MULTIPOLAR: MULTI-SOURCE POLICY AGGREGATION FOR TRANSFER REINFORCEMENT LEARNING BETWEEN DIVERSE ENVIRONMENTAL DYNAMICS

Mohammadamin Barekatain*
Technical University of Munich
Munich, Germany
m.barekatain@tum.de

Ryo Yonetani & Masashi Hamaya
OMRON SINIC X
Tokyo, Japan
{ryo.yonetani,masashi.hamaya}@sinicx.com

ABSTRACT

This work explores a new challenge in transfer reinforcement learning (RL), where only a set of source policies collected under diverse unknown dynamics is available for quickly learning a target task. To address this problem, we propose MULTI-source POLicy AggRegation (MULTIPOLAR), which comprises two key techniques. 1) Learning to aggregate the actions provided by the source policies adaptively to maximize the target task performance. 2) Learning an auxiliary network that predicts residuals around the aggregated actions, which ensures the target policy's expressiveness even when some of the source policies perform poorly. We confirmed the significant effectiveness of MULTIPOLAR across six simulated environments ranging from classic control problems to challenging robotics simulations, under both continuous and discrete action spaces. The videos and code are available on the project webpage: https://yonetaniryo.github.io/multipolar/.

1 INTRODUCTION

We envision a future scenario where a variety of robotic systems, which are each trained or manually engineered to solve a similar task, provide their policies for a new robot to learn a relevant task quickly. For example, imagine various pick-and-place robots working in factories all over the world. Depending on the manufacturer, these robots will differ in their kinematics (*e.g.*, link length, joint orientation) and dynamics (*e.g.*, link mass, joint damping, friction, inertia). They could provide their policies to a new robot (Devin et al., 2017), even though their dynamics factors, on which the policies are implicitly conditioned, are not typically available (Chen et al., 2018). Moreover, we cannot rely on a history of their individual experiences, as they may be unavailable due to a lack of communication between factories or prohibitively large dataset sizes. In such scenarios, a key technique to develop would be the ability to transfer knowledge from a collection of robots to a new robot quickly *only by exploiting their policies while being agnostic to their different kinematics and dynamics*, rather than collecting a vast amount of samples to train the new robot from scratch.

The scenario illustrated above poses a new challenge in the transfer learning for reinforcement learning (RL) domains. Formally, consider multiple instances of a single environment with diverse state transition dynamics, *e.g.*, independent robots presented in Figure 1 (left), which reach different states by executing the same actions due to the differences in their kinematics and dynamics designs. Some source agents interacting with one of the environment instances provide their deterministic policy to a new target agent in another environment instance. Then, our problem is: *can we efficiently learn the policy of a target agent given only the collection of source policies?* Note that information about source environmental dynamics, such as the exact state transition distributions and the history of environmental states, is not visible to the target agent. Also, the source policies are neither trained nor hand-engineered for the target environment instance, and therefore not guaranteed to work optimally and may even fail (Chen et al., 2018). Importantly, these conditions prevent us from adopting existing work on transfer RL between different environmental dynamics, as they require access to source environment instances or their dynamics for training a target policy (*e.g.*, Chen et al. (2018);

*Work done as an intern at OMRON SINIC X

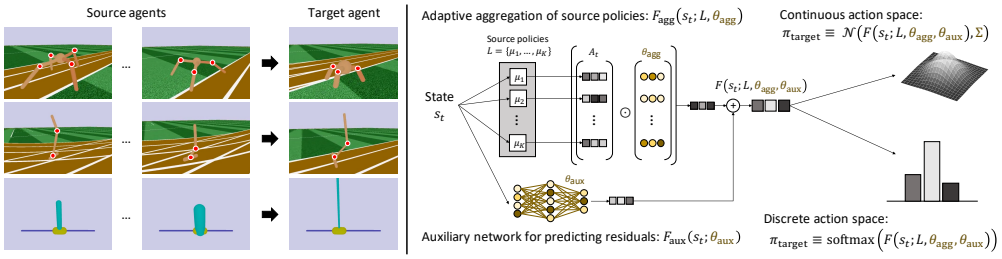

Figure 1: (Left) Examples of environmental dynamics. (Right) Overview of the proposed MULTIPOLAR.

Yu et al. (2019); Tirinzoni et al. (2018)). Similarly, meta-learning approaches (Vanschoren, 2018) cannot be used here because they typically train an agent on a diverse set of tasks (*i.e.*, environment instances). Also, existing techniques that utilize a collection of source policies, *e.g.*, policy reuse frameworks (Rosman et al., 2016; Zheng et al., 2018) and option frameworks (Sutton et al., 1999; Bacon et al., 2017; Mankowitz et al., 2018), are not a promising solution because, to our knowledge, they assume source tasks to have the same environmental dynamics but have different goals.

As a solution to the problem, we propose a new transfer RL approach named **MULTI-source POLicy AggRegation (MULTIPOLAR)**. As shown in Figure 1 (right), our key idea is twofold; 1) In a target policy, we adaptively aggregate the deterministic actions produced by a collection of source policies. By learning aggregation parameters to maximize the expected return at a target environment instance, we can better adapt the aggregated actions to unseen environmental dynamics of the target instance without knowing source environmental dynamics nor source policy performances. 2) We also train an auxiliary network that predicts a residual around the aggregated actions, which is crucial for ensuring the expressiveness of the target policy even when some source policies are not useful. As another advantage, MULTIPOLAR can be used for both continuous and discrete action spaces with few modifications while allowing a target policy to be trained in a principled fashion. In our extensive experimental evaluation, we demonstrated the effectiveness of MULTIPOLAR in a variety of environments ranging from classic control problems to challenging robotics simulations.

2 Multi-Source Policy Aggregation

Problem setting. We formulate our problem under the standard RL framework (Sutton & Barto, 1998), where an agent interacts with its environment modeled by a Markov decision process (MDP). An MDP is represented by the tuple $\mathcal{M} = (\rho_0, \gamma, \mathcal{S}, \mathcal{A}, R, T)$ where ρ_0 is the initial state distribution and γ is a discount factor. At each timestep t, given the current state $s_t \in \mathcal{S}$, the agent executes an action $a_t \in \mathcal{A}$ based on its policy $\pi(a_t \mid s_t; \theta)$ parameterized by θ. Importantly, in this work, we consider both cases of continuous and discrete for action space $\mathcal{A}$. The environment returns a reward $R(s_t, a_t) \in \mathbb{R}$ and transitions to the next state s_{t+1} based on the state transition distribution $T(s_{t+1} \mid s_t, a_t)$. Similar to some prior works on transfer RL (Song et al., 2016; Tirinzoni et al., 2018; Yu et al., 2019), we consider K instances of the same environment which differ only in their state transition dynamics. Namely, we model each environment instance by an indexed MDP: $\mathcal{M}_i = (\rho_0, \gamma, \mathcal{S}, \mathcal{A}, R, T_i)$ where no two state transition distributions $T_i, T_j; i \neq j$ are identical. Unlike the prior works, we assume that *each T_i is unknown when training a target policy*, *i.e.*, agents cannot access the exact form of T_i nor a collection of states sampled from T_i. For each of the K environment instances, we are given a deterministic source policy $\mu_i : \mathcal{S} \to \mathcal{A}$ that only maps states to actions. Each source policy μ_i can be either parameterized (*e.g.*, learned by interacting with its environment instance $\mathcal{M}_i$) or non-parameterized (*e.g.*, heuristically designed by humans). Either way, we assume that *no prior knowledge about the source policies is available for a target agent, such as their representations or original performances, except that they were acquired from a source environment instance $\mathcal{M}_i$ with an unknown T_i*. This is one of the assumptions that make our problem unique from others, such as policy reuse and option frameworks.

Given the set of source policies $L = \{\mu_1, \ldots, \mu_K\}$, our goal is to train a new target agent's policy $\pi_{\text{target}}(a_t \mid s_t; L, \theta)$ in a sample efficient fashion, where the target agent interacts with another environment instance $\mathcal{M}_{\text{target}} = (\rho_0, \mathcal{S}, \mathcal{A}, R, T_{\text{target}})$ and T_{target} is not identical to the source T_i $(i = 1 \ldots, K)$ due to their distinct dynamics.

MULTIPOLAR. Here we present the proposed MULTIPOLAR for the continuous action space, and extend it to the discrete space in the appendix. Let us denote by $a_t^{(i)} = \mu_i(s_t)$ the action predicted deterministically by source policy μ_i given the current state s_t. For the continuous action space, $a_t^{(i)} \in \mathbb{R}^D$ is a D-dimensional real-valued vector representing D actions performed jointly in each timestep. For the collection of source policies L, we derive the matrix of their deterministic actions as $A_t = \left[(a_t^{(1)})^\top, \ldots, (a_t^{(K)})^\top\right] \in \mathbb{R}^{K \times D}$. Our key idea is to aggregate A_t adaptively in an RL loop, *i.e.*, to maximize the expected return. This adaptive aggregation gives us a "baseline" action that could introduce a strong inductive bias in the training of a target policy *even without knowing each source environmental dynamics* T_i. More specifically, we define the adaptive aggregation function $F_{\text{agg}} : \mathcal{S} \to \mathcal{A}$ that produces the baseline action based on the current state s_t as $F_{\text{agg}}(s_t; L, \theta_{\text{agg}}) = \frac{1}{K}\mathbb{1}^K (\theta_{\text{agg}} \odot A_t)$, where $\theta_{\text{agg}} \in \mathbb{R}^{K \times D}$ is a matrix of trainable parameters, $\odot$ is the element-wise multiplication, and $\mathbb{1}^K$ is the all-ones vector of length K. θ_{agg} is neither normalized nor regularized, and can therefore scale each action of each policy independently.

Moreover, we learn auxiliary network $F_{\text{aux}} : \mathcal{S} \to \mathcal{A}$ jointly with F_{agg}, to predict residuals around the aggregated actions. F_{aux} is used to improve the target policy training in two ways. 1) If the aggregated actions from F_{agg} are already useful in the target environment instance, F_{aux} will correct them for a higher expected return. 2) Otherwise, F_{aux} learns the target task while leveraging F_{agg} as a prior to have a guided exploration process. By denoting trainable parameters of F_{aux} as θ_{aux}, the MULTIPOLAR function is: $F(s_t; L, \theta_{\text{agg}}, \theta_{\text{aux}}) = F_{\text{agg}}(s_t; L, \theta_{\text{agg}}) + F_{\text{aux}}(s_t; \theta_{\text{aux}})$.

Target policy π_{target} can be modeled by reparameterizing the MULTIPOLAR function as a Gaussian distribution $\mathcal{N}(F(s_t; L, \theta_{\text{agg}}, \theta_{\text{aux}}), \Sigma)$, where Σ is a covariance matrix estimated based on what the used RL algorithm requires. Since we regard $\mu_i \in L$ as fixed functions mapping states to actions, this Gaussian policy π_{target} is differentiable with respect to θ_{agg} and θ_{aux}, and hence could be trained with any RL algorithm that explicitly updates policy parameters.

3 Experimental Evaluation

We aim to empirically demonstrate the sample efficiency of a target policy trained with MULTIPOLAR. To complete the experiments in a reasonable amount of time, we set the number of source policies to be $K = 4$ unless mentioned otherwise. Moreover, we investigate the factors that affect the performance of MULTIPOLAR. To ensure fair comparisons and reproducibility of experiments, we followed the guidelines introduced by Henderson et al. (2018) and François-Lavet et al. (2018) for conducting and evaluating all of our experiments. To show the benefits of leveraging source policies, we compared our MULTIPOLAR policy to the standard multi-layer perceptron (MLP) trained from scratch, which is typically used in RL literature (Schulman et al., 2017; François-Lavet et al., 2018). As another baseline, we also used MULTIPOLAR with $K = 1$, which is an extension of residual policy learning (Silver et al., 2018; Johannink et al., 2019), denoted by "RPL", with adaptive residuals as well as the ability to deal with both continuous and discrete action spaces. In the Appendix, we describe the evaluation metric and implementational details.

Environments. To show the general effectiveness of the MULTIPOLAR policy, we conducted comparative evaluations of MULTIPOLAR in the following six OpenAI Gym environments: Roboschool Hopper, Roboschool Ant, Roboschool InvertedPendulumSwingUp, Acrobot, CartPole, and LunarLander. We chose these six environments because 1) the parameterization of their dynamics and kinematics is flexible enough, 2) they cover discrete action space (Acrobot and CartPole) as well as continuous action space, and 3) they are samples of three distinct categories of OpenAI Gym environments, namely Box2d, Classic Control, and Roboschool. For each of the six environments, we created 100 environment instances with diverse dynamics and kinematics (described in the Appendix B.1), which we used to evaluate MULTIPOLAR.

Results. Figure 2 and Table 1 clearly show that on average, in all the environments, MULTIPOLAR substantially outperformed baseline policies in terms of sample efficiency and sometimes the final episodic reward[1]. For example, in Hopper over 2M training samples, MULTIPOLAR with

[1]Episodic rewards in Figure 2 are averaged over 3 random seeds and 3 random source policy sets on 100 environment instances. Table 1 reports the mean of this average over training samples.

Table 1: **MULTIPOLAR vs. Baselines**. Bootstrap mean and 95% confidence bounds of average episodic rewards over various training samples.

Methods	CartPole	
	50K	100K
MLP	229 (220,237)	291 (282,300)
RPL	238 (231,245)	289 (282,296)
MULTIPOLAR	**252 (245,260)**	**299 (292,306)**
	Acrobot	
	100K	200K
MLP	-164 (-172,-156)	-111 (-117,-106)
RPL	-120 (-124,-116)	-98 (-101,-95)
MULTIPOLAR	**-117 (-121,-113)**	**-96 (-99,-93)**
	LunarLander	
	250K	500K
MLP	112 (104,121)	216 (210,221)
RPL	178 (174,182)	246 (243,248)
MULTIPOLAR	**181 (177,185)**	**246 (244,249)**
	Roboschool Hopper	
	1M	2M
MLP	43 (42,45)	92 (88,96)
RPL	75 (70,79)	152 (142,160)
MULTIPOLAR	**138 (132,143)**	**283 (273,292)**
	Roboschool Ant	
	1M	2M
MLP	1088 (1030,1146)	1500 (1430,1572)
RPL	1120 (1088,1152)	1432 (1391,1473)
MULTIPOLAR	**1397 (1361,1432)**	**1744 (1705,1783)**
	Roboschool InvertedPendulumSwingup	
	1M	2M
MLP	267 (260,273)	409 (401,417)
RPL	195 (192,198)	322 (317,326)
MULTIPOLAR	**476 (456,495)**	**588 (571,605)**

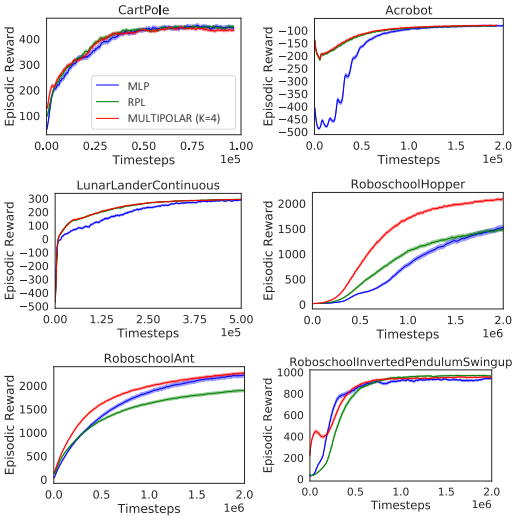

Figure 2: **Average Learning Curves** of MLP, RPL, and MULTIPOLAR over all the experiments for each environment. Shaded areas represent 1 standard error.

Table 2: Results with different source policy sampling in Hopper.

MULTIPOLAR	1M	2M
Random	138 (132,143)	283 (273,292)
4 high performance	**214 (208,220)**	**420 (409,430)**
2 high & low performance	98 (94,102)	208 (200,215)
4 low performance	45 (44,47)	92 (88,95)

$K = 4$ achieved a mean of average episodic reward about three times higher than MLP (*i.e.*, training from scratch) and about twice higher than RPL (*i.e.*, using only a single source policy). It is also noteworthy that MULTIPOLAR had consistently on par or better performance than RPL, which indicates the effectiveness of leveraging multiple source policies.

We also investigate the effect of source policies performances on MULTIPOLAR sample efficiency in the Hopper environment. Figure 3 in Appendix shows that the source policies were diverse in terms of the performance in their original environment instance. We created two separate pools of source policies, where one contained only high-performing and the other only low-performing ones[2]. Table 2 summarizes the results of sampling source policies from these pools (4 high, 2 high & low, and 4 low performances) and compares them to the original MULTIPOLAR (shown as 'Random') also reported in Table 1. Not surprisingly, MULTIPOLAR performed the best when all the source policies were sampled from the high-performance pool. However, we emphasize that such high-quality policies are not always available in practice, due to the variability of how they are learned or hand-crafted under their own environment instance. Finally, Figure 4 in Appendix shows an example where MULTIPOLAR successfully learned to suppress the useless low-performing sources.

Conclusion. We presented a new problem setting of transfer RL, which aims to train a policy efficiently using a collection of source policies acquired under diverse unknown environmental dynamics. Our proposed MULTIPOLAR, achieved a high training sample efficiency in a variety of environments. Future work seeks to involve other types of environmental disparities (*e.g.*, different reward functions and state/action spaces) and to leverage MULTIPOLAR for transferring knowledge in real-world robotics tasks. For further discussions and results, we refer the reader to the Appendix.

[2]Here, policies with final episodic reward over 2K are high-performing and below 1K are low-performing.

REFERENCES

Pierre-Luc Bacon, Jean Harb, and Doina Precup. The Option-Critic Architecture. In *AAAI*, 2017.

Tao Chen, Adithyavairavan Murali, and Abhinav Gupta. Hardware Conditioned Policies for Multi-Robot Transfer Learning. In *NeurIPS*, 2018.

Coline Devin, Abhishek Gupta, Trevor Darrell, Pieter Abbeel, and Sergey Levine. Learning Modular Neural Network Policies for Multi-Task and Multi-Robot Transfer. In *ICRA*, 2017.

Bradley Efron and Robert Tibshirani. *An Introduction to the Bootstrap*. Springer, 1993.

Fernando Fernández and Manuela Veloso. Probabilistic Policy Reuse in a Reinforcement Learning Agent. In *AAMAS*, 2006.

Vincent François-Lavet, Peter Henderson, Riashat Islam, Marc G. Bellemare, and Joelle Pineau. An Introduction to Deep Reinforcement Learning. *Foundations and Trends in Machine Learning*, 11 (3-4):219–354, 2018.

Kevin Frans, Jonathan Ho, Xi Chen, Pieter Abbeel, and John Schulman. Meta Learning Shared Hierarchies. In *ICLR*, 2018.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor. In *ICML*, 2018.

Peter Henderson, Riashat Islam, Philip Bachman, Joelle Pineau, Doina Precup, and David Meger. Deep Reinforcement Learning That Matters. In *AAAI*, 2018.

Ashley Hill, Antonin Raffin, Maximilian Ernestus, Adam Gleave, 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.

Tobias Johannink, Shikhar Bahl, Ashvin Nair, Jianlan Luo, Avinash Kumar, Matthias Loskyll, Juan Aparicio Ojea, Eugen Solowjow, and Sergey Levine. Residual Reinforcement Learning for Robot Control. In *ICRA*, 2019.

Tejas D Kulkarni, Karthik Narasimhan, Ardavan Saeedi, and Josh Tenenbaum. Hierarchical Deep Reinforcement Learning: Integrating Temporal Abstraction and Intrinsic Motivation. In *NeurIPS*, 2016.

Alessandro Lazaric, Marcello Restelli, and Andrea Bonarini. Transfer of Samples in Batch Reinforcement Learning. In *ICML*, 2008.

Siyuan Li and Chongjie Zhang. An Optimal Online Method of Selecting Source Policies for Reinforcement Learning. In *AAAI*, 2018.

Siyuan Li, Fangda Gu, Guangxiang Zhu, and Chongjie Zhang. Context-Aware Policy Reuse. In *AAMAS*, 2019.

Daniel J Mankowitz, Timothy A Mann, Pierre-Luc Bacon, Doina Precup, and Shie Mannor. Learning Robust Options. In *AAAI*, 2018.

Takayuki Osa, Voot Tangkaratt, and Masashi Sugiyama. Hierarchical Reinforcement Learning via Advantage-Weighted Information Maximization. In *ICLR*, 2019.

Benjamin Rosman, Majd Hawasly, and Subramanian Ramamoorthy. Bayesian Policy Reuse. *Machine Learning*, 104(1):99–127, 2016.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal Policy Optimization Algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Tom Silver, Kelsey Allen, Josh Tenenbaum, and Leslie Kaelbling. Residual Policy Learning. *arXiv preprint arXiv:1812.06298*, 2018.

Jinhua Song, Yang Gao, Hao Wang, and Bo An. Measuring the Distance Between Finite Markov Decision Processes. In *AAMAS*, 2016.

Richard S. Sutton and Andrew G. Barto. *Introduction to Reinforcement Learning*. MIT Press, 1st edition, 1998.

Richard S Sutton, Doina Precup, and Satinder Singh. Between MDPs and Semi-MDPs: A Framework for Temporal Abstraction in Reinforcement Learning. *Artificial intelligence*, 112(1-2):181–211, 1999.

Ole Tange. *GNU Parallel 2018*. Ole Tange, 2018. URL https://doi.org/10.5281/zenodo.1146014.

Andrea Tirinzoni, Andrea Sessa, Matteo Pirotta, and Marcello Restelli. Importance Weighted Transfer of Samples in Reinforcement Learning. In *ICML*, 2018.

Joaquin Vanschoren. Meta-Learning: A Survey. *arXiv preprint arXiv:1810.03548*, 2018.

Hao Wang, Shaokang Dong, and Ling Shao. Measuring Structural Similarities in Finite MDPs. In *IJCAI*, 2019.

Wenhao Yu, C. Karen Liu, and Greg Turk. Policy Transfer with Strategy Optimization. In *ICLR*, 2019.

Yan Zheng, Zhaopeng Meng, Jianye Hao, Zongzhang Zhang, Tianpei Yang, and Changjie Fan. A Deep Bayesian Policy Reuse Approach against Non-Stationary Agents. In *NeurIPS*, 2018.

Table 3: Sampling range for Ant kinematic and dynamic parameters.

	Kinematics
Links	Length Range
Legs	[0.4, 1.4] $\times$ default length
	Dynamics
Damping	[0.1, 5]
Friction	[0.4, 2.5]
Armature	[0.25, 3]
Links mass	[0.7, 1.1] $\times$ default mass

Table 4: Sampling range for CartPole kinematic and dynamic parameters.

	Kinematics
Links	Length Range (m)
Pole	[0.1, 3]
	Dynamics
Force	[6, 13]
Gravity	[-14, -6]
Poll mass	[0.1, 3]
Cart mass	[0.3, 4]

ACKNOWLEDGMENTS

The authors would like to thank Robert Lee, Felix von Drigalski, Yoshihisa Ijiri, Tatsunori Taniai, and Daniel Plop, for the insightful discussions and helpful feedback on the manuscript.

A MULTIPOLAR FOR DISCRETE ACTION SPACES

We can formulate the target policy of MULTIPOLAR in a principled fashion for actions in a discrete space. Specifically, instead of a D-dimensional real-valued vector, here we have a D-dimensional one-hot vector $a_t^{(i)} \in \{0,1\}^D,\ \sum_j (a_t^{(i)})_j = 1$ as outputs of μ_i, where $(a_t^{(i)})_j = 1$ indicates that the j-th action is to be executed. Following Eq ($F(s_t; L, \theta_{\text{agg}}, \theta_{\text{aux}}) = F_{\text{agg}}(s_t; L, \theta_{\text{agg}}) + F_{\text{aux}}(s_t; \theta_{\text{aux}})$), the output of $F(s_t; L, \theta_{\text{agg}}, \theta_{\text{aux}})$ can be viewed as D-dimensional un-normalized action scores, from which we can sample a discrete action after normalizing it by the softmax function.

B EXPERIMENTAL DETAILS

B.1 EXPERIMENTAL PROCEDURE

For each of the six environments, we first created 100 environment instances by randomly sampling the dynamics and kinematics parameters from a specific range shown in Tables 3, 4, 5, 6, 7 and 8. For example, Table 5 provides the sampling ranges for Hopper environment parameters defined similar to Chen et al. (2018). Note that we defined the sampling ranges for each environment such that the resulting environment instances involve significantly different dynamics. Also, these parameters were used only for simulating environment instances and were not available when training a target policy.

Then, for each environment instance, we trained an MLP policy that was used in two ways: a) the baseline MLP policy for each environment instance, and b) one of the 100 members of the source policy candidate pool from which we sample K of them to train MULTIPOLAR policies and one of them to train RPL policies[3]. Specifically, for each environment instance, we trained three MULTIPOLAR and three RPL policies with distinct sets of source policies selected randomly from the candidate pool. The learning procedure explained above was done three times with fixed different random seeds to reduce variance in results due to stochasticity. As a result, for each of the six environments, we had 100 environment instances $\times$ 3 random seeds = 300 experiments for MLP and 100 environment instances $\times$ 3 choices of source policies $\times$ 3 random seeds = 900 experiments for RPL and MULTIPOLAR. The aim of this large number of experiments is to obtain correct insights into the distribution of performances (Henderson et al., 2018). Due to the large number of experiments for all the environments, our detailed analysis and ablation study of MULTIPOLAR

[3]Although we used trained MLPs as source policies for reducing experiment times, any type of policies including hand-engineered ones could be used for MULTIPOLAR in principle.

Table 5: Sampling range for Hopper kinematic and dynamic parameters.

	Kinematics
Links	Length Range (m)
Leg	[0.35, 0.65]
Foot	[0.29, 0.49]
Thigh	[0.35,0.55]
Torso	[0.3,0.5]
	Dynamics
Damping	[0.5, 4]
Friction	[0.5, 2]
Armature	[0.5, 2]
Links mass	[0.7, 1.1] $\times$ default mass

Table 6: Sampling range for InvertedPendulum-Swingup kinematic and dynamic parameters.

	Kinematics
Links	Length Range (m)
Pole	[0.2, 2]
	Dynamics
Damping	[0.1, 5]
Friction	[0.5, 2]
Armature	[0.5, 3]
Gravity	[-11, -7]
Links mass	[0.4, 3] $\times$ default mass

Table 7: Sampling range for Acrobot kinematic and dynamic parameters.

	Kinematics
Links	Length Range (m)
Link 1&2	[0.3, 1.3]
	Dynamics
Links mass	[0.5, 1.5]
Links center mass	[0.05, 0.95] $\times$ link length
Links inertia moments	[0.25, 1.5]

Table 8: Sampling range for LunarLander kinematic and dynamic parameters.

	Kinematics
Side engine height	[10, 20]
	Dynamics
Scale	[25, 50]
Initial Random	[500, 1500]
Main engine power	[10, 40]
Side engine power	[0.5, 2]
Side engine away	[8, 18]

components were conducted with only Hopper, as its sophisticated second-order dynamics plays a crucial role in agent performance (Chen et al., 2018).

B.2 Source Policies Histograms

To generate environment instances, we uniformly sampled the dynamics and kinematics parameters from the ranges defined in Section B.1. Figure 3 illustrates the histograms of the final episodic rewards of source policies on the original environment instances in which they were acquired.

C Evaluation Metric

Following the guidelines of (Henderson et al., 2018), to measure sampling efficiency of training policies, *i.e.*, how quick the training progresses, we used the average episodic reward over training samples. Also, to ensure that higher average episodic reward is representative of better performance and to estimate the variation of it, we used the sample bootstrap method to estimate statistically relevant 95% confidence bounds of the results of our experiments. Across all the experiments, we used 10K bootstrap iterations and the pivotal method.

Specifically, we calculated the mean of average episodic rewards in Tables 1, 2, 11, and 12, over a specific number of training samples (the numbers at the header of the tables *e.g.*, 50K and 100K for the CartPole) which we denote by T, as follows. For each experiment in an environment instance, we computed the average episodic reward by taking the average of the rewards over all the episodes the agent played from the beginning of the training until collecting T number of training samples. Then we collected the computed average episodic rewards of all the experiments, *i.e.*, all the combinations of three random seeds, three random sets of source policies (for RPL

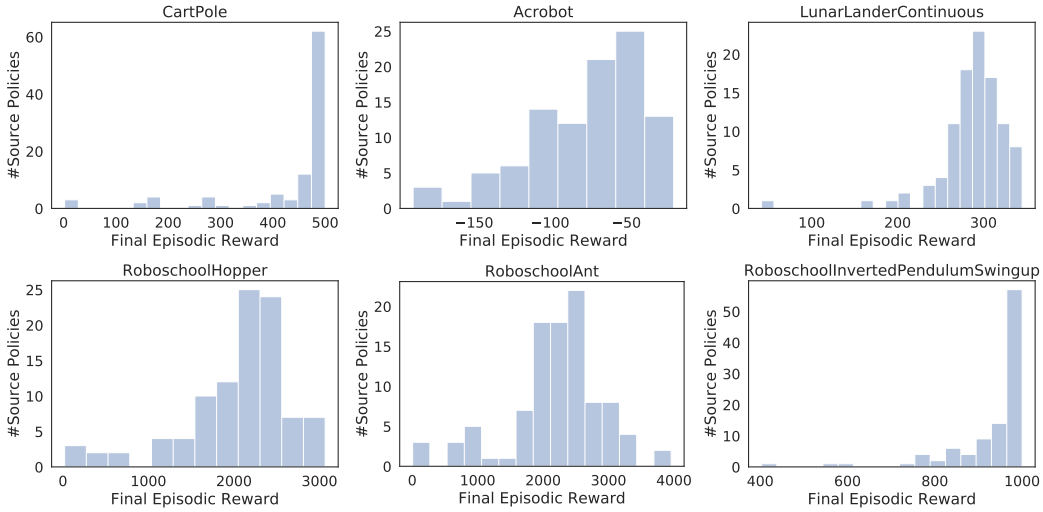

Figure 3: Histogram of final episodic rewards obtained by source policies per environment.

and MULTIPOLAR), and 100 target environment instances. Finally, we used the sample bootstrap method (Efron & Tibshirani, 1993) to estimate the mean and the 95% confidence bounds of the collected average episodic rewards. We used the Facebook Boostrapped implementation: https://github.com/facebookincubator/bootstrapped.

D Implementation Details

All the experiments were done using the Stable Baselines (Hill et al., 2018) implementation of learning algorithms as well as its default hyperparameters and MLP network architecture for each environment. Based on the performance of learning algorithms reported in (Hill et al., 2018), all the policies were trained with Soft Actor-Critic (Haarnoja et al., 2018) in the LunarLander environment and with Proximal Policy Optimization (Schulman et al., 2017) in the rest of the environments. For fair comparisons, in all experiments, auxiliary network F_{aux} had an identical architecture to that of the MLP. Therefore, the only difference between MLP and MULTIPOLAR was the aggregation part F_{agg}, which made it possible to evaluate the contribution of transfer learning based on adaptive aggregation of source policies. Also, we avoided any random seed optimization since it has been shown to alter the policies' performance (Henderson et al., 2018). Note that we did not do any hyperparameter-tuning but followed the default parameters of Hill et al. (2018). We used the Roboschool implementation of Hopper, Ant, and InvertedPendulumSwingup since they are based on an open-source engine, which makes it possible for every researcher to reproduce our experiments. To run our experiments in parallel, we used GNU Parallel tool (Tange, 2018).

Tables 9 and 10 summarize all the hyperparameters used for experiments on each environment. As done by Hill et al. (2018), to have a successful training, rewards and input observations are normalized using their running average and standard deviation for all the environments except CartPole and LunarLander. Also, in all of the experiments, θ_{agg} is initialized to be the all-ones matrix.

E Additional Results

E.1 Ablation Study

To demonstrate the importance of each component of MULTIPOLAR, we evaluated the following degraded versions: (1) θ_{agg} *fixed to 1*, which just averages the deterministic actions from the source policies without adaptive weights (similar to the residual policy learning methods that use raw action outputs of a source policy), and (2) F_{aux} *learned independent of* s_t, which replaces the state-dependent MLP with an adaptive "placeholder" parameter vector making actions a linear combination of source policy outputs. As shown in Table 11, the full version of MULTIPOLAR

Table 9: Hyperparameters for Acrobot, CartPole, Hopper, Ant and InvertedPendulumSwingup.

PPO Parameters	Acrobot	CartPole	Hopper	Ant	InvertedPendulumSwingup
#Training samples	200K	100K	2M	2M	2M
#Updates per rollout	4	20	10	10	10
Learning rate	2.5e-4	1e-3	2.5e-4	2.5e-4	2.5e-4
Mini batch size	8	1	128	32	32
Discount factor	0.99	0.98	0.99	0.99	0.99
GAE λ	0.94	0.8	0.95	0.95	0.95
Clip ratio	0.2	0.2	0.2	0.2	0.2
Value function coefficient	0.5	0.5	0.5	0.5	0.5
Entropy coefficient	0	0	0	0	0
Gradient clipping value	0.5	0.5	0.5	0.5	0.5
Optimizer	Adam	Adam	Adam	Adam	Adam
MLP & F_{aux} Parameters					
Hidden layers	64-64	64-64	64-64	16	64-64
Activation functions	tanh	tanh	tanh	tanh	tanh

Table 10: Hyperparameters for LunarLander.

SAC Parameters	LunarLander
#Training samples	500K
#Steps before learning starts	1K
Buffer size	50K
Learning rate	3e-4
Mini batch size	256
Discount factor	0.99
Soft update coefficient τ	5e-3
Entropy coefficient	learned automatically
Model training frequency	1
Target network training frequency	1
#Gradient updates after each step	1
Probability of taking a random action	0
Action noise	none
Optimizer	Adam
MLP & F_{aux} Parameters	
Hidden layers	64-64
Activation functions	relu

significantly outperformed the degraded ones, suggesting that adaptive aggregation and predicting residuals are both critical.

E.2 Effect of Number of Source Policies.

Finally, we show how the number of source policies contributes to MULTIPOLAR's sample efficiency in Table 12. Specifically, we trained MULTIPOLAR policies up to $K = 16$ to study how the mean of average episodic rewards changes. The monotonic performance improvement over K (for $K \leq 16$), is achieved at the cost of increased training and inference time. In practice, we suggest balancing this speed-performance trade-off by using as many source policies as possible before reaching the inference time limit required by the application.

Table 11: Ablation study in Hopper.

MULTIPOLAR ($K = 4$)	1M	2M
Full version	**138 (132,143)**	**283 (273,292)**
θ_{agg} fixed to 1	118 (111,126)	237 (222,250)
F_{aux} learned independent of s_t	101 (95,108)	187 (175,200)

Table 12: Results with different number of source policies in Hopper.

MULTIPOLAR	1M	2M
$K = 4$	138 (132,143)	283 (273,292)
$K = 8$	160 (154,167)	323 (312,335)
$K = 16$	**177 (172,182)**	**357 (348,367)**

E.3 LEARNED AGGREGATION PARAMETERS VISUALIZATION

Figure 4 visualizes an example of how the aggregation parameters θ_{agg} for the four policies and their three actions were learned during the 2M timestep training of MULTIPOLAR ($K = 4$) policy in the Hopper environment. In this example, the source policies in the first and second rows were sampled from low-performance pools whereas those in the third and fourth rows were sampled from high-performance pools. It illustrates that MULTIPOLAR can successfully suppress the two useless low-performing policies as the training progresses.

F RELATED WORK

Transfer between Different Dynamics. Our work is broadly categorized as an instance of transfer RL between different environmental dynamics, in which a policy for a target task is trained using information collected from source tasks. Much related work requires training samples collected from source tasks, which are then used for measuring the similarity between source and target environment instances (Lazaric et al., 2008; Tirinzoni et al., 2018) or for conditioning a target policy to predict actions (Chen et al., 2018). Alternative means to quantify the similarity is to use a full specification of MDPs (Song et al., 2016; Wang et al., 2019) or environmental dynamics (Yu et al., 2019). In contrast, the proposed MULTIPOLAR allows the knowledge transfer only through the policies acquired from source environment instances with diverse unknown dynamics, which is beneficial when source and target environments are not always connected to exchange information about their dynamics and training samples.

Leveraging Multiple Policies. The idea of utilizing multiple source policies can be found in the literature of policy reuse frameworks (Fernández & Veloso, 2006; Rosman et al., 2016; Li & Zhang, 2018; Zheng et al., 2018; Li et al., 2019). The basic motivation behind these works is to provide "nearly-optimal solutions" (Rosman et al., 2016) for short-duration tasks by reusing one of the source policies, where each source would perform well on environment instances with different rewards (*e.g.*, different goals in maze tasks). In our problem setting, where environmental dynamics behind each source policy are different, reusing a single policy without an adaptation is not the right approach, as described in (Chen et al., 2018) and also demonstrated in our experiment. Another relevant idea is hierarchical RL (Kulkarni et al., 2016; Osa et al., 2019) that involves a hierarchy of policies (or action-value functions) to enable temporal abstraction. In particular, option frameworks (Sutton et al., 1999; Bacon et al., 2017; Mankowitz et al., 2018) make use of a collection of policies as a part of "options". However, they assumed all the policies in the hierarchy to be learned in a single environment instance. Another relevant work along this line of research is (Frans et al., 2018), which meta-learns a hierarchy of multiple sub-policies by training a master policy over the distribution of tasks. Nevertheless, hierarchical RL approaches are not useful for leveraging multiple source policies each acquired under diverse environmental dynamics.

Learning Residuals in RL. Some recent works adopt residual learning to mitigate the limited performance of hand-engineered policies (Silver et al., 2018; Johannink et al., 2019). We are interested in a more extended scenario where various source policies with unknown performances are

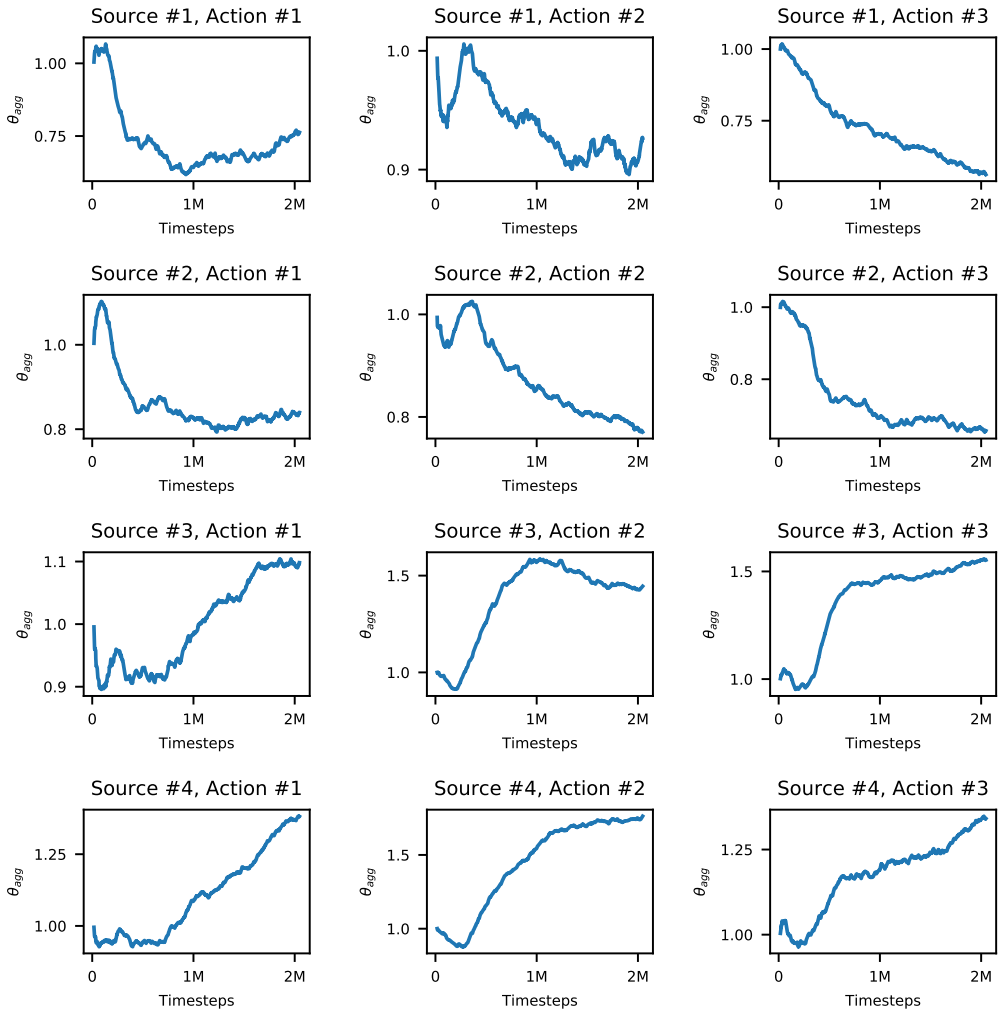

Figure 4: Aggregation parameters θ_{agg} during the training of MULTIPOLAR ($K = 4$) in the Hopper that has 3-dimensional actions. Here, the first two source policies are low-performing and the last two are high-performing in their original environment instance.

provided instead of a single sub-optimal policy. Also, these approaches focus only on robotic tasks in the continuous action space, while our approach could work on both of continuous and discrete action spaces in a broad range of environments.