Robots are Exploring to Learn From Each Other: Multi-Robot Multi-Task Transfer

Abstract

Motivation: In this project, we explored a multi-robot, multi-task transfer framework with the objective of scalability across various tasks and robots. This framework is designed to be adaptable to different dynamics and reward structures, emphasizing data efficiency and support for target robots and tasks, especially those with limited resources. This lies in the observation that similar agents often share structural similarities, noise profiles, and challenges. Therefore, enabling robots to learn from each other could prove more effective than starting from scratch. This approach aligns with the initial human fascination with robotics: the aspiration to create machines capable of performing tasks in a human-like manner. An ideal robot would be able to observe human actions and learn directly from them. However, the differing dynamics and morphologies between humans and robots introduce significant noise in action learning and transfer. Understanding how to learn from these differences is crucial, particularly for offline learning and learning from videos. With humans exhibiting varied dynamics, it becomes essential to consider these variations in our learning models. Furthermore, we explore the potential of optimizing robot dynamics. Can we demonstrate that learning from robots is more effective than learning from other sources? A critical question we address is how to effectively train a low-resource robot alongside a high-resource robot in mastering a new task, considering their differing dynamics and morphologies.

Research Question: Our primary research challenge involves a scenario where we possess a well-resourced source robot trained on a specific task, contrasted with a low-resource robot characterized by distinct morphology and dynamics. Additionally, we have a different target task, which also has limited resources but differs in objectives and reward structures. Our goal is to propose a transfer learning method that facilitates efficient learning between these disparate entities with minimal interaction. This method should enable the target robot to learn the source task and vice versa. We also explore various approaches to address this problem, aiming to enhance the effectiveness and efficiency of transfer learning in robotics, especially under resource constraints.

Method: Our approach employs reward compensation as a key strategy for facilitating dynamics transfer in multi-robot, multi-task scenarios. This same principle of reward compensation is also applied to task transfer. Essentially, our method utilizes reward compensation as a universal tool for both dynamics and task transfers in a multi-robot context. Specifically, we adopt the method outlined in [4] for dynamic transfer, leveraging the differences in rewards as a basis for reward compensation during task transfer.

Novelty: Distinct Approach: Our setup introduces a significantly unique approach. Our literature review indicates that none of the previous works have implemented a setup like ours, distinguishing it from other approaches by emphasizing a more data-efficient robot interaction.

Reward Compensation for Multi-Task, Multi-Robot Transfer: We introduce reward compensation as a key strategy in our multi-task, multi-robot transfer framework.

Enhanced Exploration through Reward Compensation: We incorporate exploration strategies based on Reward Compensation to improve convergence. This is based on the principle that higher reward compensation indicates a greater need for exploration.

Mutual Training of Source and Target Agents (Co-Training): Our method involves co-training source and target agents, utilizing reward compensation as the basis for this mutual learning process.

Versatile Structure for Efficient Transfer Learning: We propose various structural models that are supported by reward compensation, aiming to enhance efficiency in multi-robot, multi-task transfer learning.

Focusing on Low-Resource and High-Resource Robotics: Our approach takes into account the disparities in resource availability, catering to both low-resource and high-resource robotics scenarios.

Experiments: In our experiments, we focus on reacher tasks involving arms with varying morphologies (differing in size and density) in both source and target scenarios. These tasks are further diversified by implementing distinct goals and reward functions for the source and target. Benchmark methods, employing RL algorithms, are applied in both the source and target settings. We have conducted several experiments to assess and analyze the effectiveness of our proposed method through comprehensive ablation studies.

Results and analysis: Overall, our analysis of various approaches indicated that both source and target training achieved satisfactory coconvergence. Initially, I anticipated that incorporating an exploration component would lead to faster convergence. However, this exploration actually resulted in greater variability in the rewards, suggesting that less exploration might be more suitable for the current tasks and for more complicated tasks this exploration is more helpful. Nonetheless, the system ultimately approached convergence, implying that with extended training, results could further improve. Additionally, cotraining initially introduced high variability, but this stabilized towards the end, yielding good outcomes. This is because when we engage in the training procedure, we achieve a better estimate of reward compensation, which is a main info sharing in the cotraining.

1 Introduction to Approach

The overall approach of the multi-robot multi-task method is depicted in Figure 1. We have a target world, which is a low-resource world, and are interested in training the target task on the target agents using a high-resource world. To achieve this, we are utilizing reward compensation based on action_reward[4] and task_reward compensation

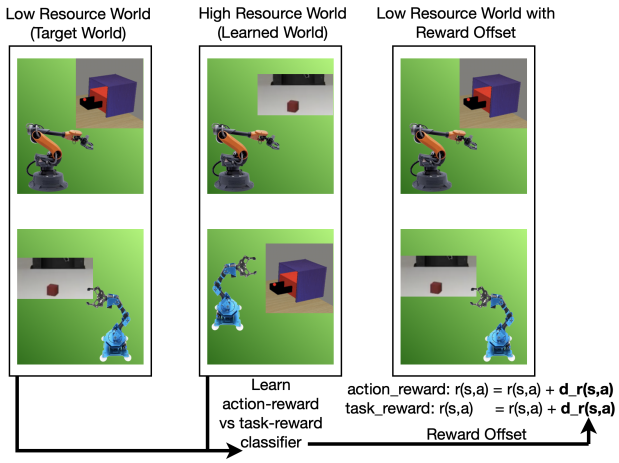

Figure 1: Overall approach

This approach can be extended to multi-task learning for a low_resource target robot, as shown in Figure 2

Figure 2: low_resource robot learning in multi-task learning

Additionally, this Low_resource transfer learning can be utilized for learning from human demonstrations, as illustrated in Figure 3.

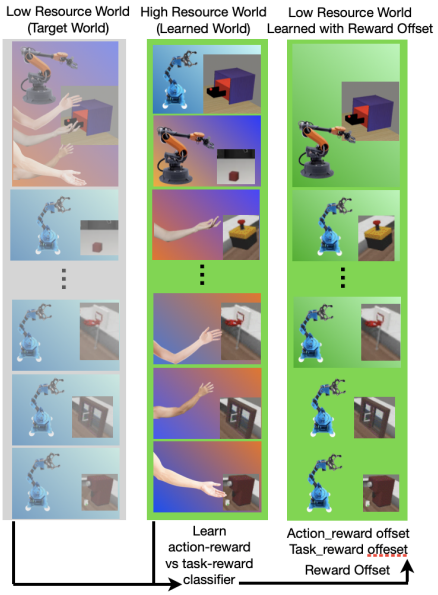

Figure 3: Low_resource learning from human demonstration.

2 Related Work

Bousmalis et al.'s RoboCat [1] is a decision transformer for mastering new skills and embodiments in robotics. Dasari et al.'s 'RoboNet' [2] fuses a substantial dataset from 7 robots with visual foresight and supervised inverse models, showcasing superior adaptability to new tasks and robots. Devin et al. 's method [3] splits neural network policies into 'task-specific' and 'robot-specific' modules, facilitating transfer learning across various robots and tasks. Eysenbach et al. [4] propose a method for domain adaptation in reinforcement learning, which uses a modified reward function learned from auxiliary classifiers and has been proven to recover a near-optimal policy for the target domain. Open X-Embodiment [5] introduces RT-X, a high-capacity model trained on data from 22 robots, enhancing multiple robots' capabilities through positive transfer and mutual learning, marking a step towards seamless task adaptation in robotics.

3 Methodology

3.1 Problem Statement

Robot $R1$ has learned policy π_1 for Task $T1$, and robot $R2$ has learned policy π_2 for Task $T2$. $R1$ and $R2$ have different dynamics, and $T1$ and $T2$ have different reward functions. Our goal is for $R1$ to learn policy $T2$ with a few interactions and for $R2$ to learn $T1$ with a few interactions (zero-shot, one-shot, few-shot) as well. (Since $R2$ knows $T2$ and $R1$ knows $T1$, this task is an extension of in-context learning.) Another suggested transfer approach is illustrated in Figure 5, and a different approach is depicted in Figure 6. The previous approach requires another framework, Online Robot Learning

Transfer, in a simpler case, as shown in Figure 4. These three methods can occur because of separate action and task transfer by reward compensation.

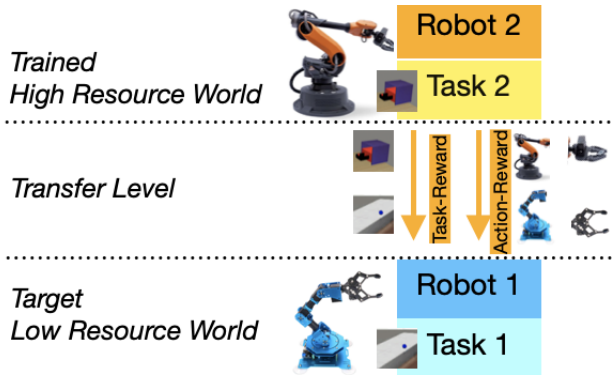

Figure 4: Single Transfer: Two robots learning from each other: Transfer learning for multi-robot, multi-task scenarios involving two robots and two tasks.

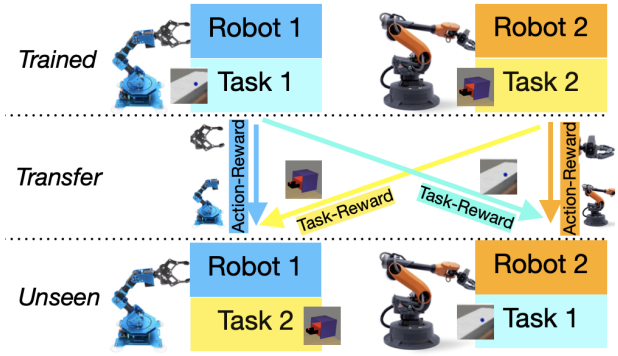

Figure 5: Type_1 of Transfer

3.2 Method overview

You can check the method overview for the setup of transfer1 in Figure 7. A similar approach applies to transfer2 as well, which is explained in the algorithm.

3.3 Possibility and Analysis

The initial reason that why the methodology work is that is we can use reward editing for having a same dynamics but different tasks (task transfer), and also we can use reward editing for learning a same task for different dynamics (dynamics transfer). So, apparently we can can use reward editing when we have two different dynamics and each know different tasks. and these two can learn from each other. Simple sequence of this transfer can help. The point is that how to use efficient exploration to have the efficient training of the agent in online one. The point is that we limit the exploration based on the reward editing formulation that we learned from $\pi_{1,1}$ and $\pi_{2,2}$. Because we believe that the state_action space

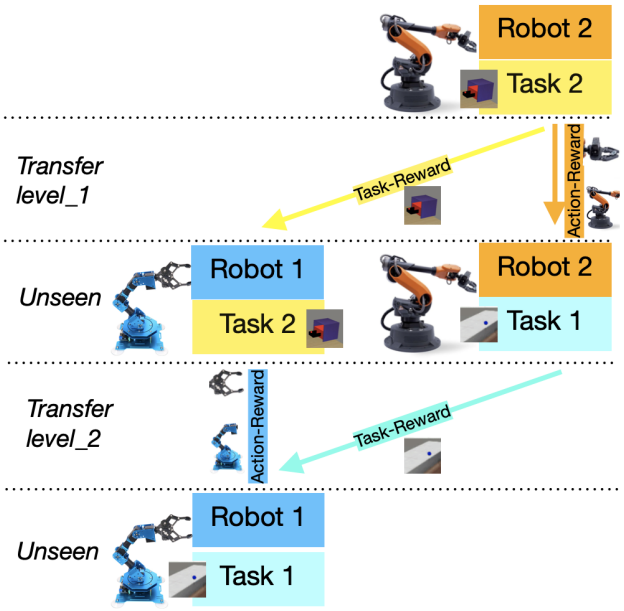

Figure 6: Type_2 of Transfer

is well explored with the $\pi_{1,1}$ and $\pi_{2,2}$ (we can have a random exploration on the other side as well for the state_action that never seen before and based on how much it improves the result, changing the θ which shows how much out of state_action exploration needed).

3.4 Algorithm

We aim to cover three types of transfer for multi-robot multi-agent systems, as depicted in Figures 4, 5, and 6, using reward compensation. For dynamics transfer, we are utilizing reward compensation from [4] as action_reward compensation, and for task_reward, we are using δ_r for each action_pair separately in each transfer.

Subsequently, we can introduce the following algorithms for multi_robot multi_task transfer: 1, 2, and 3. Considering that reward compensation indicates how far the target action state space is from the source action state space, we use reward compensation as an exploration parameter. Moreover, training is conducted based on the reward compensation learned from the target to update the source policy.

4 Experiments and Results

The experimental setup is as follows: The task involves a reacher task featuring different reward functions, varying robot dynamics and morphology, as well as different densities. It includes two distinct arms, each with unique dynamics, goal locations, and reward formulas. One arm uses the Euclidean distance for reward calculation, while the other employs the Manhattan distance.

To examine various ablation studies related to our distinct elements of reward compensation in the context of

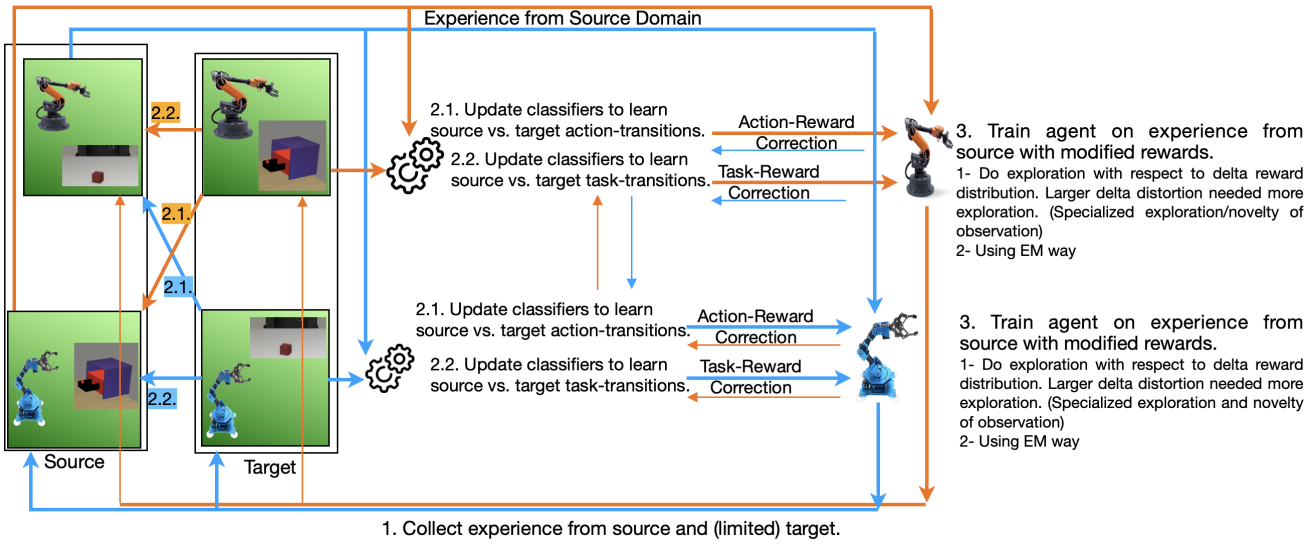

Figure 7: Two robots learning from each other: Transfer learning for multi-robot, multi-task scenarios involving two robots and two tasks.

Algorithm 1 Multi-Robot/Multi-Task(Multi-Reward) Transfer

Input: source MDP M_{source}, target MDP M_{target}, ratio r of experience from source vs. target
Initialize: replay buffers D_{source}, D_{target}, policy π, parameters $\theta = (\theta_{\text{SAS}}, \theta_{\text{SA}})$
for $t = 1, \ldots,$ num iterations **do**
  $D_{\text{source}} \leftarrow D_{\text{source}} \cup \text{ROLLOUT}(\pi, M_{\text{source}})$ ▷ Collect source data
  if $t \bmod r = 0$ **then**
    $D_{\text{target}} \leftarrow D_{\text{target}} \cup \text{ROLLOUT}(\pi, M_{\text{target}})$ ▷ Collect target data
  end if
  $\theta \leftarrow \theta - \eta \nabla_\theta l(\theta)$ ▷ Update parameters
  $\tilde{r}(s_t, a_t, s_{t+1}) \leftarrow r(s_t, a_t) + \Delta r(s_t, a_t, s_{t+1})$ ▷ Compute reward compensation (action-reward transfer or task-reward transfer)
  $\pi \leftarrow \text{MAXENT RL}(\pi, D_{\text{source}}, \tilde{r})$ ▷ Update policy
end for
return π

transfer learning, we consider the following setup for the experiments: (In all the experiments, RL on the source and target is considered as the benchmark for each target or source policies, applied separately to the source and target worlds and computed and illustrated in all the diagrams for reference.).

- $V1$: Arm_1 as a source agent and Arm_2 as a target agent, considering the same reward function as Arm_1. (Different dynamics, same reward function)
- $V2$: Arm_1 as a source agent and Arm_1 as a target agent, considering the same reward function as Arm_2. (Different dynamics, same reward function) For these $V1$ and $V2$ experiments, refer to plot 4.
- $V3$: The dynamics of the source and target are the same as in Arm_1, but the reward functions differ; one is Euclidean and the other is Manhattan.
- $V4$: The dynamics of the source and target are the same as in Arm_2, but the reward functions differ (one is Euclidean and the other is Manhattan). For these $V3$ and $V4$ experiments, refer to plot 9.
- $V5 - a$: In this experiment, Arm_1 runs on $Task_1$ and transfers to Arm_2 on $Task_2$, using the approach of Algorithm 1.
- $V5 - b$: Arm_2 on $Task_2$ transfers to Arm_1 on $Task_1$, using the approach of Algorithm 1. For these $V5 - a$ and $V5 - b$ experiments, refer to plot 10.
- $V6$ (reward compensation based exploration with SAC training): Using δr with reward compensation exploration. For the $V6$ experiments, refer to plot 12.
- $V7$: Experiments using the $Transfer_1$ approach and co-training.

Algorithm 2 Single Multi-Robot Multi-Task Transfer

Input: source MDP M_{source}, target MDP M_{target}, ratio r_{target} of experience from source vs. target, ratio r_{co} of source training, ratio r_{co} of source target co_training
Initialize: replay buffers D_{source}, D_{target}, policy π_{source}, policy $\pi_{target} = \pi_{source}$,
parameters $\theta_{st} = ((\theta_{\text{SAS\_action/dynamics}}, \theta_{\text{SA\_action/dynamics}}), (\theta_{\text{SAS\_task}}, \theta_{\text{SA\_task}}))$
parameters $\theta_{ts} = ((\theta_{\text{SAS\_action/dynamics}}, \theta_{\text{SA\_action/dynamics}}), (\theta_{\text{SAS\_task}}, \theta_{\text{SA\_task}}))$
for $t = 1, \ldots,$ num_iterations **do**
  $D_{\text{source}} \leftarrow D_{\text{source}} \cup \text{ROLLOUT}(\pi, M_{\text{source}})$ ▷ Collect source data
  if $t \bmod r_{target} = 0$ **then**
    $D_{\text{target}} \leftarrow D_{\text{target}} \cup \text{ROLLOUT}(\pi, M_{\text{target}})$ ▷ Collect target data
    $\theta\_st \leftarrow \theta\_st - \eta \nabla_{\theta\_st} l(\theta\_st)$ ▷ Update parameters
    $\tilde{r}(s_t, a_t, s_{t+1}) \leftarrow r(s_t, a_t) + \alpha.\Delta r_{action/dynamics}(s_t, a_t, s_{t+1}) + (1-\alpha).\Delta \tilde{r}_{action/dynamics}(s_t, a_t, s_{t+1})$ ▷ Compute reward
    $\pi_{target} \leftarrow \text{MAXENT RL}(\pi_{target}, D_{\text{target}}, \tilde{r}, \Delta r_{task}, \Delta r_{action})$ ▷ Update policy by Δr exploration in entropy
  end if
  $D_{\text{source}} \leftarrow D_{\text{source}} \cup \text{ROLLOUT}(\pi, M_{\text{source}})$ ▷ Collect source data
  $\theta\_ts \leftarrow \theta\_ts - \eta \nabla_{\theta\_ts} l(\theta\_ts)$ ▷ Update parameters
  if $t \bmod r_{co} = 0$ **then**
    $\tilde{r}(s_t, a_t, s_{t+1}) \leftarrow r(s_t, a_t) + \beta.\Delta r_{action/dynamics}(s_t, a_t, s_{t+1}) + (1-\beta).\Delta \tilde{r}_{action/dynamics}(s_t, a_t, s_{t+1})$
    ▷ Update policy by training on reward compensation based on the source
  else
    $\tilde{r}(s_t, a_t, s_{t+1}) \leftarrow r(s_t, a_t) + \Delta r_{action/dynamics}(s_t, a_t, s_{t+1})$ ▷ Update policy by training on reward compensation based on the source
  end if
  $\pi_{source} \leftarrow \text{MAXENT RL}(\pi_{source}, D_{\text{source}}, \tilde{r}, \Delta r_{task}, \Delta r_{action})$ ▷ Update policy by Δr exploration in entropy
end for
return π

- $V8$: Experiments using the $Transfer_2$ approach and co-training. For the $V7$ and $V8$ experiments, refer to plot 13.

- $V9$: Experiments using the $Transfer_1$ approach and co-training with reward compensation exploration. and $V10$: Experiments using the $Transfer_2$ approach and co-training with reward compensation exploration. For the $V9$ and $V10$ experiments, refer to plot 14.

Also, here is the explanation of the diagram's legends for plots: In all diagram legends, the setup vi is explained as above. The tag $\_low\_r$ indicates that the agent is in a low-resource case, meaning that there are limited interactions. The high_r case implies that the target agent is considered to be in a high-resource scenario, allowing for one-to-one updates in the transfer learning. The tag "_algo" signifies that the results are from running the main algorithm of the study in the corresponding experiments. The tag "_target" indicates that, instead of using the algorithm of the study, the Reinforcement Learning (RL) algorithm (in this case, SAC) is run as a benchmark on the target, using reward compensation for exploration in transformation (if the source and target are different). The tag "_source" implies that, similarly, the RL algorithm (SAC) is run as a benchmark on the target, but with the focus on reward compensation for exploration in transformation (if the source and target differ).

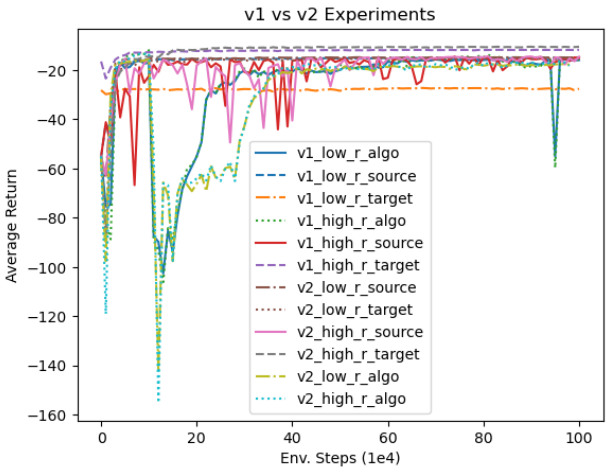

Figure 8: $V1$: Arm_1 as a source agent and Arm_2 as a target agent, considering the same reward function as Arm_1. $V2$: Arm_1 as a source agent and Arm_1 as a target agent, considering the same reward function as Arm_2. (Different dynamics, same reward function)

5 Discussion and Analysis

- In the V1 and V2 experiments, as shown in Plot 4 ('Different Dynamics, Same Reward Function'), it is evident that v1_low_r_algo and v1_high_r_algo exhibit sim-

Algorithm 3 Multiple Multi-Robot Multi-Task Transfer

Input: source MDP M_{11} and M_{22}, target MDP M_{12} and M_{21}, ratio r_{target} of experience from source vs. target, ratio r_{co} of source training, ratio r_{co} of source target co_training
Initialize: replay buffers D_{22}, D_{11}, D_{12}, D_{21} policy π_{11}, π_{22}, π_{12}, π_{21}.
parameters $\theta_{22}, \theta_{11}, \theta_{21}, \theta_{12}$
parameters $\theta_{21} = ((\theta_{\text{SAS\_action/dynamics}}, \theta_{\text{SA\_action/dynamics}})_{source2}, (\theta_{\text{SAS\_task}}, \theta_{\text{SA\_task}})_{source1})$
parameters $\theta_{12} = ((\theta_{\text{SAS\_action/dynamics}}, \theta_{\text{SA\_action/dynamics}})_{source1}, (\theta_{\text{SAS\_task}}, \theta_{\text{SA\_task}})_{source2})$ ▷ The parameters for action_reward and task_reward could be shared across transfers of the same type.
for $t = 1, \ldots,$ num_iterations **do**
    $D_{11} \leftarrow D_{11} \cup \text{ROLLOUT}(\pi, M_{11})$ ▷ Collect source data
    $D_{22} \leftarrow D_{22} \cup \text{ROLLOUT}(\pi, M_{22})$ ▷ Collect source data
    if $t \bmod r_{11} = 0$ **then**
        $D_{12} \leftarrow D_{12} \cup \text{ROLLOUT}(\pi, M_{12})$ ▷ Collect target data
        $D_{21} \leftarrow D_{21} \cup \text{ROLLOUT}(\pi, M_{21})$ ▷ Collect target data
        $\theta\_11 \leftarrow \theta\_11 - \eta \nabla_{\theta\_11} l(\theta\_11)$ ▷ Update parameters
        $\theta\_12 \leftarrow \theta\_12 - \eta \nabla_{\theta\_12} l(\theta\_12)$ ▷ Update parameters
        $\theta\_22 \leftarrow \theta\_22 - \eta \nabla_{\theta\_22} l(\theta\_22)$ ▷ Update parameters
        $\theta\_21 \leftarrow \theta\_21 - \eta \nabla_{\theta\_21} l(\theta\_21)$ ▷ Update parameters
        $\tilde{r}(s_t, a_t, s_{t+1}) \leftarrow r(s_t, a_t) + \alpha.\Delta r_{action/dynamics}(s_t, a_t, s_{t+1}) + (1-\alpha).\Delta \tilde{r}_{action/dynamics}(s_t, a_t, s_{t+1})$ ▷ Compute reward compensation of each policy based on θ values of each policy
        $\pi_{target} \leftarrow \text{MAXENT RL}(\pi_{target}, D_{\text{target}}, \tilde{r}, \Delta r_{task}, \Delta r_{action})$ ▷ Update policy by Δr exploration in entropy
    end if
    $D_{11} \leftarrow D_{11} \cup \text{ROLLOUT}(\pi, M_{11})$ ▷ Collect source data
    $\theta_{11} \leftarrow \theta_{11} - \eta \nabla_{\theta_{11}} l(\theta_{11})$ ▷ Update parameters
    $\theta_{22} \leftarrow \theta_{22} - \eta \nabla_{\theta_{22}} l(\theta_{22})$ ▷ Update parameters
    if $t \bmod r_{co} = 0$ **then**
        $\tilde{r}(s_t, a_t, s_{t+1}) \leftarrow r(s_t, a_t) + \beta.\Delta r_{action/dynamics}(s_t, a_t, s_{t+1}) + (1-\beta).\Delta \tilde{r}_{action/dynamics}(s_t, a_t, s_{t+1})$
        ▷ Update policy by training on reward compensation based on the source
    else
        $\tilde{r}(s_t, a_t, s_{t+1}) \leftarrow r(s_t, a_t) + \Delta r_{action/dynamics}(s_t, a_t, s_{t+1})$
        ▷ Update policy by training on reward compensation based on the source
    end if
    $\pi_{11} \leftarrow \text{MAXENT RL}(\pi_{11}, D_{11}, \tilde{r}, \Delta r_{task}, \Delta r_{action})$ ▷ Update policy by Δr exploration in entropy
    $\pi_{22} \leftarrow \text{MAXENT RL}(\pi_{22}, D_{22}, \tilde{r}, \Delta r_{task}, \Delta r_{action})$ ▷ Update policy by Δr exploration in entropy
end for
return π

ilar learning curves. This similarity suggests that the transfer learning is updating both trainings in a comparable manner. A similar behavior is observed in the V2 version as well. Therefore, in the default case where we have the same reward function but different dynamics, the main approach is deemed reliable, particularly when considering Algorithm 1, which is a simpler setup compared to Algorithms 2 and 3.

- Figure 9 illustrates the transfer case when there are the same dynamics but different reward functions (one is Euclidean and the other is Manhattan, different tasks and goals). As observed, in v3_low_r_algo and v3_high_r_algo, when utilizing Algorithm 1, both low and high resources behave similarly and converge in the same way during transfer learning. This pattern is consistent for V4 as well, indicating that the reward compensation is effective in scenarios where the dynamics are the same but the rewards differ.

- Therefore, we have already determined that in the default case of identical reward and dynamics, this type of reward compensation works in reward compensation transfer learning. Next, let's examine the results of other types of transfer by reward compensation.

- In the configuration where we have different rewards and dynamics (fig. 4) 10, by comparing v5_low_r_algo with v5_high_r_algo and v5_b_low_r_algo with v5_b_high_r_algo, it is observed that the low_resource agent learns at approximately the same speed as the high_resource agent. Although there are more fluctuations than in the two previous learning curves, the training process follows a similar pattern. Additionally, the return convergence is lower compared to the previous default setup. Let's examine whether adding more exploration to the SAC algorithm using reward compensation aids in achieving faster convergence.

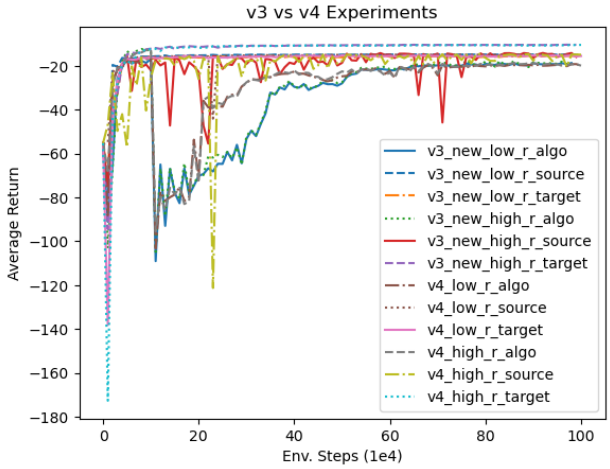

Figure 9: The dynamics of the source and target are the same as in Arm_1, but the reward functions differ; one is Euclidean and the other is Manhattan. The dynamics of the source and target are the same as in Arm_2, but the reward functions differ (one is Euclidean and the other is Manhattan).

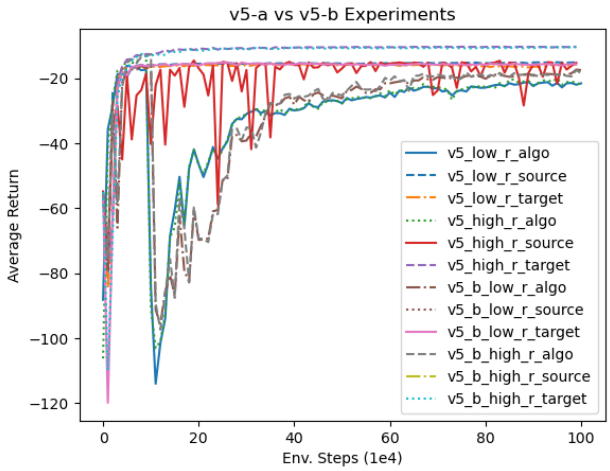

Figure 10: $V5-a$: In this experiment, Arm_1 runs on $Task_1$ and transfers to Arm_2 on $Task_2$, using the approach of Algorithm 1. $V5-b$: Arm_2 on $Task_2$ transfers to Arm_1 on $Task_1$, using the approach of Algorithm 1.

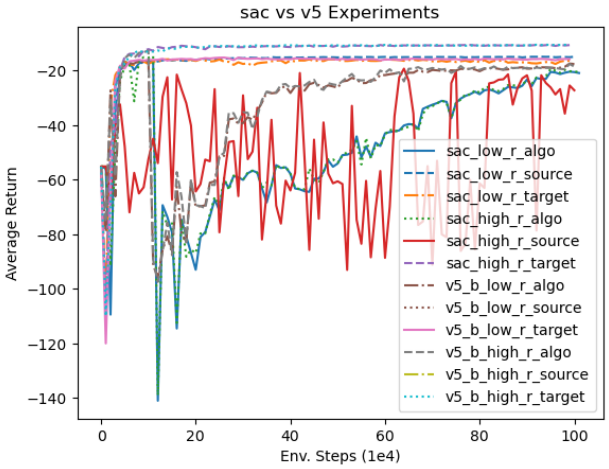

Figure 11: Using δr as reward compensation exploration in SAC algorithm of approach $V5$ and comparison with approach $V5$

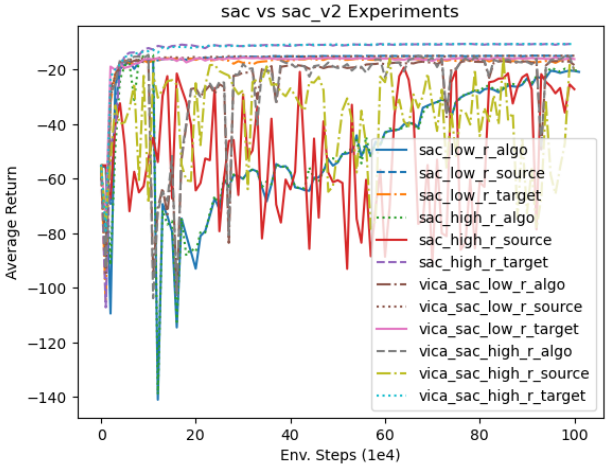

Figure 12: $V6$: Experiments using the SAC and reward compensation exploration in setup of approach in $V5$. In tag "vica" we reversed that source and target agents and tasks and redo the experiments with current setup.

value.

- Plot 11 illustrates the impact of exploration. In the current setup of the experiment, adding reward exploration appears to be unhelpful for the setup of Experiment 5, introducing more fluctuations in the transfer learning process. We anticipated that sac_low_r_algo would converge faster than v5_b_low_r_algo, and sac_high_r_algo would converge faster than v5_b_high_r_algo, but neither occurred. Instead, we observed lower rewards and more fluctuations (which are due to increased exploration in the new SAC augmented with reward compensation exploration). However, the exploration-based method eventually achieves the return value of v5. More environment steps may result in a better final return

- Let's consider changing the target and source, as one of the tasks or agents might be more challenging than the other, and exploration could fail when we use transfer. If this is the case, we should see better results when we swap the target and source. This is because if the previous target's exploration was more complex, then now the source would cover the required state space, making learning targets easier. However, if we observe the same pattern in exploration, it indicates that a more complicated state space is needed for exploration to be effective. Therefore, as shown in 11, we notice better convergence, suggesting that reward compensation exploration was helpful. But we did not get better results by just

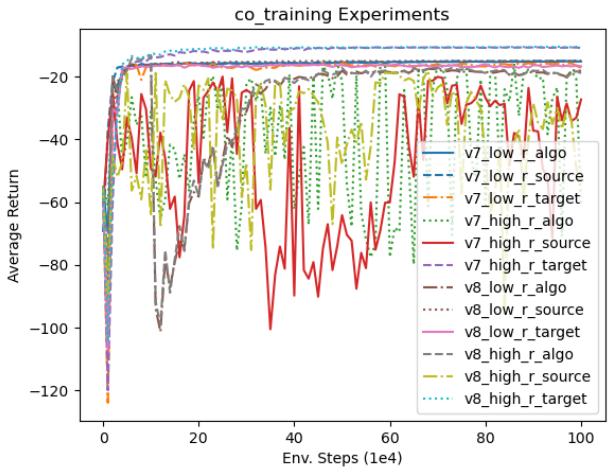

Figure 13: $V7$: Experiments using the $Transfer_1$ approach and co-training in algorithm 2 and $V8$: Experiments using the $Transfer_2$ approach and co-training in algorithm 3

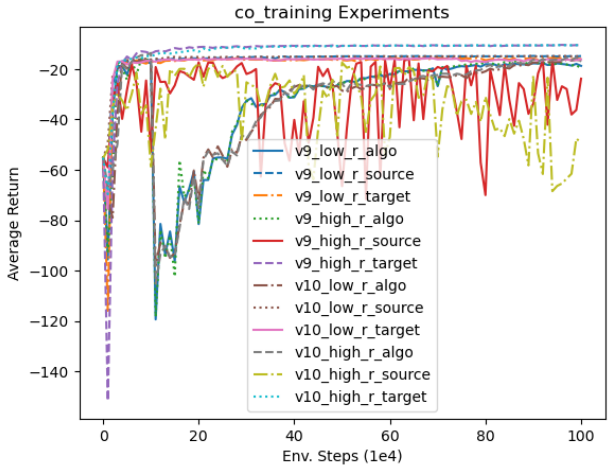

Figure 14: $V9$: Experiments using the $Transfer_1$ approach and co-training in algorithm 2 with reward compensation exploration and $V10$: Experiments using the $Transfer_2$ approach and co-training in algorithm 3 with reward compensation exploration

simply running RL on the target.

- Now, we aim to study the result of co-training on transfer method one and transfer method two in Algorithms 2 and 3. Based on 13, in $V7$, which demonstrates the transfer_1 method, a comparison between v7_low_r_algo and v7_high_r_algo indicates that there is not effective transfer in co-training, at least in the early stages. However, for v8_low_r_algo and v8_high_r_algo, it is observed that there is good convergence between the source and target in the transfer_2 method. See 13 for details.

- Plot 14 demonstrates how reward compensation exploration aids in co-training. As observed in $V9$ and $V10$, the fluctuations in transfer_1 become smaller, indicating that exploration has assisted co-training in transfer_1 (as seen by comparing v9_low_r_algo with v9_high_r_algo). However, it did not significantly affect the performance of transfer_2, as shown by the comparison of v10_low_r_algo with v10_high_r_algo.

6 Limitations

The main limitation of our project is that the state spaces of the tasks remain identical across different transfer problems. This issue can be mitigated by adopting an observation pixel-based state space. However, a more diverse dataset may be required, depending on the type of task.

7 Conclusion and Future works

In this project, we investigated the problem of multi-agent (with different dynamics) and multi-task (with different rewards) transfer. We conducted studies and experiments on reward compensation and exploration based on this compensation in multi-agent and multi-task transformers, using various setups. The types of task transfers can be extended to more agents and more complicated tasks with additional steps. It is likely that a more diverse range of tasks could lead to improved performance. The offline extension of this method is also beneficial. In addition to various types of transfers, the S4 and S5 extensions demonstrate strong performance in in-context RL. I believe that extending reward compensation aids in training sequence models through multi-task, multi-agent transitions and enhances the capacity of different transfer models for multi-agent, multi-task transitions.

8 Acknowledgment

The initial idea for this paper originated during the Deep Reinforcement Learning course in the fall of 2023 at UC Berkeley. I would like to thank the instructor team for their support.

References

[1] Bousmalis, Konstantinos, Giulia Vezzani, Dushyant Rao, Coline Devin, Alex X. Lee, Maria Bauza, Todor Davchev et al. "RoboCat: A Self-Improving Foundation Agent for Robotic Manipulation." arXiv preprint arXiv:2306.11706 (2023).
[2] Dasari, Sudeep, Frederik Ebert, Stephen Tian, Suraj Nair, Bernadette Bucher, Karl Schmeckpeper, Siddharth Singh, Sergey Levine, and Chelsea Finn. "Robonet: Large-scale multi-robot learning." arXiv preprint arXiv:1910.11215 (2019).
[3] Devin, C., Gupta, A., Darrell, T., Abbeel, P., & Levine, S. (2017, May). Learning modular neural network policies for multi-task and multi-robot transfer. In 2017 IEEE international conference on robotics and automation (ICRA) (pp. 2169-2176). IEEE.
[4] Eysenbach, B., Asawa, S., Chaudhari, S., Levine, S., & Salakhutdinov, R. (2020). Off-dynamics reinforcement learning: Training for transfer with domain classifiers. arXiv preprint arXiv:2006.13916.

[5] Padalkar, Abhishek, Acorn Pooley, Ajinkya Jain, Alex Bewley, Alex Herzog, Alex Irpan, Alexander Khazatsky et al. "Open x-embodiment: Robotic learning datasets and rt-x models." arXiv preprint arXiv:2310.08864 (2023).