] Benjamin Eysenbach, Swapnil Asawa, Shreyas Chaudhari, Ruslan Salakhutdinov, and Sergey Levine. Off-dynamics reinforcement learning: Training for transfer with domain classifiers. *arXiv preprint arXiv:2006.13916*, 2020.

[Farchy *et al.*, 2013] Alon Farchy, Samuel Barrett, Patrick MacAlpine, and Peter Stone. Humanoid robots learning to walk faster: From the real world to simulation and back. In *Proceedings of the 2013 international conference on Autonomous agents and multi-agent systems*, pages 39–46, 2013.

[Gimelfarb *et al.*, 2020] Michael Gimelfarb, Scott Sanner, and Chi-Guhn Lee. Contextual policy reuse using deep mixture models. *arXiv preprint arXiv:2003.00203*, 2020.

[Hanna and Stone, 2017] Josiah Hanna and Peter Stone. Grounded action transformation for robot learning in simulation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31, 2017.

[Hill *et al.*, 2018] Ashley Hill, Antonin Raffin, Maximilian Ernestus, Adam Gleave, Anssi Kanervisto, Rene Traore, Prafulla Dhariwal, Christopher Hesse, Oleg Klimov, Alex Nichol, Matthias Plappert, Alec Radford, John Schulman, Szymon Sidor, and Yuhuai Wu. Stable baselines. https://github.com/hill-a/stable-baselines, 2018. Accessed: 2022-05-23.

[Jakobi, 1997] Nick Jakobi. Evolutionary robotics and the radical envelope-of-noise hypothesis. *Adaptive behavior*, 6(2):325–368, 1997.

[Killian *et al.*, 2017] Taylor W Killian, Samuel Daulton, George Konidaris, and Finale Doshi-Velez. Robust and efficient transfer learning with hidden parameter markov decision processes. In *Advances in neural information processing systems*, pages 6250–6261, 2017.

[Kurutach *et al.*, 2018] Thanard Kurutach, Ignasi Clavera, Yan Duan, Aviv Tamar, and Pieter Abbeel. Model-ensemble trust-region policy optimization. *arXiv preprint arXiv:1802.10592*, 2018.

[Lee *et al.*, 2021] Kimin Lee, Michael Laskin, Aravind Srinivas, and Pieter Abbeel. Sunrise: A simple unified framework for ensemble learning in deep reinforcement learning. In *International Conference on Machine Learning*, pages 6131–6141. PMLR, 2021.

[Perez *et al.*, 2020] Christian F Perez, Felipe Petroski Such, and Theofanis Karaletsos. Generalized hidden parameter mdps transferable model-based rl in a handful of trials. *arXiv preprint arXiv:2002.03072*, 2020.

[Pinto *et al.*, 2017] Lerrel Pinto, James Davidson, Rahul Sukthankar, and Abhinav Gupta. Robust adversarial reinforcement learning. In *International Conference on Machine Learning*, pages 2817–2826, 2017.

[Plisnier *et al.*, 2019] Hélène Plisnier, Denis Steckelmacher, Diederik M. Roijers, and Ann Nowé. Transfer reinforcement learning across environment dynamics with multiple advisors. *CEUR-WS*, 2019.

[Raffin, 2018] Antonin Raffin. Rl baselines zoo. https://github.com/araffin/rl-baselines-zoo, 2018. Accessed: 2022-05-23.

[Rajeswaran *et al.*, 2017] Aravind Rajeswaran, Sarvjeet Ghotra, Sergey Levine, and Balaraman Ravindran. Epopt: Learning robust neural network policies using model ensembles. *ArXiv*, abs/1610.01283, 2017.

[Schulman *et al.*, 2015] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International conference on machine learning*, pages 1889–1897. PMLR, 2015.

[Taylor and Stone, 2009] Matthew E Taylor and Peter Stone. Transfer learning for reinforcement learning domains: A survey. *Journal of Machine Learning Research*, 10(Jul):1633–1685, 2009.

[Torabi *et al.*, 2018] Faraz Torabi, Garrett Warnell, and Peter Stone. Generative adversarial imitation from observation. *arXiv preprint arXiv:1807.06158*, 2018.

[Wang *et al.*, 2020] Yue Wang, Yuting Liu, Wei Chen, Zhi-Ming Ma, and Tie-Yan Liu. Target transfer q-learning and its convergence analysis. *Neurocomputing*, 2020.

[Yang *et al.*, 2020a] Jiachen Yang, Brenden Petersen, Hongyuan Zha, and Daniel Faissol. Single episode policy transfer in reinforcement learning. In *International Conference on Learning Representations*, 2020.

[Yang *et al.*, 2020b] Tianpei Yang, Jianye Hao, Zhaopeng Meng, Zongzhang Zhang, Yujing Hu, Yingfeng Chen, Changjie Fan, Weixun Wang, Zhaodong Wang, and Jiajie Peng. Efficient deep reinforcement learning through policy transfer. In *Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems*, pages 2053–2055, 2020.

[Zhang *et al.*, 2020] Qiang Zhang, Tete Xiao, Alexei A Efros, Lerrel Pinto, and Xiaolong Wang. Learning cross-domain correspondence for control with dynamics cycle-consistency. *arXiv preprint arXiv:2012.09811*, 2020.