Constrained GPI for Zero-Shot Transfer in Reinforcement Learning

Anonymous Author(s)
Affiliation
Address
email

Abstract

For zero-shot transfer in reinforcement learning where the reward function varies between different tasks, the successor features framework has been one of the popular approaches. However, in this framework, the transfer to new target tasks with generalized policy improvement (GPI) relies on only the source successor features [5] or additional successor features obtained solely with the function approximators' generalization to novel inputs [10]. The goal of this work is to improve the transfer by further bounding the value approximation errors of successor features on the new target tasks. To this end, we first present lower and upper bounds of optimal values for the novel tasks that are expressible as linear combinations of source tasks given the reward decomposition structure. Based on the bounds, we then propose *constrained GPI* as a simple test-time approach that can improve transfer by constraining action-value approximation errors on new target tasks. With experiments in the Scavenger and Reacher environments, we show that the proposed constrained GPI significantly outperforms the prior GPI's transfer performance.

1 Introduction

In the field of sequential decision making, deep reinforcement learning (RL) has been shown to be effective for various types of problems, including games [25] and robotics [18, 20]. With these successes, interest in multi-task RL has also been surged, where its goal is to train a single RL agent that can efficiently solve multiple varying tasks. In multi-task RL, we especially focus on the *transfer learning* setting, where the agent learns a shared structural knowledge from a set of source tasks during training and exploits and generalizes them in new, unseen target tasks at test time.

One popular approach to transfer in reinforcement learning is to leverage the successor features (SFs) framework [1, 5, 6, 10, 21], which can transfer policies learned on source tasks to target tasks, where the tasks share the same environment dynamics but differ in their reward functions. Successor features build a representation of value functions decoupled from reward functions, and enable their transfer to tasks with arbitrary reward functions by taking an inner product with corresponding task vectors. They utilize generalized policy improvement (GPI) [5], which is a generalized version of policy improvement by combining multiple policies, and it provides the performance lower bounds for GPI policies.

However, GPI does not take into account any information from the *smoothness* of action-value functions with respect to task vectors. Tackling this issue, Borsa et al. [10] propose universal successor features approximators (USFAs), which can estimate the optimal successor features for given novel task vectors. Nevertheless, the function approximators can make high approximation errors on the task vectors, especially when the new task vectors are distant from the source task vectors. For instance, when USFAs are trained with source tasks to get close to given goals, they may

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

not generalize well to the target tasks where the agent should get away from the given goals. That is, if we test on target task vectors whose elements have the opposite signs from the source task vectors, USFAs could output successor features with high approximation errors.

Therefore, we aim at bounding value approximation errors on the new target tasks to improve the successor features approximation of USFAs for the new tasks. We first introduce a new theorem on bounding the optimal values for the tasks that are expressible as linear combinations of source tasks, generalizing the conical combination condition used by the prior theorem by Nemecek and Parr [21]. Using the bounds that this theorem suggests as constraints, we can train the successor features approximators for bounding their action-value approximation errors on novel tasks. We extend this idea further so that we can accomplish a similar effect with no additional training; as a result, we propose *constrained GPI* as a test-time approach to bounding the approximation error. Despite its simplicity and no need for any modification to the training procedure, we empirically attain large performance improvements using the constrained GPI compared to the original GPI in the Scavenger [7, 8] and Reacher [11] environments.

Our main contributions can be summarized as follows:

- We present a novel theorem on lower- and upper-bounding optimal values for novel tasks that can be expressed as *linear* combinations of source tasks. It extends and generalizes the previous theorem for *conical* combinations by Nemecek and Parr [21], to enable a broader application of the bounds.
- Based on our new theorem, we propose *constrained GPI* as a simple test-time approach that can improve transfer to novel tasks by constraining action-value approximation errors on new target tasks, without any modification to the training procedure.
- We empirically show that our approach can improve the performance over the GPI baselines by a large margin in the Scavenger and Reacher environments. We also provide analyses for a better understanding of the results of our method.

2 Preliminaries

We describe the problem setting and background on successor features and universal successor features approximators. Please refer to Appendix for a more in-depth discussion of other related work.

2.1 The Zero-Shot Transfer Problem in RL

We define a Markov Decision Process (MDP) as $M \equiv (\mathcal{S}, \mathcal{A}, P, R, \gamma)$. $\mathcal{S}$ and $\mathcal{A}$ are the state and action spaces, and P is the transition probability where $P(\cdot|s,a)$ defines the probability distribution of the next states given $s \in \mathcal{S}$ and $a \in \mathcal{A}$. R is the reward function where $R(s,a,s')$ is the reward for taking action a at state s resulting in s', and $\gamma \in (0,1]$ is the discount factor. We assume that rewards are bounded.

We consider the zero-shot transfer problem in reinforcement learning; as in [5], each task is defined by its task vector $\boldsymbol{w} \in \mathbb{R}^d$, and only the reward functions differ across tasks, being decomposed as

$$R_{\boldsymbol{w}}(s,a,s') = \phi(s,a,s')^\top \boldsymbol{w}, \tag{1}$$

where $\phi(s,a,s') \in \mathbb{R}^d$ is the features of (s,a,s'). We denote the set of source task vectors as $\mathcal{T}$, which is used for training. We employ the assumption that the features $\phi(s,a,s')$ are available to the agent [5, 10], but our approach is orthogonal to the availability of ϕ and also applicable to the case where ϕ is learned from samples. At test time, we evaluate the transferred policy on each target task $\boldsymbol{w}' \notin \mathcal{T}$ without any additional update of pre-trained components.

2.2 Successor Features and Universal Successor Features Approximators

We now review successor features [5] and how they are transferred to different tasks. Equation (1) allows expressing the action-value function for policy π on task $\boldsymbol{w}$ as

$$Q^{\pi}_{\boldsymbol{w}}(s,a) = \mathbb{E}^{\pi}\left[\sum_{i=0}^{\infty}\gamma^i r_{t+i}\middle| S_t = s, A_t = a\right] = \mathbb{E}^{\pi}\left[\sum_{i=0}^{\infty}\gamma^i \phi_{t+i}\middle| S_t = s, A_t = a\right]^{\top}\boldsymbol{w} = \psi^{\pi}(s,a)^{\top}\boldsymbol{w}, \tag{2}$$

where $\phi_t = \phi(s_t, a_t, s_{t+1}) \in \mathbb{R}^d$. Here, $\psi^{\pi}(s,a) \in \mathbb{R}^d$ is called the successor features for policy π at (s,a), and taking its inner product with an arbitrary task $\boldsymbol{w}$ results in the action-value for π on $\boldsymbol{w}$; *i.e.*, $Q^{\pi}_{\boldsymbol{w}}$(s, a). Thanks to the analogy between (rewards r, action-value functions Q) and (features ϕ, successor features ψ), the Bellman equation applies to the successor features and thus they can be trained similarly as action-value functions are learned; *e.g.*, Q-learning.

The policy improvement theorem [9] states that a new policy that takes a greedy action according to a given policy's value function at each state performs at least as good as the original policy. Generalized policy improvement (GPI) [5] extends policy improvement to the case where the value functions of multiple policies are used. Given a task $\boldsymbol{w}'$, a set of policies $\pi_1, \ldots, \pi_n$, their action-value functions $Q^{\pi_1}_{\boldsymbol{w}'}, \ldots, Q^{\pi_n}_{\boldsymbol{w}'}$ and their approximations $\tilde{Q}^{\pi_1}_{\boldsymbol{w}'}, \ldots, \tilde{Q}^{\pi_n}_{\boldsymbol{w}'}$, the GPI policy is defined as

$$\pi_{\text{GPI}}(s) \in \operatorname*{argmax}_a \max_i \tilde{Q}^{\pi_i}_{\boldsymbol{w}'}(s,a), \tag{3}$$

and Barreto et al. [5] suggest that $Q^{\pi_{\text{GPI}}}_{\boldsymbol{w}'}(s,a) \geq \max_i Q^{\pi_i}_{\boldsymbol{w}'}(s,a) - \frac{2}{1-\gamma}\max_i \left\| Q^{\pi_i}_{\boldsymbol{w}'} - \tilde{Q}^{\pi_i}_{\boldsymbol{w}'}\right\|_{\infty}$. They also provide the upper bound on the suboptimality of the GPI policy as

$$\|Q^{*}_{\boldsymbol{w}'} - Q^{\pi_{\text{GPI}}}_{\boldsymbol{w}'}\|_{\infty} \leq \frac{2}{1-\gamma}\left\{\min_i \|\phi\|_{\infty}\|\boldsymbol{w}' - \boldsymbol{w}_i\| + \max_i \left\| Q^{\pi_i}_{\boldsymbol{w}'} - \tilde{Q}^{\pi_i}_{\boldsymbol{w}'}\right\|_{\infty}\right\}, \tag{4}$$

where each π_i is an optimal policy for $\boldsymbol{w}_i$.

While the GPI theorem allows transfer of learned successor features to arbitrary tasks with shared environment dynamics, it is limited in the following aspect; GPI uses the action-values for source tasks on target tasks based on the reward decomposition assumption (Equation (1)) *i.e.*, $\tilde{Q}^{\pi_i}_{\boldsymbol{w}'}(s,a) = \tilde{\psi}^{\pi_i}(s,a)^{\top}\boldsymbol{w}'$ for each i, but does not take any advantage of the *smoothness* of the optimal action-value functions with respect to different task vectors [10].

To overcome this limitation, Borsa et al. [10] introduce universal successor features approximators (USFAs). Inspired by universal value functions (UVFs) [24], they extend the original successor features with policy vectors $\boldsymbol{z} \in \mathbb{R}^l$ as input to their approximators. More specifically, universal successor features (USFs) are defined to satisfy

$$\psi^{\pi_{\boldsymbol{z}}}(s,a) \;\equiv\; \psi(s,a,\boldsymbol{z}) \;\approx\; \tilde{\psi}(s,a,\boldsymbol{z}), \tag{5}$$

where $\boldsymbol{z}$ is a policy vector for the policy $\pi_{\boldsymbol{z}}$, and USFAs $\tilde{\psi}$ are the learned approximators of USFs ψ. Naturally, the value functions are expressed as

$$\tilde{Q}^{\pi_{\boldsymbol{z}}}_{\boldsymbol{w}}(s,a) = \tilde{\psi}(s,a,\boldsymbol{z})^{\top}\boldsymbol{w} \approx \psi(s,a,\boldsymbol{z})^{\top}\boldsymbol{w} = Q^{\pi_{\boldsymbol{z}}}_{\boldsymbol{w}}(s,a). \tag{6}$$

Each reward function induces optimal policies, which can be encoded using the corresponding task vectors. That is, one can simply choose to define the policy vector space to be the same as the task vector space ($l = d$) and let $\boldsymbol{z} = \boldsymbol{w}$ be a policy vector of an optimal policy for task $\boldsymbol{w}$. Then, $\pi_{\boldsymbol{w}}$ and $Q^{\pi_{\boldsymbol{w}}}_{\boldsymbol{w}}$ denote an optimal policy for $\boldsymbol{w}$ and its action-value function, respectively.

The training of USFAs is similar to that of successor features, except for that it additionally involves sampling of policy vectors given task vectors. The update of USFAs at the k-th iteration is

$$\tilde{\psi}^{(k+1)} \leftarrow \operatorname*{argmin}_{\psi} \mathbb{E}_{\boldsymbol{w}\sim\mathcal{T},\boldsymbol{z}\sim\mathcal{D}_{\boldsymbol{z}}(\cdot|\boldsymbol{w}),(s,a,s')\sim\mu}\left[\left\|\phi(s,a,s') + \gamma\tilde{\psi}^{(k)}(s',a',\boldsymbol{z}) - \psi(s,a,\boldsymbol{z})\right\|^2\right] \tag{7}$$

for $a' = \operatorname{argmax}_b \tilde{\psi}^{(k)}(s',b,\boldsymbol{z})^{\top}\boldsymbol{z}$. $\mathcal{D}_{\boldsymbol{z}}(\cdot|\boldsymbol{w})$ is the policy vector distribution; for instance, $\mathcal{N}(\boldsymbol{w}, \sigma I)$ can be used for better training with diversified inputs. μ is the transition sampling distribution, which involves the GPI policy of the samples from $\mathcal{D}_{\boldsymbol{z}}(\cdot|\boldsymbol{w})$ or a replay buffer.

USFAs provide a benefit that they allow a GPI policy to use an arbitrary set of policies $\{\pi_{\boldsymbol{z}}\}_{\boldsymbol{z}\in\mathcal{C}}$ as $\pi_{\text{GPI}}(s) \in \text{argmax}_a \max_{\boldsymbol{z}\in\mathcal{C}} \tilde{Q}^{\pi_{\boldsymbol{z}}}_{\boldsymbol{w}'}(s,a)$. However, the generalization of USFAs to new policy vectors depends on a function approximator ψ, and thus if $\mathcal{C}$ contains policy vector(s) distant from source vectors, a GPI policy with $\mathcal{C}$ may have high approximation errors and perform poorly or even worse than a GPI policy with only source vectors [10], as will be demonstrated later in our experiments.

3 Constrained GPI for Improved Zero-Shot Transfer of Successor Features

To mitigate the aforementioned issue of the possibly unlimited approximation errors of USFAs, we propose a simple yet effective method that improves the transfer of successor features by further leveraging the reward decomposition structure in Equation (1). We first present, under a more relaxed condition, lower and upper bounds on the optimal values for novel task vectors, which can be expressed as linear combinations of source task vectors (Section 3.1). Then, we propose a novel approach called *constrained GPI*, which effectively confines the approximated action-values inside the computed lower and upper bounds (Section 3.2).

3.1 Bounding Optimal Values for New Tasks

Theorem 1 of [21] provides the lower and upper bounds on the value of an optimal policy for a new task, whose vector $\boldsymbol{w}'$ is a *positive conical combination* of source task vectors *i.e.*, $\boldsymbol{w}' = \sum_{\boldsymbol{w}\in\mathcal{T}} \alpha_{\boldsymbol{w}} \boldsymbol{w}$ such that $\alpha_{\boldsymbol{w}} \geq 0, \forall \boldsymbol{w} \in \mathcal{T}$ and $\sum_{\boldsymbol{w}\in\mathcal{T}} \alpha_{\boldsymbol{w}} > 0$[1]. However, for broad applications of such bounds, the positive conical combination condition could be too restrictive, since the resulting bounds only apply to the task vectors that appear inside the conical hull of source task vectors.

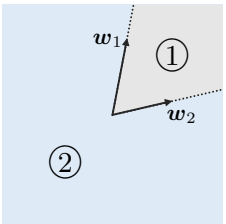


Figure 1: An example comparison of task space coverage. While the conical combinations of $\boldsymbol{w}_1$ and $\boldsymbol{w}_2$ covers only area ①, the linear combinations covers both area ①–②.

Therefore, we suggest a more relaxed theorem, which holds for an arbitrary task vector $\boldsymbol{w}'$ that is expressed as a linear combination of the source task vectors *i.e.*, $\boldsymbol{w}' = \sum_{\boldsymbol{w}\in\mathcal{T}} \alpha_{\boldsymbol{w}} \boldsymbol{w}$ for $\alpha_{\boldsymbol{w}} \in \mathbb{R}, \forall \boldsymbol{w} \in \mathcal{T}$. Figure 1 compares an example of the task space coverage of conical [21] and our linear combinations. With our extended task space coverage, we can use the bounds for improving transfer to more general target tasks outside of the conical hull, which will be further discussed in the next section.

We define $\epsilon^{\pi_{\boldsymbol{w}_1}}_{\boldsymbol{w}_2}$ to be an upper bound on the approximation error of $\tilde{Q}^{\pi_{\boldsymbol{w}_1}}_{\boldsymbol{w}_2}$ for arbitrary tasks $\boldsymbol{w}_1, \boldsymbol{w}_2$ such that

$$|Q^{\pi_{\boldsymbol{w}_1}}_{\boldsymbol{w}_2}(s,a) - \tilde{Q}^{\pi_{\boldsymbol{w}_1}}_{\boldsymbol{w}_2}(s,a)| \leq \epsilon^{\pi_{\boldsymbol{w}_1}}_{\boldsymbol{w}_2}(s,a), \quad \forall (s,a) \in \mathcal{S}\times\mathcal{A}, \tag{8}$$

and we present our theorem as follows.

Theorem 1. *Given a task vector $\boldsymbol{w}' = \sum_{\boldsymbol{w}\in\mathcal{T}} \alpha_{\boldsymbol{w}} \boldsymbol{w}$ for $\alpha_{\boldsymbol{w}} \in \mathbb{R}, \forall \boldsymbol{w} \in \mathcal{T}$, for all state-action pairs $(s,a) \in \mathcal{S}\times\mathcal{A}$, the action-value of $\pi_{\boldsymbol{w}'}$ for an optimal policy for task $\boldsymbol{w}'$ is lower- and upper-bounded as $L_{\boldsymbol{w}',\mathcal{T}}(s,a) \leq Q^{\pi_{\boldsymbol{w}'}}_{\boldsymbol{w}'}(s,a) \leq U_{\boldsymbol{w}',\mathcal{T},\boldsymbol{\alpha}}(s,a)$ for*

$$L_{\boldsymbol{w}',\mathcal{T}}(s,a) \coloneqq \max_{\boldsymbol{w}\in\mathcal{T}} \left[\tilde{Q}^{\pi_{\boldsymbol{w}}}_{\boldsymbol{w}'}(s,a) - \epsilon^{\pi_{\boldsymbol{w}}}_{\boldsymbol{w}'}(s,a)\right], \tag{9}$$

$$U_{\boldsymbol{w}',\mathcal{T},\boldsymbol{\alpha}}(s,a) \coloneqq \sum_{\boldsymbol{w}\in\mathcal{T}} \max\left\{\alpha_{\boldsymbol{w}}\left(\tilde{Q}^{\pi_{\boldsymbol{w}}}_{\boldsymbol{w}}(s,a) + \epsilon^{\pi_{\boldsymbol{w}}}_{\boldsymbol{w}}(s,a)\right), \alpha_{\boldsymbol{w}}\frac{1}{1-\gamma} r^{min}_{\boldsymbol{w}}\right\}, \tag{10}$$

where $r^{min}_{\boldsymbol{w}}$ is the minimum reward on $\boldsymbol{w}$ i.e.*, $r^{min}_{\boldsymbol{w}} = \min_{(s,a)\in\mathcal{S}\times\mathcal{A}} R_{\boldsymbol{w}}(s,a)$ and $\boldsymbol{\alpha} = \{\alpha_{\boldsymbol{w}}\}_{\boldsymbol{w}\in\mathcal{T}}$.*

Proof. For the derivation of the lower bound $L_{\boldsymbol{w}',\mathcal{T}}(s,a)$, since $Q^{\pi_{\boldsymbol{w}'}}_{\boldsymbol{w}'}$ is the optimal action-value function for task $\boldsymbol{w}'$ and $Q^{\pi_{\boldsymbol{w}'}}_{\boldsymbol{w}'}(s,a) \geq Q^{\pi_{\boldsymbol{w}}}_{\boldsymbol{w}'}(s,a)$ for arbitrary task $\boldsymbol{w}$ and state-action pair (s,a),

$$Q^{\pi_{\boldsymbol{w}'}}_{\boldsymbol{w}'}(s,a) \;\geq\; \max_{\boldsymbol{w}\in\mathcal{T}} Q^{\pi_{\boldsymbol{w}}}_{\boldsymbol{w}'}(s,a) \;\geq\; \max_{\boldsymbol{w}\in\mathcal{T}} \left[\tilde{Q}^{\pi_{\boldsymbol{w}}}_{\boldsymbol{w}'}(s,a) - \epsilon^{\pi_{\boldsymbol{w}}}_{\boldsymbol{w}'}(s,a)\right]. \tag{11}$$

[1] We slightly abuse the notation and let $\alpha_{\boldsymbol{w}}$ denote the coefficient for vector $\boldsymbol{w}$.

For the upper bound $U_{\boldsymbol{w}',\mathcal{T},\boldsymbol{\alpha}}(s,a)$, we use that $Q^{\pi_{\boldsymbol{w}'}}_{\boldsymbol{w}}(s,a) \leq Q^{\pi_{\boldsymbol{w}}}_{\boldsymbol{w}}(s,a)$ and $Q^{\pi_{\boldsymbol{w}'}}_{\boldsymbol{w}}(s,a) \geq \frac{1}{1-\gamma} r^{\min}_{\boldsymbol{w}}$ for arbitrary task $\boldsymbol{w}$ and state-action pair (s,a), which leads to

$$
\begin{aligned}
Q^{\pi_{\boldsymbol{w}'}}_{\boldsymbol{w}'}(s,a) &= \sum_{\boldsymbol{w}\in\mathcal{T}} \alpha_{\boldsymbol{w}} \left(Q^{\pi_{\boldsymbol{w}'}}_{\boldsymbol{w}}(s,a) - \frac{1}{1-\gamma} r^{\min}_{\boldsymbol{w}} \right) + \frac{1}{1-\gamma} \sum_{\boldsymbol{w}\in\mathcal{T}} \alpha_{\boldsymbol{w}} r^{\min}_{\boldsymbol{w}} && (12)\\
&\leq \sum_{\boldsymbol{w}\in\mathcal{T}} \max\left\{ \alpha_{\boldsymbol{w}} \left(Q^{\pi_{\boldsymbol{w}'}}_{\boldsymbol{w}}(s,a) - \frac{1}{1-\gamma} r^{\min}_{\boldsymbol{w}} \right), 0 \right\} + \frac{1}{1-\gamma} \sum_{\boldsymbol{w}\in\mathcal{T}} \alpha_{\boldsymbol{w}} r^{\min}_{\boldsymbol{w}} && (13)\\
&\leq \sum_{\boldsymbol{w}\in\mathcal{T}} \max\left\{ \alpha_{\boldsymbol{w}} \left(Q^{\pi_{\boldsymbol{w}}}_{\boldsymbol{w}}(s,a) - \frac{1}{1-\gamma} r^{\min}_{\boldsymbol{w}} \right), 0 \right\} + \frac{1}{1-\gamma} \sum_{\boldsymbol{w}\in\mathcal{T}} \alpha_{\boldsymbol{w}} r^{\min}_{\boldsymbol{w}} && (14)\\
&= \sum_{\boldsymbol{w}\in\mathcal{T}} \left\{ \max\left\{ \alpha_{\boldsymbol{w}} \left(Q^{\pi_{\boldsymbol{w}}}_{\boldsymbol{w}}(s,a) - \frac{1}{1-\gamma} r^{\min}_{\boldsymbol{w}} \right), 0 \right\} + \alpha_{\boldsymbol{w}} \frac{1}{1-\gamma} r^{\min}_{\boldsymbol{w}} \right\} && (15)\\
&= \sum_{\boldsymbol{w}\in\mathcal{T}} \max\left\{ \alpha_{\boldsymbol{w}} Q^{\pi_{\boldsymbol{w}}}_{\boldsymbol{w}}(s,a), \alpha_{\boldsymbol{w}} \frac{1}{1-\gamma} r^{\min}_{\boldsymbol{w}} \right\} && (16)\\
&\leq \sum_{\boldsymbol{w}\in\mathcal{T}} \max\left\{ \alpha_{\boldsymbol{w}} \left(\tilde{Q}^{\pi_{\boldsymbol{w}}}_{\boldsymbol{w}}(s,a) + \epsilon^{\pi_{\boldsymbol{w}}}_{\boldsymbol{w}}(s,a) \right), \alpha_{\boldsymbol{w}} \frac{1}{1-\gamma} r^{\min}_{\boldsymbol{w}} \right\}. && (17)
\end{aligned}
$$

□

Intuitively, this theorem states the condition that the optimal action-value of an arbitrary target task must satisfy, for the given optimal action-values of the source tasks. Thus, as will be shown in the next section, this can be employed as a constraint on the action-values for novel target tasks, for the guidance of transfer. Also, note that our Theorem 1 recovers Theorem 1 of [21] when $\boldsymbol{w}'$ is a conical combination of $\boldsymbol{w}$'s from $\mathcal{T}$ *i.e.*, $\alpha_{\boldsymbol{w}} \geq 0, \forall \boldsymbol{w} \in \mathcal{T}$. In Sections 4.1 and 4.2, we will empirically show that the application of our Theorem 1 can result in significant performance improvement in the case where target tasks are outside the conical hull of source tasks.

3.2 Constrained Training and Constrained GPI

As described in Section 2.2, the universal successor features approximators (USFAs) [10] improve the original successor features so that arbitrary policy vectors, including the ones for target tasks, can be used for GPI. However, the use of arbitrary policy vectors with USFAs solely relies on the generalization power of the approximators (*e.g.* neural networks). Thus, the obtained successor features on novel tasks might contain high approximation errors, which could make the GPI policy perform poorly.

Our high-level idea to tackle the issue is to exploit the reward decomposition structure in Equation (1) even for obtaining successor features for new tasks, instead of solely relying on the approximators. We employ the lower and upper bounds on optimal values from Theorem 1 to enforce the bounds on the approximate successor features. As a result, the approximation errors can be reduced by restricting the estimated optimal values to be inside those bounds around the optimal values, which can prevent the use of erroneous values during the transfer to unseen tasks.

For now, we will first introduce how to train the successor features approximators to output the successor features that satisfy the bounds on novel tasks. Then, we will point out that an analogous effect can be accomplished by modifying only the inference algorithm, and propose *constrained GPI* as a simple yet effective *test-time* approach to improving zero-shot transfer to novel tasks.

Constrained training of SF approximators. In the original training of USFAs, the approximators are learned with a set of source tasks $\mathcal{T}$ in Equation (7). We propose to guide the training by employing Theorem 1; we impose constraints for the approximators using the lower and upper bounds on the optimal values for arbitrary linear combinations of source tasks. Specifically, for the training of USFAs, we use Equation (7) but with the following constraints:

$$
L_{\boldsymbol{w}',\mathcal{T}}(s,a) \leq \tilde{\psi}(s,a,\boldsymbol{w}')^\top \boldsymbol{w}' \leq U_{\boldsymbol{w}',\mathcal{T},\xi(\boldsymbol{w}',\mathcal{T},s,a)}(s,a) \quad \text{for } \boldsymbol{w}' \in \mathcal{W}, \tag{18}
$$

where (s,a) is the same sample as the main objective of Equation (7). $\mathcal{W}$ is a set of task vectors for the constraints, which can be independent of the source task set $\mathcal{T}$, and $\xi(\cdot)$ determines the

coefficients $\boldsymbol{\alpha}$ given a target task $\boldsymbol{w}'$ and $\mathcal{T}$. We will explain later how to determine $\xi(\cdot)$. $\mathcal{W}$ can be any subset of the linear span of source task vectors, but practically, we randomly sample a number of vectors from the span at each update. Since the targets of the constraints are not fixed with respect to both $\boldsymbol{w}'$ and (s, a) throughout the training, we use penalty terms (or soft constraints) that linearly penalize the constraint violations as

$$\frac{1}{|\mathcal{W}|}\sum_{\boldsymbol{w}'\in\mathcal{W}}\Big(\Big\{L_{\boldsymbol{w}',\mathcal{T}}(s,a)-\tilde{Q}^{\pi_{\boldsymbol{w}'}}_{\boldsymbol{w}'}(s,a)\Big\}_{+}+\Big\{\tilde{Q}^{\pi_{\boldsymbol{w}'}}_{\boldsymbol{w}'}(s,a)-U_{\boldsymbol{w}',\mathcal{T},\xi(\boldsymbol{w}',\mathcal{T},s,a)}(s,a)\Big\}_{+}\Big),$$

where $\{x\}_+$ denotes $\max\{x, 0\}$.

The constrained training suggested above can make the approximators comply with the bounds for any tasks without requiring any additional interactions with the environment. However, it has some downsides. First, since it is a new training procedure, existing pre-trained models cannot be used. It requires some additional computational cost compared to the naive training of successor features approximators. Second, the enforcement of the constraints for training can introduce additional hyperparameters (*e.g.* the weight coefficient of the penalty terms). Thus, their suboptimal choice may introduce instability in training or decrease performance.

Test-time constrained GPI. Our idea starts from the observation that in the constrained training, the learned successor features from source tasks are considered the "trustworthy" features for the constraints, because the USFAs are trained on the source tasks. Besides, only the source successor features are used for the computation of the constraints for all the other tasks. It also implies that it is better that the learning of the source successor features is not affected by other criteria and more accurate source successor features would produce better constraints for other tasks with smaller errors.

Based on the implication, we propose *constrained GPI*, which can not only overcome the limitation of USFA as done by the aforementioned constrained training but also have two additional practical merits: (i) it is computationally simpler, and (ii) it is a test-time approach with no training. Simply put, we propose replacing the usual GPI policy with the constrained GPI policy as

$$\pi_{\text{CGPI}}(s)\in\operatorname*{argmax}_{a}\max_{\boldsymbol{z}\in\mathcal{C}}\left[\min\left\{\max\left\{\tilde{Q}^{\pi_{\boldsymbol{z}}}_{\boldsymbol{w}'}(s,a),L_{\boldsymbol{w}',\mathcal{T}}(s,a)\right\},U_{\boldsymbol{w}',\mathcal{T},\xi(\boldsymbol{w}',\mathcal{T},s,a)}(s,a)\right\}\right],\quad(19)$$

where the target task $\boldsymbol{w}'$ is expressible as a linear combination of the source tasks and $\xi(\cdot)$ again outputs $\boldsymbol{\alpha}$ given $\boldsymbol{w}'$ and $\mathcal{T}$ as in Equation (18). $\mathcal{C}$ is a set of policies that we can freely choose when applying the constrained GPI.

The constrained GPI policy selects the actions that maximize the maximum action-values as the original GPI policy does but also caps the action-values by the lower and upper bound constraints derived from the source successor features. The upper bound constraint fixes the overestimation of values computed with approximate successor features for either the target task $\boldsymbol{w}'$ or any other tasks used for constrained GPI. The lower bound constraint ensures that action-values on the target task used for the greedy action selection are at least as close to the optimal target action-values as the lower bounds.

The approximation error terms in the lower and upper bounds *i.e.*, $\epsilon^{\pi_{\boldsymbol{w}}}_{\boldsymbol{w}'}(s,a)$ and $\epsilon^{\pi_{\boldsymbol{w}}}_{\boldsymbol{w}}(s,a)$ in Theorem 1 could be ignored in practice, as long as the approximation errors of the source successor features themselves can be considered to be small. Also, we can obtain the tightest possible upper bound by defining $\xi(\cdot)$ as

$$\xi(\boldsymbol{w}',\mathcal{T},s,a)\coloneqq\operatorname*{argmin}_{\{\alpha_{\boldsymbol{w}}\}_{\boldsymbol{w}\in\mathcal{T}}}U_{\boldsymbol{w}',\mathcal{T},\{\alpha_{\boldsymbol{w}}\}_{\boldsymbol{w}\in\mathcal{T}}}(s,a)\quad\text{subject to}\ \ \boldsymbol{w}'=\sum_{\boldsymbol{w}\in\mathcal{T}}\alpha_{\boldsymbol{w}}\boldsymbol{w}.\qquad(20)$$

The objective $U_{\boldsymbol{w}',\mathcal{T},\{\alpha_{\boldsymbol{w}}\}_{\boldsymbol{w}\in\mathcal{T}}}(s,a)$ is the sum of the piecewise linear functions, and thus Equation (20) can be solved with linear programming.

We observe that using the lower bound constraint with $L_{\boldsymbol{w}',\mathcal{T}}(s,a)$ is equivalent to including the successor features of source tasks in the input to the constrained GPI; *i.e.*, $\mathcal{T}\subseteq\mathcal{C}$. Also, since $L_{\boldsymbol{w}',\mathcal{T}}(s,a)\leq U_{\boldsymbol{w}',\mathcal{T},\xi(\boldsymbol{w}',\mathcal{T},s,a)}(s,a)$, there would be no difference between GPI and constrained GPI when $\mathcal{C}=\mathcal{T}$. Thus, in our experiments, we mainly use $\mathcal{C}=\{\boldsymbol{w}'\}$, which is equivalent to using $\mathcal{C}=\mathcal{T}\cup\{\boldsymbol{w}'\}$.

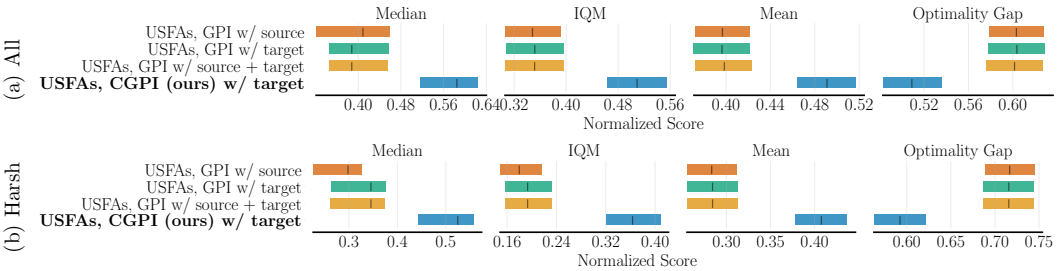


Figure 2: The aggregated performance metrics with 95% bootstrap confidence intervals [2, 14] of different approaches with universal successor features approximators (USFAs) on Scavenger with $d = 2$. CGPI represents *constrained GPI* where GPI stands for generalized policy improvement. (a) "All" is the evaluation on the entire set of target task vectors from $\{-1, 1\}^d$, whereas (b) "Harsh" denotes the evaluation on a subset consisting of 'harsh' tasks, whose number of -1's is no less than that of 1's (*e.g.*, $\boldsymbol{w} = (-1, 1)$). Per each method, eight models trained for 1M steps are evaluated 10 times for each target task with a fixed set of 10 random seeds. Scores for each task are normalized by the minimum and maximum scores across all the agents' evaluation episodes on that task.

4 Experiments

4.1 Scavenger Experiments

We start our experiments in the Scavenger environment [7, 8], to assess the effectiveness of our approach with as minimal influence from external causes as possible. In Scavenger, the agent is positioned at one of the cells in a $G \times G$ grid, and the goal is to maximize the return by collecting objects. Not only the agent but also objects are spawned at random locations, and there are d classes of objects where the class determines the value of the reward for each object. The state space is $\mathcal{S} = \{0, 1\}^{G \times G \times (d+1)}$, where the first d channels describe the current locations of the objects on the map and the last channel specifies the walls where the agent cannot go and objects do not appear. There are four actions available: $\mathcal{A} = \{\texttt{UP}, \texttt{DOWN}, \texttt{LEFT}, \texttt{RIGHT}\}$, and the agent picks up an object automatically by visiting the cell of the object, which makes a new object of a random class appear at a random location. Each element of the feature $\phi(s, a, s') \in \mathbb{R}^d$ represents whether the agent is picking up an object of that type or not with 1 or 0, and the task vector $\boldsymbol{w} \in \mathbb{R}^d$ determines reward values for the d different classes of objects. Please see Barreto et al. [8] for the full details.

We evaluate the zero-shot transfer performance of agents with different approaches *i.e.*, we measure the performance of the same set of trained USFAs with GPI and constrained GPI policies on target tasks without any further policy updates. We perform the evaluation with $G = 11$, 20 objects in total and different numbers of classes; $d = 2$ and $d = 4$, and with $d = 4$, we also include USFAs agents learned with the constrained training for a comparison. We use the standard basis vectors of $\mathbb{R}^d$ as the set of source tasks similarly to [10], and evaluate agents on the set of target tasks defined by $\{-1, 1\}^d$. Therefore, all the target tasks except for the all-ones vector, $\mathbf{1}$, are not covered by the conical hull of source tasks, which requires Theorem 1 for bounding of values.

We train eight USFAs agents for 1M environment steps, and evaluate them on each target vector 10 times with a fixed set of 10 random seeds. To evaluate them regardless of the reward scale difference between different tasks, we normalize the scores (or returns) from the environment by the minimum and maximum scores with respect to all the agents' evaluation episodes on each task.

Table 1: The portion of the maximum action-values and resulting actions actually affected by the lower and upper bounds used by constrained GPI, during the evaluation on Scavenger.

Setting	Lower-bounding	Upper-bounding	Action change
$d = 2$	23.94%	43.01%	22.11%
$d = 4$	5.60%	46.86%	20.78%

Figures 2 and 3 compare the performance of the USFAs agents with GPI and constrained GPI for exploitation, following the evaluation scheme suggested by [2]. The results show that although they use the same set of trained USFAs, using constrained GPI brings a notable performance improvement in comparison with the original GPI. Also, Figure 3

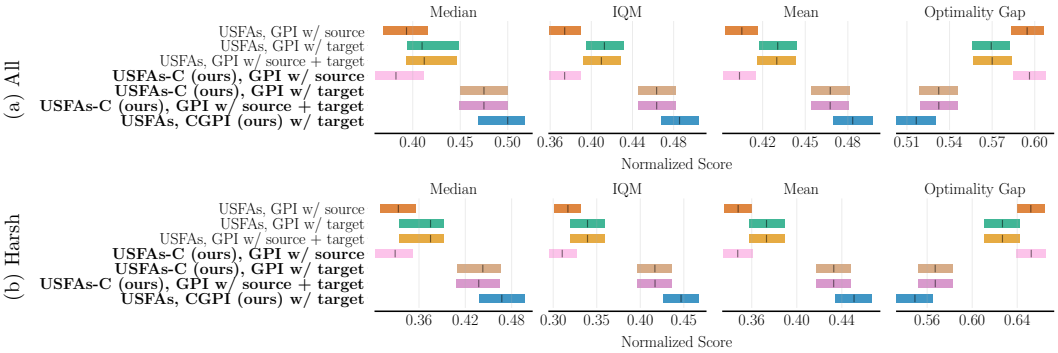


Figure 3: The aggregated performance metrics with 95% bootstrap confidence intervals [2, 14] of different approaches on Scavenger with $d = 4$. USFAs-C denotes USFAs trained with the constrained training. We refer the reader to Figure 2 for the full description.

suggests that constrained GPI, the test-time method, can match or even outperform the agents learned with the constrained training. One possible explanation is that the constrained training might experience some instability in the learning depending on the choice of the hyperparameters, as described in Section 3.2.

Table 1 presents the portion of the action-values after taking the maximum and resulting greedy actions that are changed by constrained GPI's lower and upper bounds. It implies that USFAs *i.e.*, the function approximators themselves may not satisfy the value bounds in Theorem 1 and applying the bounds could meaningfully improve the performance.

4.2 Robotic Locomotion Experiments

To evaluate our approach in a more realistic and physical domain, following [5, 21], we employ Reacher, a MuJoCo's robotic locomotion environment [26] from OpenAI Gym [11]. Reacher simulates a robotic arm with two joints, and its state space $\mathcal{S}$ is 11-dimensional. Its original action space is continuous and two-dimensional, which represents torques at the two hinge joints, and we discretize each action dimension into three values resulting in nine discrete actions in total, as in [5].

For its use in the zero-shot transfer problem, we first set four fixed goal locations at $(0.1, 0.0), (0.0, 0.1), (-0.1, 0.0), (0.0, -0.1)$, and define $\phi(s, a, s') \in \mathbb{R}^4$ to be a vector whose elements are the negative distances between the agent's fingertip and the four goals. USFAs agents are trained with the four standard basis vectors as source tasks, learning to reach or get close to one of the four goals on each of the four source tasks. However, for the evaluation, we define $\{-1, 1\}^d$ to be the set of target tasks. A negative value at each dimension implies not only that the target vector is outside the conical hull of the source tasks but also that the agent would obtain higher rewards with respect to that dimension by getting away from the corresponding goal, instead of reaching the goal. This can make this zero-shot transfer problem challenging, as it requires the agent to do very different behaviors suddenly at test time. For better understanding, Figure 5 shows a rendered scene of Reacher, where the four red dots represent the goal locations.

In our experiments, we train 16 USFAs agents for 1M environment steps to obtain statistically more meaningful results. We evaluate each of the trained agents with 10 episodes for each target vector, again with a set of 10 environment random seeds. Similarly to the Scavenger experiments in the previous section, to take into account the difference in reward scales between different target tasks, for each of the target tasks, we normalize the returns *i.e.*, scores using the minimum and maximum scores that any of the agents achieve during the evaluation.

Figure 4 provides the comparison of GPI and constrained GPI in terms of the normalized scores. As it shows, the inference with constrained GPI significantly outperforms the ones with GPI on the zero-shot transfer problem. Especially, the gap becomes even larger when they are evaluated on the *Harsh* set of target task vectors, which implies that constrained GPI can be helpful for transferring to different types of target tasks outside the conical hull of source tasks. On the other hand, Figure 7 presents the performance profiles [2, 13] of the approaches with the matching evaluation categories

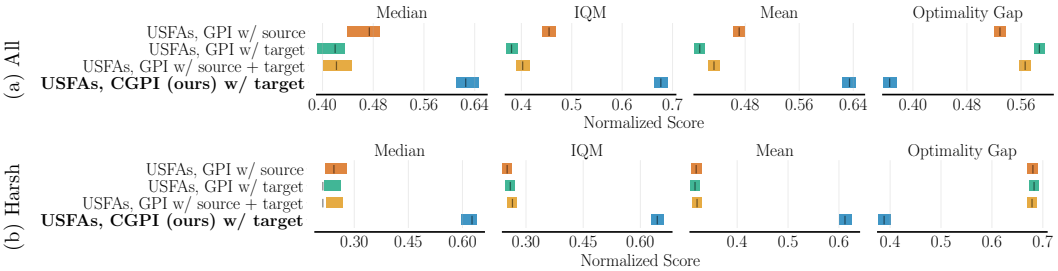


Figure 4: The aggregated performance metrics with 95% bootstrap confidence intervals [2, 14] of different approaches on Reacher. CGPI represents *constrained GPI*. Per each method, 16 models trained for 1M steps are evaluated 10 times for each target task with a fixed set of 10 random seeds. We refer the reader to Figure 2 for the full description.

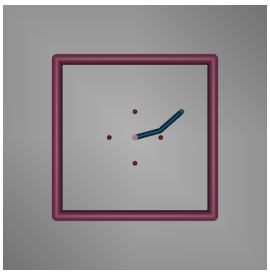
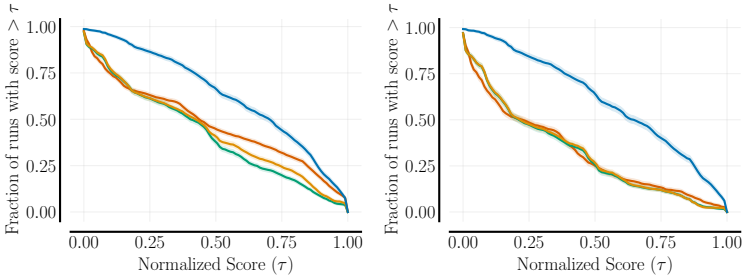


Figure 5: An example scene of Reacher. The four red dots indicate the four goal locations.



(a) All (b) Harsh

Figure 7: The performance profiles [2, 13] of the inference with GPI and constrained GPI on Reacher. The categories *i.e.*, (a) All and (b) Harsh, and the colors match the ones in Figure 4, and thus the blue lines represent constrained GPI.

and color mapping. It qualitatively suggests that constrained GPI is more likely to achieve higher scores than GPI.

5 Conclusion and Discussion

We presented constrained GPI, a simple yet effective test-time approach for transfer with approximate successor features. We first focused on the issue that although universal successor features approximators (USFAs) exploit the smoothness of optimal values across different tasks, their approximation errors on novel target tasks could be large especially when those tasks are quite distant from source tasks. Thus, we presented a theorem about lower and upper bounds on the optimal values for novel tasks which are expressible as linear combinations of source tasks, relaxing the conical combination condition used by the theorem by Nemecek and Parr [21]. We proposed the constrained training scheme making use of those bounds for reducing the action-value approximation errors of the learned approximators on novel tasks. We then suggested *constrained GPI* and using the bounds only at test time to achieve an analogous effect, allowing the use of existing trained models, and empirically showed that this test-time approach can improve the zero-shot transfer performance by a large margin.

Limitations and future directions. There may be cases where the minimum rewards for source tasks *i.e.*, $r_{\boldsymbol{w}}^{\min}$'s are overly small, which could lead to less changes of action-values and thus behaviors induced by the upper-bounding in Theorem 1. Also, as our approach makes use of the successor features framework, it needs ϕ to be observable or learned with a function approximator. On the other hand, extending our constrained GPI to other non-linear forms of reward or value decompositions could be an intriguing direction for future research. We don't see direct negative societal impacts of this work.

References

[1] Majid Abdolshah, Hung Le, Thommen George Karimpanal, Sunil Gupta, Santu Rana, and Svetha Venkatesh. A new representation of successor features for transfer across dissimilar environments. In *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, pages 1–9. PMLR, 2021.

[2] Rishabh Agarwal, Max Schwarzer, Pablo Samuel Castro, Aaron C Courville, and Marc Bellemare. Deep reinforcement learning at the edge of the statistical precipice. *Advances in Neural Information Processing Systems*, 34, 2021.

[3] A. Agrawal, B. Amos, S. Barratt, S. Boyd, S. Diamond, and Z. Kolter. Differentiable convex optimization layers. In *Advances in Neural Information Processing Systems*, 2019.

[4] Safa Alver and Doina Precup. Constructing a good behavior basis for transfer using generalized policy updates. In *10th International Conference on Learning Representations, ICLR 2022*, 2022.

[5] André Barreto, Will Dabney, Rémi Munos, Jonathan J. Hunt, Tom Schaul, David Silver, and Hado van Hasselt. Successor features for transfer in reinforcement learning. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 4055–4065, 2017.

[6] André Barreto, Diana Borsa, John Quan, Tom Schaul, David Silver, Matteo Hessel, Daniel J. Mankowitz, Augustin Zídek, and Rémi Munos. Transfer in deep reinforcement learning using successor features and generalised policy improvement. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, pages 510–519. PMLR, 2018.

[7] André Barreto, Diana Borsa, Shaobo Hou, Gheorghe Comanici, Eser Aygün, Philippe Hamel, Daniel Toyama, Jonathan J. Hunt, Shibl Mourad, David Silver, and Doina Precup. The option keyboard: Combining skills in reinforcement learning. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 13031–13041, 2019.

[8] André Barreto, Shaobo Hou, Diana Borsa, David Silver, and Doina Precup. Fast reinforcement learning with generalized policy updates. *Proceedings of the National Academy of Sciences*, (48):30079–30087, 2020.

[9] Richard E. Bellman. Dynamic programming. *Princeton University Press*, 1957.

[10] Diana Borsa, André Barreto, John Quan, Daniel J. Mankowitz, Hado van Hasselt, Rémi Munos, David Silver, and Tom Schaul. Universal successor features approximators. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019.

[11] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. OpenAI Gym. *arXiv preprint arXiv:1606.01540*, 2016.

[12] Peter Dayan. Improving generalization for temporal difference learning: The successor representation. *Neural Computation*, (4):613–624, 1993.

[13] Elizabeth D Dolan and Jorge J Moré. Benchmarking optimization software with performance profiles. *Mathematical programming*, 91(2):201–213, 2002.

[14] Bradley Efron. Bootstrap methods: another look at the jackknife. In *Breakthroughs in statistics*, pages 569–593. Springer, 1992.

[15] Fernando Fernández and Manuela Veloso. Probabilistic policy reuse in a reinforcement learning agent. In *Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems*, pages 720–727, 2006.

[16] Zhang-Wei Hong, Ge Yang, and Pulkit Agrawal. Bilinear value networks. In *10th International Conference on Learning Representations, ICLR 2022*, 2022.

[17] Jonathan J. Hunt, André Barreto, Timothy P. Lillicrap, and Nicolas Heess. Composing entropic policies using divergence correction. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, pages 2911–2920. PMLR, 2019.

[18] Dmitry Kalashnikov, Alex Irpan, Peter Pastor, Julian Ibarz, Alexander Herzog, Eric Jang, Deirdre Quillen, Ethan Holly, Mrinal Kalakrishnan, Vincent Vanhoucke, and Sergey Levine. Qt-opt: Scalable deep reinforcement learning for vision-based robotic manipulation. *ArXiv*, abs/1806.10293, 2018.

[19] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2015.

[20] Ashish Kumar, Zipeng Fu, Deepak Pathak, and Jitendra Malik. Rma: Rapid motor adaptation for legged robots. *ArXiv*, abs/2107.04034, 2021.

[21] Mark W. Nemecek and Ron Parr. Policy caches with successor features. In *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, pages 8025–8033. PMLR, 2021.

[22] Chris Nota. The autonomous learning library. `https://github.com/cpnota/autonomous-learning-library`, 2020.

[23] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, 2019.

[24] Tom Schaul, Daniel Horgan, Karol Gregor, and David Silver. Universal value function approximators. In *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, pages 1312–1320. JMLR.org, 2015.

[25] Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, L. Sifre, Simon Schmitt, Arthur Guez, Edward Lockhart, Demis Hassabis, Thore Graepel, Timothy P. Lillicrap, and David Silver. Mastering atari, go, chess and shogi by planning with a learned model. *Nature*, 588 7839:604–609, 2020.

[26] Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5026–5033. IEEE, 2012.

Checklist

The checklist follows the references. Please read the checklist guidelines carefully for information on how to answer these questions. For each question, change the default **[TODO]** to [Yes] , [No] , or [N/A] . You are strongly encouraged to include a **justification to your answer**, either by referencing the appropriate section of your paper or providing a brief inline description. For example:

- Did you include the license to the code and datasets? [Yes] See Section **??**.
- Did you include the license to the code and datasets? [No] The code and the data are proprietary.
- Did you include the license to the code and datasets? [N/A]

Please do not modify the questions and only use the provided macros for your answers. Note that the Checklist section does not count towards the page limit. In your paper, please delete this instructions block and only keep the Checklist section heading above along with the questions/answers below.

1. For all authors...

(a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
(b) Did you describe the limitations of your work? [Yes]
(c) Did you discuss any potential negative societal impacts of your work? [Yes]
(d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]

2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes]
   (b) Did you include complete proofs of all theoretical results? [Yes]

3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] We will include in the supplementary material.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] We will include in the supplementary material.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] We will include in the supplementary material.

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [N/A]
   (b) Did you mention the license of the assets? [N/A]
   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]

5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]