Enhancing Reinforcement Learning Agents with Local Guides

Paul Daoudi*
Huawei Noah's Ark Lab
Paris, France
paul.daoudi1@huawei.com

Bogdan Robu
GIPSA-Lab
Grenoble, France
bogdan.robu@gipsa-lab.grenoble-inp.fr

Christophe Prieur
GIPSA-Lab
Grenoble, France
christophe.prieur@gipsa-lab.fr

Ludovic Dos Santos*
Criteo AI Lab
Paris, France
l.dossantos@criteo.com

Merwan Barlier*
Huawei Noah's Ark Lab
Paris, France
merwan.barlier@huawei.com

ABSTRACT

This paper addresses the problem of integrating local guide policies into a Reinforcement Learning agent. For this, we show how to adapt existing algorithms to this setting before introducing a novel algorithm based on a noisy policy-switching procedure. This approach builds on a proper Approximate Policy Evaluation (APE) scheme to provide a perturbation that carefully leads the local guides towards better actions. We evaluated our method on a set of classical Reinforcement Learning problems, including safety-critical systems where the agent cannot enter some areas at the risk of triggering catastrophic consequences. In all the proposed environments, our agent proved to be efficient at leveraging those policies to improve the performance of any APE-based Reinforcement Learning algorithm, especially in its first learning stages.

KEYWORDS

Reinforcement Learning, Local Guides, External Knowledge, Sample Efficiency

ACM Reference Format:
Paul Daoudi*, Bogdan Robu, Christophe Prieur, Ludovic Dos Santos*, and Merwan Barlier*. 2023. Enhancing Reinforcement Learning Agents with Local Guides. In *Proc. of the 22nd International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2023), London, United Kingdom, May 29 – June 2, 2023*, IFAAMAS, 10 pages.

1 INTRODUCTION

Reinforcement Learning (RL) aims at learning an optimal policy in an unknown environment by interacting with it. This discipline has known many successes on a wide range of simulated systems from recommender systems [66, 88] to complex video games [20, 55]. Despite some advances in real-world environments such as balloon navigation [8] or plasma control in Tokamaks [16], RL is not ready to be applied to most real-world applications. Too many challenges must first be resolved [18, 85]. In particular, the agent requires many interactions with the system to learn a good policy, which can be prohibitive in various cases: running real-world experiments may be more time - or money - consuming by many orders of magnitude compared to simulations, or when failing would eventually damage the system. In fact, these interactions may be completely restricted in the case of safety-critical environments, as they would have catastrophic consequences on themselves or their surroundings if an inadequate action is chosen in the wrong state. Hence, the agent must be able to learn with a small, potentially narrow, amount of samples.

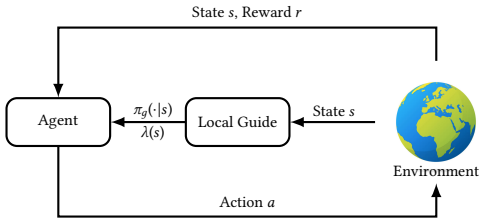


Figure 1: Reinforcement Learning with Local Guides
At each step, the agent receives a state s, the reward r of the previous state-action pair, as well as the guide policy $\pi_g(\cdot|s)$ with its associated confidence function $\lambda(s)$.

One of the main reasons for this requirement is the poor quality of the data sampled by the agent at the beginning of learning: an improper initialization of the policy drives the agent to visit meaningless or even catastrophic parts of the environment. In the former case, the agent would need a significant number of interactions with the system before discovering suitable areas of the environment. This results in running a mediocre policy for a long period, which is unacceptable on most real-world systems. For instance, a traditional way of teaching a robot how to walk would result in a substantial amount of falling, which would eventually damage it. Another example is the control of the cooling system of a server room: forcing the room to stay at a high temperature would damage the servers. This problem is all the more pronounced in high-dimensional environments with sparse rewards where relevant information are located in a narrow subspace of the environment. Real-life exploration is similar to what can be encountered when playing Montezuma's Revenge, an arcade game notoriously difficult to solve [7, 9] because of its sparse signals provided only after completing specific series of actions over extended periods.

Consequently, satisfactory RL agents must be able to quickly discover meaningful information about the environment with carefully chosen interactions with the system, that are both informative and safe. In the literature, a common way to go towards this purpose

*Core Contributors

Proc. of the 22nd International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2023), A. Ricci, W. Yeoh, N. Agmon, B. An (eds.), May 29 – June 2, 2023, London, United Kingdom.

is to consider information provided by a guide policy (potentially sub-optimal). One proposition is to rely on external data that has been gathered with this guide to properly initialize any RL agent [45, 51, 56], and/or to lead the subsequent optimization process [64, 71]. This has been successful in solving complex tasks, including the first levels of Montezuma's Revenge [6, 61]. However, the suggested algorithms require a substantial amount of data to be efficient [57] which can be difficult to get in some real-world environments where sampling is expensive. In fact, data can be nearly impossible to gather from safety-critical environments since states leading to disastrous repercussions cannot be visited. Another line of work directly considers having access to the global sub-optimal guide which avoids the need for sampling from a costly environment. It acts as a teacher for the RL agent [89], which can either be *attractive* [3, 72, 82] to lead the agent towards meaningful areas, or *repulsive* [81, 84] to prevent it from entering unsafe states. Although presenting considerable improvements, almost none of the methods introducing a guide tackles the policy initialization with the exception of [82], or [3] at the cost of initial sampling.

More importantly, both of these families assume access to a guide that would be efficient in all the state space of the environment, named in this paper as *global guide*. Considering the increasing complexity of real-world systems, such global information becomes too difficult to build using classical tools [44, 77]. Nonetheless, information can often still be extracted, regardless of the complexity of the environment. For instance, experts may build heuristics such as catch-up policies to avoid going into unsafe states, or, when the dynamics can be inferred on some parts of the system, apply some of the many tools from Optimal Control (OC) theory [37, 79] for building attractive policies to guide the agent towards relevant information. In particular, Gain Scheduling [47, 69] is thoroughly used in the industry [4, 15, 23] to provide controllers that are relevant only at the vicinity of chosen operating points.

Inspired by these real-world considerations, we propose to address a novel framework called *Reinforcement Learning with Local Guides* (RLLG), that seeks to reduce the number of interactions between the RL agent and the environment. Within this setting, the agent must find a suitable global policy in the entire state-action space with the help of a controller that would be relevant only in a known region of the state space, named *local guide*. This extension is representative of real-world demands and includes any kind of local controller - whether attractive or repulsive - that may be present in real-world systems. This setting generalizes to any external policy as it also encompasses global ones.

In this paper, we first present how to adapt the Approximate Policy Iteration scheme to take an external policy into account to the RLLG setting, and analyze their advantages and drawbacks. Then, we introduce a novel algorithm to cope with the presented weaknesses and validate our approach on a variety of complex environments in two different use cases. The various algorithms are first compared to guide the exploration process to improve the quality of the gathered data during the first episodes. They are further tested to prevent the agent from entering catastrophic states. In both cases, the relevance of introducing local controllers as well as the advantages of our method are outlined in the proposed environments.

2 RELATED WORK

Before diving into the RLLG setting, we present how expertise has previously been integrated into the RL framework. This expert information is often global and can take different forms: data, reward, or policy. We cover them below and include the few existing approaches leveraging local information in the last paragraph.

Imitation Learning with Demonstrations. Many works focused on a setting where the agent has access to a large amount of demonstration trajectories from a guide of the task. A first proposition to recover the policy of the guide is Behavioral Cloning (BC) [75] which uses the traditional Supervised Learning scheme. When the dataset is complete and comes from an expert policy, this technique was successful in special cases of autonomous driving [62] and flying [70]. However, when the agent finds itself in a situation not described by the dataset, it may choose catastrophic actions and its performance can quickly degrade [14]. In fact, Ross and Bagnell [67] show that this distribution shift induces a quadratic error in the length of the episode for standard Supervised Learning algorithms, so Ross et al. [68] proposed DAGGER, an approach mixing the expert and the learnt policy, to get a linear error. Among the many different Imitation Learning frameworks lies Inverse RL [1, 5, 33] which attempts at correcting the weaknesses of BC by encoding the expert policy into a reward function that could be optimized. This approach has shown impressive results in a wide variety of environments, including high-dimensional ones [11, 32, 87]. Nonetheless, this set of techniques seeks to reproduce and generalize the guide policy, not improve it. It can be problematic when the guide is sub-optimal.

Reinforcement Learning from Demonstrations (RLfD). Instead of copying and generalizing the guide policy from the provided dataset, RLfD [64, 71] uses RL tools to find a better policy. In practice, the additional data is mostly used to overcome unnecessary exploration and lead the agent towards interesting parts of the MDP [57]. They were originally employed to initialize the policy with BC [40, 60], and then throughout the entire optimization process. For example, DQNfD [31] and DDPGfD [83] include the demonstration data in the replay buffer and add new regularizations to force the optimization process to better consider the demonstrations. In a similar fashion, POfD [36] and LOGO [65] force the policy of the agent to stay close to the guide policy by penalizing or constraining the RL objective. However, these methods are only possible when the expert policy is attractive as no data could be retrieved near states having catastrophic consequences. Besides, they might require substantial data in order to guide the RL agent [31, 57, 65], which can be difficult to obtain on systems that are costly to sample.

Reward Shaping. An additional intuitive idea to include domain knowledge in the RL process is to modify the reward function [17, 52], but an arbitrary reward shaping is dangerous as it might deviate the agent from its original goal [63]. Thus, Ng et al. [58] proposed Potential Based Reward Shaping, a framework that constrains the added rewards to be in a specific potential form to guarantee for the original goal to be unchanged, though finding such potential from any external knowledge is difficult [29]. A clever alternative strategy is to add the heuristic in the Q-function instead [38, 39]

which has fewer chances of deviating from the original goal, and/or to decrease the impact of the heuristic over time [12].

Reinforcement Learning with a Global Guide. Closely related to our work is to transfer the knowledge of a known teacher [89] - or guide policy - to the RL agent [72, 82]. Despite the need of building a global guide, this circumvents the problem of acquiring a large amount of data. Similarly than in RLfD, the guide policy leads the RL agent in its first learning stages [39, 72]. More recently, Uchendu et al. [82] and Agarwal et al. [3] took advantage of the access to a global guide to properly initialize an RL policy, even though [3] introduced QDagger, an algorithm that is pre-trained on the data sampled by the teacher that could be unavailable in practice. Rather than finding a global policy that would be better than the global guide, Jacq et al. [34] proposed the Lazy-MDPs framework to learn to take over the guide only when it is noticeably sub-optimal. While it provides interesting insights into the properties of the environment, it does not prioritize sample efficiency.

Local Expertise. All the above methods rely on a global guide or heuristic that can be difficult to acquire. Very few works focused on local ones. One approach is COG [74] which leverages data from a sub-task to guide the agent in the RL optimization process. However, it suffers from the drawbacks of relying on data discussed in the previous paragraph and is only applied in the sparse reward setting. Most related to our work is to directly consider having access to local guides. It has been studied in two cases. First, when the local controller is optimal, as the one for those built with Optimal Control (OC) theory [37, 49, 53], some works proposed a switching mechanism between the RL and the local controller [10, 26, 46, 90]. Such switching cannot overcome the sub-optimality of the local policy. Another line of research considered emergency procedures that are activated when the agent enters a dangerous part of the environment to prevent the agent from entering catastrophic states [81, 84]. In addition to the local controller, Turchetta et al. [81] assume a curriculum is available to maintain safety, and Wagener et al. [84] rely on an advantage estimate of the local policy which might not be available in practice. Contrary to these works, our method does not require any additional heavy-to-build knowledge.

3 PRELIMINARIES AND PROBLEM SETTING

3.1 Preliminaries

Let $\Delta(\mathcal{X})$ be the set of all probability measures on $\mathcal{X}$. The agent-environment framework is modeled as a Markov Decision Process (MDP) $(\mathcal{S}, \mathcal{A}, r, P, \rho, \gamma)$. It is composed of a state space $\mathcal{S}$, an action space $\mathcal{A}$, a transition kernel $P : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$, a reward function $r : \mathcal{S} \times \mathcal{A} \to [R_{min}, R_{max}]$, an initial state distribution ρ and a discount factor $\gamma \in [0, 1)$. We focus on the general setting where $\mathcal{A}$ is continuous and propose an extension to handle a discrete action space in Appendix **??**.

A policy $\pi : \mathcal{S} \to \Delta(\mathcal{A})$ is a decision rule mapping a state over a distribution of actions. The RL objective is to find a policy maximizing the expected discounted cumulative reward $G_t = \sum_{i=0}^{\infty} \gamma^t r(s_{t+i}, a_{t+i})$ over the distribution induced by the policy π and the transition kernel P. The value of a policy π is measured through the value function $V^\pi(s) = \mathbb{E}_P \left[G_t \mid s_t = s,\ a_{t+i} \sim \pi(\cdot|s_{t+1}) \forall i \geq 0) \right]$ and its associated Q-value function $Q^\pi(s, a) = \mathbb{E}_P \left[G_t \mid s_t = s,\ a_t = a, a_{t+i} \sim \pi(\cdot|s_{t+i}) \ \forall i \geq 1 \right]$. Let V^* and Q^* be the optimal value functions associated with the highest expected cumulative rewards.

Let the Bellman operator for the Q-value function $\mathcal{B}^\pi \left[Q \right](s, a) = r(s, a) + \gamma \mathbb{E}_{s' \sim P(\cdot|s,a), a' \sim \pi(\cdot|s')} \left[Q(s', a') \right]$, and the Bellman optimality operator $\mathcal{B}^* \left[Q \right](s, a) = r(s, a) + \gamma \mathbb{E}_{s' \sim P(\cdot|s,a)} \left[\max_{a'} Q(s', a') \right]$. Being γ-contractions, iteratively applying these operators to any initial Q^0 converges to their associated fixed points Q^π or Q^*.

3.2 Approximate Policy Iteration

In view of the usual high dimensions of the state-action space and the lack of access to the environment and its transition kernel, these operators cannot be computed. Instead, the RL approach uses data to approximate them. We denote $\hat{\mathcal{B}}^\pi$ and $\hat{\mathcal{B}}^*$ the associated empirical Bellman operators that use samples to estimate the expectation under $P(\cdot|s, a)$. A general framework to find a good policy is Approximate Policy Iteration, which is at the core of many state-of-the-art algorithms [21, 28, 50, 54, 73] that are efficient on environments with a continuous action space. As we will see in the following section, such scheme is particularly relevant to the RRLG setting.

At each epoch k, the agent collects data with the current policy π^k, evaluates its associated Q-function via Approximate Policy Evaluation (APE) and improves its policy through Approximate Policy Improvement (API). Given a dataset $\mathcal{D} = \{(s_i, a_i, r_i, s_{i+1})_{i=1}^N\}$, $\hat{\mathbb{E}}$ the empirical expectation of the state-action pair (s, a) induced by $\mathcal{D}$, $\omega \in \Omega$ and $\theta \in \Theta$ the respective parameters of Q and π, $\bar{\omega}_k \in \Omega$ the frozen weights associated to the Q-target, it is formalized as:

$$Q_\omega^{k+1} \leftarrow \arg\min_{\omega \in \Omega} \hat{\mathbb{E}} \left[\left(Q_\omega - \hat{\mathcal{B}}^{\pi_\theta^k} \left[Q_{\bar{\omega}_k}^k \right] \right)^2 \right], \tag{APE}$$

$$\pi_\theta^{k+1} \leftarrow \arg\max_{\theta \in \Theta} J_{\pi_\theta}^{\mathcal{D}} \left(Q_\omega^{k+1} \right). \tag{API}$$

The objective $J_{\pi_\theta}^{\mathcal{D}} \left(Q_\omega^{k+1} \right)$ in the Approximate Policy Improvement step API may vary depending on the RL algorithm, but is always a function of the estimated Q-values.

Using this process greedily could lead to poor policies [35, 75], especially at the beginning of learning when the estimates are inaccurate. The agents thus need to explore the environment to gather relevant data that would improve these estimations. Since little information about the environment is known, this exploration often relies on random noise added to the policy. It allows the discovery of interesting parts of the environment, but can also very well guide the agent towards meaningless regions. Even worse, this random exploration might lead the agent to catastrophic states that would be unacceptable in real-world systems.

3.3 Problem setting

The goal is to find a good policy on the entire state space with as few interactions as possible with the environment while avoiding catastrophic states in a setting where the system has been previously studied by experts. The idea is to integrate local state space expertise from, *e.g.* any available source, to lead and possibly constrain the RL optimization process. This information is formalized as a local guide $\pi_g : \mathcal{S} \to \Delta(\mathcal{A})$ that is relevant only in a potentially small region of the state space $\mathcal{S}_g \in \mathcal{S}$. Note this controller may be the concatenation of N different local guides that are relevant

Method	Good initialization	Hyper-parameter tuning	Overcome bad guide	Safe agent
Strict Action Guided (SAG)	✓	✓	✗	✓
Reward Guided (RG)	✗	✗	✓	✗
Policy Improvement Guided (PIG)	✗	✗	✓	✗
Parameterized Action Guided (PAG)	✓	✓	✓	✓

Table 1: Advantages and Drawbacks of the different agents in the Reinforcement Learning with Local Guides (RLLG) setting

in non-overlapping regions of the state space (S_g^i for expert i), in which case $\mathcal{S}_g = \bigcup_{i=1}^N S_g^i$.

Along with the local heuristic, we consider having access to a function $\lambda : \mathcal{S} \to [0, 1]$ reporting the confidence of the local guide in state s. This could be a binary function when perfect knowledge about the local policy and the environment is known, where $\lambda(s) = 1$ if $s \in \mathcal{S}_g$ else $\lambda(s) = 0$. While this parameter may not be known with precision, it can be estimated by practitioners.

We do not assume that the local guide is optimal as this is not necessarily the case in real-world applications. This relaxes the need to know with precision the confidence function λ. Additionally, we do not assume a combination of the local guides can solve the task, hence the necessity of introducing RL.

In addition, this work is agnostic to the type of the guide policy, *e.g.* stochastic or deterministic, but for the rest of the paper, we consider the realistic setting where the guide policy is deterministic that may be prevalent in the real world. It is unlikely that experts design randomized policies that are then applied in real-world systems: actions of the tail distribution of the controller may harm the system. We thus denote a_g^s the only action the guide policy outputs in the state s. The extension to stochastic policies is straightforward as a_g^s could be replaced with an expectation over $\pi_g(\cdot|s)$.

4 REINFORCEMENT LEARNING WITH LOCAL GUIDES

In this section, we first introduce three straightforward ways to adapt APE RL algorithms to our setting and analyze their advantages and disadvantages. We then propose a new simple yet efficient approach to retain the identified advantages while avoiding the presented drawbacks.

4.1 Classical integration of the local controller

Strict Action Guided (SAG). To include the local controller in this setting, one can simply use it when available. Thanks to the provided indicator function $\lambda(\cdot)$, the agent will switch between the different policies when the guide is judged sufficiently relevant, that is when $\lambda(s) \geq \lambda^-$. The global policy at iteration k, π_{SAG}^k, can be written:

$$\pi_{\text{SAG}}^k(\cdot|s) = \begin{cases} a_g^s & \text{if } \lambda(s) \geq \lambda^-, \\ \pi_\theta^k(\cdot|s) & \text{otherwise.} \end{cases} \tag{1}$$

This method is most appropriate when the local policy is optimal, or when the sub-optimality of the local controller is deemed sufficient for the problem at hand. Applying this switching mechanism seems intuitive and has been successfully used in prior works [26, 90]. However, the use of Deep Neural Networks combined with a bootstrapped target loss in the (APE) step might lead to a distributional shift previously studied in the offline setting [21, 43, 48]. Indeed, when evaluating the Q-function of a policy π with a dataset that has been gathered with a policy far from the evaluated one, using π to bootstrap its target would wrongly and continually back-propagate bad estimates. To cope with this issue, we propose to estimate the Q-values of the switched policy π_{SAG}^k instead of π_θ^k, simply by building the target with $\hat{\mathcal{B}}^{\pi_{\text{SAG}}^k}$ instead of $\hat{\mathcal{B}}^{\pi_\theta^k}$. This step would be impossible in the Approximate Value Iteration scheme, comforting our choice of focusing on API-based algorithms.

Reward Guided (RG). While the previous policy definitely provides a boost in learning, especially for initializing the policy and making sure it respects the potential constraints of the environment, the agent is stuck with the potential sub-optimality of the guide. A first way to cope with this is to shape the reward to include the local information [38, 39]. Let $\mathcal{M}_{\pi_\theta^k}^{\pi_g}(s)$ be a behavioral cloning (BC) metric that reports how close the policy of the agent π_θ^k is to the guide policy π_g. Different BC metrics can be used depending on the considered setting, *e.g.* $-\|a - a_g^s\|^2$ when π_θ^k is deterministic, or $\log \pi_\theta^k(a_g^s|s)$ when the density of the policy is available. Given these notations, considering a hand-crafted scheduler β_{RG}^k, the shaped reward can be written as:

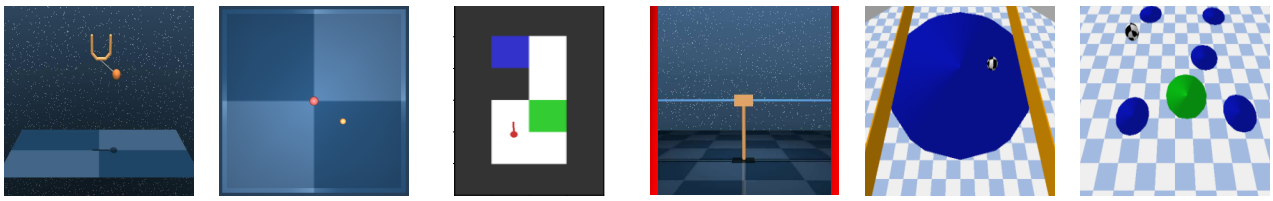
$$\tilde{r}^k(s, a, \pi_g, \pi_\theta^k, \lambda) = r(s, a) + \beta_{\text{RG}}^k \, \lambda(s) \, \mathcal{M}_{\pi_\theta^k}^{\pi_g}(s). \tag{2}$$

However, as discussed in the related works section, playing with the rewards is hazardous as it might lead to a completely modified goal [59]. Besides, the impact of the metric $\mathcal{M}_{\pi_\theta^k}^{\pi_g}(s)$ is difficult to control when it is added to the reward function. This phenomenon was observed in our experiments where we did not get good results, see Appendix **??**.

Policy Improvement Guided (PIG). Another approach to guide the RL agent with the local controller is to constrain the optimization process, in particular in the Approximate Policy Improvement step [39, 42, 86]. It is formalized as a Trust Region approach to make sure the agent explores at the right location:

$$\begin{aligned} &\underset{\theta \in \Theta}{arg\,max} && J_{\pi_\theta}^{\mathcal{D}}\left(Q_\omega^{k+1}\right) \\ &\text{subject to} && \hat{\mathbb{E}}_{s\sim\mathcal{D}}\left[\lambda(s)\mathcal{M}_{\pi_\theta}^{\pi_g}(s)\right] \leq M_k, \end{aligned} \tag{3}$$

where M_k is an unknown constant that would have to be determined by the practitioner. However, when learning policies are parameterized by Deep Neural Networks, this optimization problem is difficultly solved analytically. Following the regularized RL framework [25], practitioners resort to solving the relaxed Lagrangian optimization problem:



(a) Ball-in-Cup (b) Point-Mass (c) Point-Fall (d) Safe Cartpole (e) Point-Circle (f) Point-Reach

Figure 2: Environment visualizations.

$$\pi_\theta^{k+1} \leftarrow \arg\max_{\theta\in\Theta} J_{\pi_\theta}^{\mathcal{D}}\left(Q_\omega^{k+1}\right) + \beta_{\text{PIG}}^k \hat{\mathbb{E}}_{s\sim\mathcal{D}}\left[\lambda(s)\mathcal{M}_{\pi_\theta}^{\pi_g}(s)\right]. \tag{4}$$

In the literature β_{PIG}^k is either kept constant [42, 86] or slightly decayed over the episodes [3, 36, 72]. This controller integration is actually a state-the-art method to guide the RL policy when a global sub-optimal guide is available and was first proposed in [72]. Other approaches to constrain a policy to remain close to another exist [41, 65], though they all rely on solving a Lagrangian problem, which would suffer from the same drawbacks as the presented method.

4.2 Relaxing the guide action strict focus

In the previous section, we discussed the advantages and drawbacks of the three introduced methods to integrate a local controller: SAG would benefit from a jump start but would not be able to outperform a sub-optimal guide, whereas RG and PIG could eventually improve it. However, in RG, shaping the reward is hard and presents the risk of deviating the agent from its original goal. The exact Trust Region procedure of Equation (3) would be a good solution to answer this problem. Indeed, setting a small M_k at the early stages of learning makes sure the learnt policy begins near the local guide, and gradually increasing it would allow a clever and safe exploration of the environment. However, the relaxed Lagrangian approach is a relaxation of the constrained optimization problem so the agent is very likely to go beyond the trust region. When it is crucial to stay close to the local controller, this is not acceptable.

In this section, we propose a novel method to take the best of all approaches: it has a good policy initialization, sets a well-defined constraint on the closeness between the learnt and guide policies while still being able to overcome the sub-optimality of the guide. Note that our agent takes advantage of the continuous structure of the action space, but we propose an extension to the discrete setting in Appendix **??**.

Perturbed Action Guided (PAG). We propose to keep the Action Guided approach from SAG that integrates the local controller action into a global policy to enjoy a good initialization. Although, in a similar fashion to [22], we introduce a parameterized perturbation ξ_ϕ^k to gradually improve the local controller and overcome its limitations. The agent would be able to visit interesting regions of the state space while in practice respecting safety constraints encoded in the guide actions a_g^s. Formally, the parameterized perturbation $\xi_\phi(\cdot|s, a_g^s, \Phi) \in [-\Phi, \Phi]$, with $\phi \in \Xi$ takes as arguments the state, the guide action and a bound Φ over the action space. This perturbation slightly transforms the guide action allowing close exploration and eventually improving the guide policy. The global policy at iteration k, π_{PAG}^k, can be written:

$$\pi_{\text{PAG}}^k(\cdot|s) = \begin{cases} a_g^s + \beta_{\text{PAG}}^k\, \xi_\phi^k(\cdot|s, a_g^s, \Phi) & \text{if } \lambda(s) \geq \lambda^-, \\ \pi_\theta^k(\cdot|s) & \text{otherwise.} \end{cases} \tag{5}$$

Where β_{PAG}^k is a scheduler introduced to further control the weight of the parameterized perturbation ξ_ϕ^k. Intuitively, β_{PAG}^k should be set close to 0 in the early stages of learning to enjoy a good policy initialization thanks to the local controller, and gradually increase to 1 as the perturbation ξ_ϕ^k gets more relevant.

The bound Φ on the perturbation ξ_ϕ^k can be chosen by the practitioner depending on the nature of the environment. For instance, when safety is at stake, practitioners might choose a small Φ to remain close to the guide actions. When it is not, Φ could be increased to have a wider exploration and eventually improve the guide policy.

Thanks to the Approximate Policy Iteration structure (APE-API), ξ_ϕ^k can directly be trained to maximize Q_ω^{k+1}, that is a global estimate of the Q-values of the global policy π_{PAG}^k:

$$\xi_\phi^{k+1} \leftarrow \arg\max_{\phi\in\Xi} J_{\xi_\phi}^{\mathcal{D}}\left(Q_\omega^{k+1}\right), \tag{6}$$

with $J_{\xi_\phi}^{\mathcal{D}}\left(Q_\omega^{k+1}\right) = \hat{\mathbb{E}}_{s\sim\mathcal{D}, a'\sim\xi_\phi(\cdot|s,a_g^s,\Phi)}\left[\lambda(s)\, Q_\omega^{k+1}(s, a_g^s + a')\right]$. See Algorithm 1 for a pseudo-code description of our proposed method PAG.

Algorithm 1 Perturbed Action Guided (PAG)

Initialize Q_ω^0, π_θ^0, and ξ_ϕ^0
for $k \in (1, \ldots, K)$ **do**
  Gather data with π_{PAG}^k from Eq. (5)
  Add data to the replay buffer $\mathcal{D}$
  Sample a batch from $\mathcal{D}$
  Update Q_ω^{k+1} with gradient descent on Eq. (APE) with $\hat{\mathcal{B}}^{\pi_{PAG}^k}$
  Update π_θ^{k+1} with gradient ascent on Eq. (API)
  Update ξ_ϕ^{k+1} with gradient ascent on Eq. (6)
end for

5 EXPERIMENTS

In this section, we evaluate the performances of previously introduced methods that integrate local controllers in the RL framework

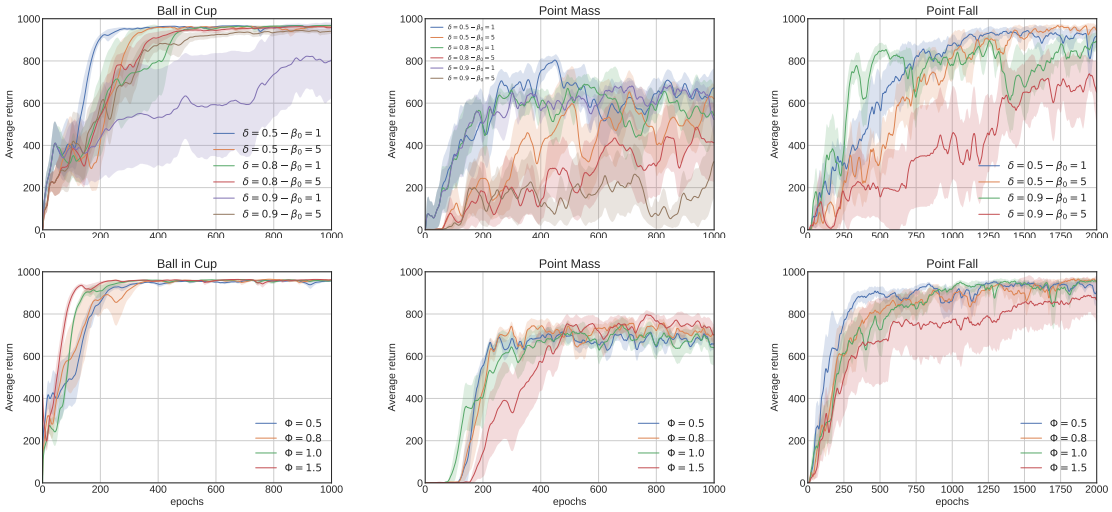


Figure 3: Hyper-parameter analysis of PIG (top) and PAG (bottom) on environments with attractive policies.

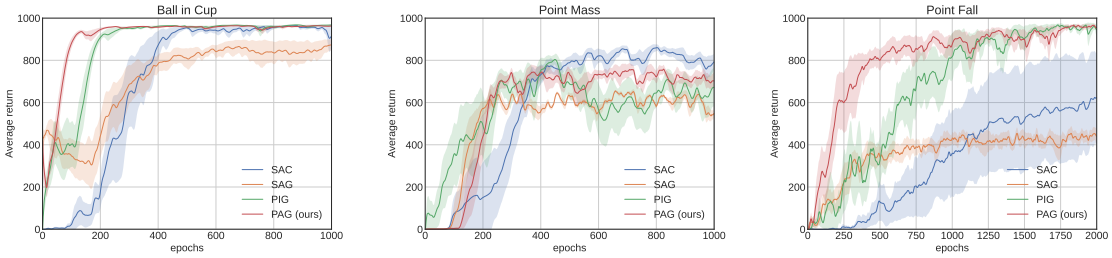


Figure 4: Overall performances comparing PAG with SAC, SAG and PIG on 3 different environments with attractive policies.

in two different settings. In the first one, the guide leads the decision-making process to maximize the performance of the agent. We denote those local controllers *attractive policies*. In the second one, the guide should prevent the agent from entering dangerous zones of the environment using conservative repulsive guide policies. We denote those local controllers *repulsive policies*.

Figure description. In all figures, the x-axis represents the number of epochs from 0 to K, which corresponds to 1000 interactions with the system as well as 1000 gradient updates of the different networks. The y-axis represents the averaged cumulative return over 5 evaluation trials. All plots represent an exponential smoothed average of 5 runs, and the shaded areas correspond to half of their standard deviation.

Agents. We conduct our experiments using Soft Actor Critic (SAC) [28], although our method can be used on any Approximate Policy Evaluation-based RL algorithm. Reward-Guided agents' performances are deferred in Appendix **??** as they did not provide interesting results.

Chosen hyper-parameters. All hyper-parameters are based on the default hyper-parameters of SAC except for the size of the hidden layers and the activation functions of the networks. Both the policy and the Q-functions are Feed-Forward Neural Networks with 2 layers of 64 neurons for all the guided environments and 2 layers of 32 neurons for the safety-critical environments. Besides, all networks have a ReLU activation with the exception of the Q-Network used in *Safe Cartpole Swingup* that has a TanH activation which produced better results. All additional hyperparameters were optimized using a grid search, and are detailed in the next sections. The Pytorch code of this work can be found in https://github.com/huawei-noah/HEBO/RLLG, and a MindSpore implementation has been released in https://gitee.com/mindspore/models/tree/master/research/rl/RLLG.

Metric. Along with the final performance of the agent and the respect of the safety constraints, we attentively pay attention to its initial performance. Thus, we compare the different agents using the normalized *Area-Under-the-Curve* (AUC) as in [30, 78] of the averaged cumulative performance of the agent during the total number of epochs. The AUC is normalized by the AUC of the perfect agent, that is the agent constantly having the optimal cumulative reward, chosen in this work as the best reward observed by the best agent. With an agent A, $CR(k)$ the cumulative return of the agent at epoch k and CR^* being the best CR of the best agent, it is formalized as:

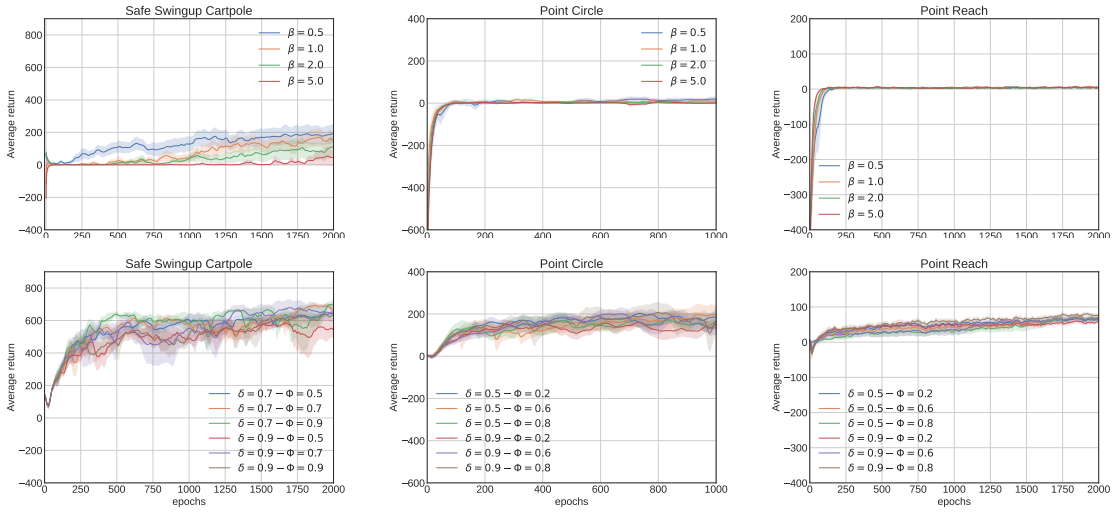


Figure 5: Hyper-parameter analysis of PIG (top) and PAG (bottom) on environments with repulsive policies.

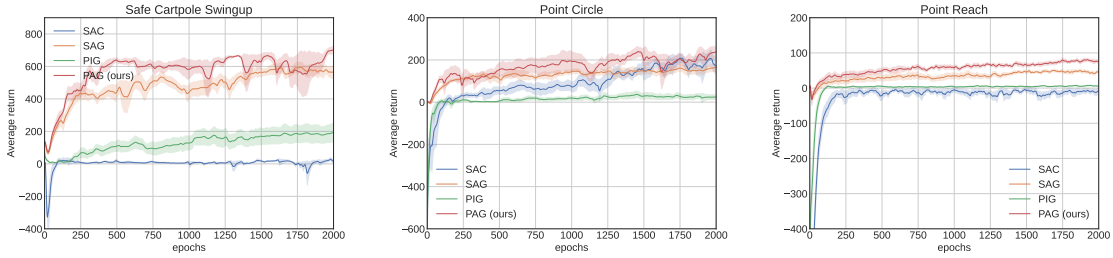


Figure 6: Overall performances comparing PAG with SAC, SAG and PIG on 3 different environments with repulsive policies.

$$AUC(A) = \frac{\int_{k=0}^{K-1} CR(k)\, dk}{K * CR^*}. \tag{7}$$

Those results are summarized in Table 2.

We perform the analysis of the different agents on different environments from MuJoCo [80] in the Deepmind Control Suite [76] and PyBullet [19] in Bullet-Safety-Gym [27]. Additional details regarding the experimental protocol can be found in Appendix **??**.

5.1 Local exploration with attractive policies

We first consider local controllers to guide the agent on some parts of the environment, so the agent can quickly access relevant information and focus on the most difficult parts of the environment using standard RL. The objective is to learn a good policy with as few interactions as possible.

In this setting, safety is not at stake, there is no need to gradually increase the impact of the parameterized perturbation ξ_ϕ^k. Hence, the scheduler $\{\beta_{\text{PAG}}^k\}_{k=0}^K$ is set to 1 in all experiments and, the scheduler $\{\beta_{\text{PIG}}^k\}_{k=0}^K$ starts at 1 and is gradually decreased with a multiplicative factor δ^κ, with κ an integer starting at 0 and increased by 1 every 50 epochs. The added heuristic has little to no impact at the end of learning (i.e $\kappa = 20$) for the PIG agent.

5.1.1 Environments. In this part, we focused on *Ball in Cup*, *Point-Mass*, and *Point-Maze*. The local guide is always a SAC agent stopped during mid-training. A complete description of these environments and guides is available in Appendix **??**, and can be visualized in Figure 2.

5.1.2 Instability of standard approaches. First, we perform an extensive analysis of the current state-of-the-art PIG agent on different environments. This empirical study highlights the difficulty of applying the PIG agent in a real-world setting. Figure 3 clearly emphasizes that PIG can perform well with a proper scheduler $\{\beta_{\text{PIG}}^k\}_{k=0}^K$. It is able to quickly find a near-optimal policy with a limited amount of samples. However, this method depends heavily on the choice of $\{\beta_{\text{PIG}}^k\}_{k=0}^K$: it requires the right initial β_{PIG}^0 with the right decay rate δ. This phenomenon is notably visible on *Point-Mass* in Figure 3 where the performance of the PIG agent is unstable. In fact, the guidance is even detrimental with some schedulers on these environments and might prevent the agent from finding a near-optimal policy. To the best of our knowledge, there currently exists no way of knowing an appropriate scheduler in advance.

This motivated our work to introduce a more stable and efficient algorithm.

5.1.3 Robustness of our PAG agent to the hyper-parameter choices. As opposed to existing approaches, our proposed algorithm PAG is more robust on the choice of its hyper-parameter Φ. On all environments, Figure 3 attests that the performance of the agent scarcely depends on the choice of Φ. Even setting $\Phi = 1.5$, where the role of the local expert is reduced as the resulting controller would cover all of the action space, is useful to guide the agent in its first learning stages. The worst case scenario is to have the same performance as the standard RL agent, notably seen on *Point-Mass*. Lower choices of Φ all lead to better results as they fully take advantage of the expertise provided by the local policy.

5.1.4 Comparison of the different agents. We include the standard RL agent (SAC) which does not use the local information and compare the best versions of the different guided agents in Figure 6. In the proposed environments, all guided agents profit from a better initialization that the unguided one. However, SAG is not able to find a near-optimal policy because of its direct dependency on the local expert. Both the other guided agents - PIG and PAG - find a near-optimal policy much faster thanks to a clever integration of the local guide. In addition, our agent PAG turns out to be the most efficient approach in all the environments.

5.2 Safe exploration with repulsive policies

Another crucial application including a local policy in the process is learning without any safety violations, that is learning without exploring states that may have catastrophic consequences. While many successful approaches using the CMDP framework have been proposed [2, 13, 24], we discard them as they need to see the constraint signal to build a safe policy. In a setting where this signal only provides information when these constraints are violated, it should never be seen. Hence, we investigate if it is possible to never see any constraint violation during learning with the help of an overly conservative emergency procedure that prevents the agent from entering such states.

In this case, the agent should always stay close to the emergency procedure, so we set the scheduler $\{\beta^k_{\text{PIG}}\}_{k=0}^{K}$ to be fixed over learning for the PIG agent. Regarding our PAG agent, we set β^0_{PAG} at 0, and gradually increase it to 1 following $(1 - \delta^\kappa)$, with κ an integer starting at 0 and increased by 1 every 50 epochs.

5.2.1 Environments. We focused on *Safe Cartpole Swingup*, *Point-Circle* and *Point-Reach*, and the local guides are scripted emergency procedures. Similarly, the description of the environments can be found in Appendix **??** and visualized in Figure 2.

5.2.2 Inefficiency of existing approaches. Similar to the previous section, we investigate how the state-of-the-art method PIG performs in this setting. We observe in Figure 5 that it is difficult to force the RL agent to stay within the safe zone only with a penalty on the API step. In fact, most of the chosen hyper-parameters β in most environments lead to safety violations at the beginning of learning in most environments. All the more surprising is that the choice of a relevant β is complicated. For instance, in the *Safe Cartpole Swingup* environment, the only agent that respects the constraints is when $\beta_{PIG} = 0.5$. Higher choices of β_{PIG} ($\beta_{\text{PIG}} \in [1, 2, 5]$) visit catastrophic regions of the environment. Once again, this analysis empirically underlines the necessity of introducing a new method.

Table 2: Normalized-AUC on the different environments.

Environment	SAC	SAG	PIG	PAG (ours)
ball-in-cup	0.73 ± 0.06	0.74 ± 0.08	0.9 ± 0.01	$\mathbf{0.95 \pm 0.01}$
point-mass	$\mathbf{0.72 \pm 0.06}$	0.63 ± 0.03	0.71 ± 0.1	$\mathbf{0.72 \pm 0.02}$
point-fall	0.34 ± 0.27	0.38 ± 0.01	0.70 ± 0.08	$\mathbf{0.85 \pm 0.06}$
cartpole	0.01 ± 0.01	0.69 ± 0.04	0.18 ± 0.09	$\mathbf{0.82 \pm 0.06}$
point-circle	0.39 ± 0.13	0.55 ± 0.1	0.06 ± 0.06	$\mathbf{0.71 \pm 0.19}$
point-reach	-0.29 ± 0.1	0.43 ± 0.12	0.01 ± 0.01	$\mathbf{0.71 \pm 0.09}$
Total Mean	0.32	0.57	0.43	**0.8**

5.2.3 Robustness of our PAG agent. Our proposed PAG agent possesses an important hyper-parameter that allows a strict closeness with the emergency procedure if needed: Φ. Hence, with a properly chosen Φ, the agent never enters dangerous parts of the environment but is still able to overcome the conservativeness of the emergency procedure if needed. In addition, Figure 5 attests that the performance PAG is robust to the choice of Φ, as long as this one does not exceed a certain threshold.

5.2.4 Comparison of the different agents. In *Safe Cartpole Swingup* and *Point-Reach* environments, the classical RL agent SAC only learns not to go into catastrophic states, it does not solve the task. In *Point-Circle*, it is able to do so but at the cost of visiting catastrophic states. The PIG agent violates the constraints less often than the unguided agent, but still consistently breaks them at the beginning of learning. In the *Safe Cartpole Swingup* environment, it may be able to avoid breaking these constraints, but it is rather out of luck than from a proper choice of hyper-parameter. On the other hand, the agents that utilize the local policy at a higher level, *e.g.* SAG and PAG agents, are successful at keeping the agent safe throughout learning. In addition, our PAG agent is also able to improve on the SAC agent that suffers from an overly conservative emergency procedure and find a better policy than the other methods.

6 CONCLUSION

We formalized a novel setting that generalizes access to any kind of controller that may be available in real-world systems. We especially study the introduction of local controllers to enhance any Approximate Policy Iteration-based RL agent thanks to a novel algorithm. We studied two important use cases: guiding the agent towards meaningful regions of the environment and preventing it from entering dangerous state spaces. Our method was stable, robust, and more efficient than standard approaches applied in this setting. In addition, we allow a good initialization and a strict constraint on the closeness with respect to the local controller that may be important on safety critical systems to learn a good policy with zero safety violations.

However, in all the tested environments, we relied on a known confidence function that details the relevance of the local controller in simple environments. This may be a strong assumption in more complex systems that would require a finer analysis, which will be the focus of our future works.

REFERENCES

[1] Pieter Abbeel and Andrew Y Ng. 2004. Apprenticeship learning via inverse reinforcement learning. In *International conference on machine learning*. 1.

[2] Joshua Achiam, David Held, Aviv Tamar, and Pieter Abbeel. 2017. Constrained policy optimization. In *International conference on machine learning*. PMLR, 22–31.

[3] Rishabh Agarwal, Max Schwarzer, Pablo Samuel Castro, Aaron Courville, and Marc G Bellemare. 2022. Beyond Tabula Rasa: Reincarnating Reinforcement Learning. *arXiv preprint arXiv:2206.01626* (2022).

[4] Eugenio Alcala, Vicenç Puig, Joseba Quevedo, and Teresa Escobet. 2018. Gain-scheduling LPV control for autonomous vehicles including friction force estimation and compensation mechanism. *IET Control Theory & Applications* 12, 12 (2018), 1683–1693.

[5] Saurabh Arora and Prashant Doshi. 2021. A survey of inverse reinforcement learning: Challenges, methods and progress. *Artificial Intelligence* 297 (2021), 103500.

[6] Yusuf Aytar, Tobias Pfaff, David Budden, Thomas Paine, Ziyu Wang, and Nando De Freitas. 2018. Playing hard exploration games by watching youtube. *Advances in neural information processing systems* 31 (2018).

[7] Marc Bellemare, Sriram Srinivasan, Georg Ostrovski, Tom Schaul, David Saxton, and Remi Munos. 2016. Unifying count-based exploration and intrinsic motivation. *Advances in neural information processing systems* 29 (2016).

[8] Marc G Bellemare, Salvatore Candido, Pablo Samuel Castro, Jun Gong, Marlos C Machado, Subhodeep Moitra, Sameera S Ponda, and Ziyu Wang. 2020. Autonomous navigation of stratospheric balloons using reinforcement learning. *Nature* 588, 7836 (2020), 77–82.

[9] Marc G Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling. 2013. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research* 47 (2013), 253–279.

[10] Ju-Seung Byun and Andrew Perrault. 2021. Training Transition Policies via Distribution Matching for Complex Tasks. *arXiv preprint arXiv:2110.04357* (2021).

[11] Xin-Qiang Cai, Yao-Xiang Ding, Yuan Jiang, and Zhi-Hua Zhou. 2019. Imitation learning from pixel-level demonstrations by hashreward. *arXiv preprint arXiv:1909.03773* (2019).

[12] Ching-An Cheng, Andrey Kolobov, and Adith Swaminathan. 2021. Heuristic-guided reinforcement learning. *Advances in Neural Information Processing Systems* 34 (2021), 13550–13563.

[13] Yinlam Chow, Ofir Nachum, Aleksandra Faust, Edgar Dueñez-Guzman, and Mohammad Ghavamzadeh. 2021. Safe Policy Learning for Continuous Control. In *Conference on Robot Learning*. PMLR, 801–821.

[14] Felipe Codevilla, Eder Santana, Antonio M. López, and Adrien Gaidon. 2019. Exploring the Limitations of Behavior Cloning for Autonomous Driving. In *ICCV*. IEEE, 9328–9337.

[15] Pedro Henrique Silva Coutinho and Reinaldo Martínez Palhares. 2021. Dynamic periodic event-triggered gain-scheduling control co-design for quasi-LPV systems. *Nonlinear Analysis: Hybrid Systems* 41 (2021), 101044.

[16] Jonas Degrave, Federico Felici, Jonas Buchli, Michael Neunert, Brendan Tracey, Francesco Carpanese, Timo Ewalds, Roland Hafner, Abbas Abdolmaleki, Diego de Las Casas, et al. 2022. Magnetic control of tokamak plasmas through deep reinforcement learning. *Nature* 602, 7897 (2022), 414–419.

[17] Marco Dorigo and Marco Colombetti. 1994. Robot shaping: Developing autonomous agents through learning. *Artificial intelligence* 71, 2 (1994), 321–370.

[18] Gabriel Dulac-Arnold, Nir Levine, Daniel J Mankowitz, Jerry Li, Cosmin Paduraru, Sven Gowal, and Todd Hester. 2021. Challenges of real-world reinforcement learning: definitions, benchmarks and analysis. *Machine Learning* 110, 9 (2021), 2419–2468.

[19] Benjamin Ellenberger. 2018–2019. PyBullet Gymperium. https://github.com/benelot/pybullet-gym.

[20] Florian Fuchs, Yunlong Song, Elia Kaufmann, Davide Scaramuzza, and Peter Dürr. 2021. Super-human performance in gran turismo sport using deep reinforcement learning. *IEEE Robotics and Automation Letters* 6, 3 (2021), 4257–4264.

[21] Scott Fujimoto, Herke Hoof, and David Meger. 2018. Addressing function approximation error in actor-critic methods. In *International conference on machine learning*. PMLR, 1587–1596.

[22] Scott Fujimoto, David Meger, and Doina Precup. 2019. Off-policy deep reinforcement learning without exploration. In *International conference on machine learning*. PMLR, 2052–2062.

[23] Antonio J Gallego, Gonzalo M Merello, Manuel Berenguel, and Eduardo F Camacho. 2019. Gain-scheduling model predictive control of a Fresnel collector field. *Control Engineering Practice* 82 (2019), 1–13.

[24] Javier Garcıa and Fernando Fernández. 2015. A comprehensive survey on safe reinforcement learning. *Journal of Machine Learning Research* 16, 1 (2015), 1437–1480.

[25] Matthieu Geist, Bruno Scherrer, and Olivier Pietquin. 2019. A theory of regularized markov decision processes. In *International conference on machine learning*. PMLR, 2160–2169.

[26] Sean Gillen, Marco Molnar, and Katie Byl. 2020. Combining deep reinforcement learning and local control for the acrobot swing-up and balance task. In *2020 59th IEEE Conference on Decision and Control (CDC)*. IEEE, 4129–4134.

[27] Sven Gronauer. 2022. Bullet-Safety-Gym: a framework for constrained Reinforcement Learning. (2022).

[28] Tuomas Haarnoja, Aurick Zhou, Kristian Hartikainen, George Tucker, Sehoon Ha, Jie Tan, Vikash Kumar, Henry Zhu, Abhishek Gupta, Pieter Abbeel, et al. 2018. Soft actor-critic algorithms and applications. *arXiv preprint arXiv:1812.05905* (2018).

[29] Anna Harutyunyan, Sam Devlin, Peter Vrancx, and Ann Nowé. 2015. Expressing Arbitrary Reward Functions as Potential-Based Advice. In *Proceedings of the AAAI Conference on Artificial Intelligence*. AAAI Press, 2652–2658.

[30] Matteo Hessel, Joseph Modayil, Hado Van Hasselt, Tom Schaul, Georg Ostrovski, Will Dabney, Dan Horgan, Bilal Piot, Mohammad Azar, and David Silver. 2018. Rainbow: Combining improvements in deep reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

[31] Todd Hester, Matej Vecerik, Olivier Pietquin, Marc Lanctot, Tom Schaul, Bilal Piot, Dan Horgan, John Quan, Andrew Sendonaris, Ian Osband, et al. 2018. Deep q-learning from demonstrations. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 32.

[32] Jonathan Ho and Stefano Ermon. 2016. Generative adversarial imitation learning. *Advances in neural information processing systems* 29 (2016).

[33] Ahmed Hussein, Mohamed Medhat Gaber, Eyad Elyan, and Chrisina Jayne. 2017. Imitation Learning: A Survey of Learning Methods. *ACM Comput. Surv.* 50, 2 (2017), 21:1–21:35.

[34] Alexis Jacq, Johan Ferret, Olivier Pietquin, and Matthieu Geist. 2022. Lazy-MDPs: Towards Interpretable RL by Learning When to Act. In *International Conference on Autonomous Agents and Multiagent Systems*. 669–677.

[35] Chi Jin, Zeyuan Allen-Zhu, Sebastien Bubeck, and Michael I Jordan. 2018. Is Q-learning provably efficient? *Advances in neural information processing systems* 31 (2018).

[36] Bingyi Kang, Zequn Jie, and Jiashi Feng. 2018. Policy optimization with demonstrations. In *International conference on machine learning*. PMLR, 2469–2478.

[37] Hassan K Khalil. 2002. *Nonlinear systems; 3rd ed.* Prentice-Hall, Upper Saddle River, NJ. https://cds.cern.ch/record/1173048 The book can be consulted by contacting: PH-AID: Wallet, Lionel.

[38] W Bradley Knox and Peter Stone. 2010. Combining manual feedback with subsequent MDP reward signals for reinforcement learning.. In *International Conference on Autonomous Agents and Multiagent Systems*. 5–12.

[39] W Bradley Knox and Peter Stone. 2012. Reinforcement learning from simultaneous human and MDP reward.. In *International Conference on Autonomous Agents and Multiagent Systems*. 475–482.

[40] Jens Kober and Jan Peters. 2008. Policy search for motor primitives in robotics. *Advances in neural information processing systems* 21 (2008).

[41] Ilya Kostrikov, Rob Fergus, Jonathan Tompson, and Ofir Nachum. 2021. Offline reinforcement learning with fisher divergence critic regularization. In *International conference on machine learning*. PMLR, 5774–5783.

[42] Aviral Kumar, Justin Fu, Matthew Soh, George Tucker, and Sergey Levine. 2019. Stabilizing off-policy q-learning via bootstrapping error reduction. *Advances in Neural Information Processing Systems* 32 (2019).

[43] Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. 2020. Conservative q-learning for offline reinforcement learning. *Advances in Neural Information Processing Systems* 33 (2020), 1179–1191.

[44] Jean B Lasserre, Didier Henrion, Christophe Prieur, and Emmanuel Trélat. 2008. Nonlinear optimal control via occupation measures and LMI-relaxations. *SIAM journal on control and optimization* 47, 4 (2008), 1643–1666.

[45] Seunghyun Lee, Younggyo Seo, Kimin Lee, Pieter Abbeel, and Jinwoo Shin. 2022. Offline-to-online reinforcement learning via balanced replay and pessimistic q-ensemble. In *Conference on Robot Learning*. PMLR, 1702–1712.

[46] Youngwoon Lee, Shao-Hua Sun, Sriram Somasundaram, Edward S Hu, and Joseph J Lim. 2018. Composing complex skills by learning transition policies. In *International Conference on Learning Representations*.

[47] Douglas J Leith and William E Leithead. 2000. Survey of gain-scheduling analysis and design. *International journal of control* 73, 11 (2000), 1001–1025.

[48] Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. 2020. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv preprint arXiv:2005.01643* (2020).

[49] Frank L Lewis, Draguna Vrabie, and Vassilis L Syrmos. 2012. *Optimal control*. John Wiley & Sons.

[50] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. 2015. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971* (2015).

[51] Jiafei Lyu, Xiaoteng Ma, Xiu Li, and Zongqing Lu. 2022. Mildly conservative Q-learning for offline reinforcement learning. *arXiv preprint arXiv:2206.04745* (2022).

[52] Maja J Mataric. 1994. Reward functions for accelerated learning. In *Machine learning proceedings 1994*. Elsevier, 181–189.

[53] David Q Mayne, James B Rawlings, Christopher V Rao, and Pierre OM Scokaert. 2000. Constrained model predictive control: Stability and optimality. *Automatica* 36, 6 (2000), 789–814.

[54] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. 2016. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*. PMLR, 1928–1937.

[55] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. 2013. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602* (2013).

[56] Ashvin Nair, Abhishek Gupta, Murtaza Dalal, and Sergey Levine. 2020. Awac: Accelerating online reinforcement learning with offline datasets. *arXiv preprint arXiv:2006.09359* (2020).

[57] Ashvin Nair, Bob McGrew, Marcin Andrychowicz, Wojciech Zaremba, and Pieter Abbeel. 2018. Overcoming exploration in reinforcement learning with demonstrations. In *2018 IEEE international conference on robotics and automation (ICRA)*. IEEE, 6292–6299.

[58] Andrew Y Ng, Daishi Harada, and Stuart Russell. 1999. Policy invariance under reward transformations: Theory and application to reward shaping. In *International conference on machine learning*, Vol. 99. 278–287.

[59] Andrew Y. Ng, Daishi Harada, and Stuart J. Russell. 1999. Policy Invariance Under Reward Transformations: Theory and Application to Reward Shaping. In *International conference on machine learning*. Morgan Kaufmann, 278–287.

[60] Jan Peters and Stefan Schaal. 2008. Reinforcement learning of motor skills with policy gradients. *Neural networks* 21, 4 (2008), 682–697.

[61] Tobias Pohlen, Bilal Piot, Todd Hester, Mohammad Gheshlaghi Azar, Dan Horgan, David Budden, Gabriel Barth-Maron, Hado Van Hasselt, John Quan, Mel Večerík, et al. 2018. Observe and look further: Achieving consistent performance on atari. *arXiv preprint arXiv:1805.11593* (2018).

[62] Dean A Pomerleau. 1988. Alvinn: An autonomous land vehicle in a neural network. *Advances in neural information processing systems* 1 (1988).

[63] Jette Randløv and Preben Alstrøm. 1998. Learning to Drive a Bicycle Using Reinforcement Learning and Shaping.. In *International conference on machine learning*, Vol. 98. Citeseer, 463–471.

[64] Harish Ravichandar, Athanasios S Polydoros, Sonia Chernova, and Aude Billard. 2020. Recent advances in robot learning from demonstration. *Annual Review of Control, Robotics, and Autonomous Systems* 3 (2020), 297–330.

[65] D Rengarajan, G Vaidya, A Sarvesh, D Kalathil, and S Shakkottai. 2022. Reinforcement Learning with Sparse Rewards using Guidance from Offline Demonstration. In *International Conference on Learning Representations)*.

[66] Pornthep Rojanavasu, Phaitoon Srinil, and Ouen Pinngern. 2005. New recommendation system using reinforcement learning. *Special Issue of the Intl. J. Computer, the Internet and Management* 13, SP 3 (2005).

[67] Stéphane Ross and Drew Bagnell. 2010. Efficient reductions for imitation learning. In *International conference on artificial intelligence and statistics*. JMLR Workshop and Conference Proceedings, 661–668.

[68] Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. 2011. A reduction of imitation learning and structured prediction to no-regret online learning. In *International conference on artificial intelligence and statistics*. JMLR Workshop and Conference Proceedings, 627–635.

[69] Wilson J Rugh and Jeff S Shamma. 2000. Research on gain scheduling. *Automatica* 36, 10 (2000), 1401–1425.

[70] Claude Sammut, Scott Hurst, Dana Kedzier, and Donald Michie. 1992. Learning to Fly. In *ML*. Morgan Kaufmann, 385–393.

[71] Stefan Schaal. 1996. Learning from demonstration. *Advances in neural information processing systems* 9 (1996).

[72] Simon Schmitt, Jonathan J Hudson, Augustin Zidek, Simon Osindero, Carl Doersch, Wojciech M Czarnecki, Joel Z Leibo, Heinrich Kuttler, Andrew Zisserman, Karen Simonyan, et al. 2018. Kickstarting deep reinforcement learning. *arXiv preprint arXiv:1803.03835* (2018).

[73] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347* (2017).

[74] Avi Singh, Albert Yu, Jonathan Yang, Jesse Zhang, Aviral Kumar, and Sergey Levine. 2020. Cog: Connecting new skills to past experience with offline reinforcement learning. *arXiv preprint arXiv:2010.14500* (2020).

[75] Richard S. Sutton and Andrew G. Barto. 1998. *Reinforcement learning - an introduction*. MIT Press.

[76] Yuval Tassa, Yotam Doron, Alistair Muldal, Tom Erez, Yazhe Li, Diego de Las Casas, David Budden, Abbas Abdolmaleki, Josh Merel, Andrew Lefrancq, et al. 2018. Deepmind control suite. *arXiv preprint arXiv:1801.00690* (2018).

[77] Andrew J Taylor, Victor D Dorobantu, Sarah Dean, Benjamin Recht, Yisong Yue, and Aaron D Ames. 2021. Towards robust data-driven control synthesis for nonlinear systems with actuation uncertainty. In *2021 60th IEEE Conference on Decision and Control (CDC)*. IEEE, 6469–6476.

[78] Yee Teh, Victor Bapst, Wojciech M Czarnecki, John Quan, James Kirkpatrick, Raia Hadsell, Nicolas Heess, and Razvan Pascanu. 2017. Distral: Robust multitask reinforcement learning. *Advances in neural information processing systems* 30 (2017).

[79] Emanuel Todorov. 2006. Optimal control theory. *Bayesian brain: probabilistic approaches to neural coding* (2006), 268–298.

[80] Emanuel Todorov, Tom Erez, and Yuval Tassa. 2012. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ international conference on intelligent robots and systems*. IEEE, 5026–5033.

[81] Matteo Turchetta, Andrey Kolobov, Shital Shah, Andreas Krause, and Alekh Agarwal. 2020. Safe reinforcement learning via curriculum induction. *Advances in Neural Information Processing Systems* 33 (2020), 12151–12162.

[82] Ikechukwu Uchendu, Ted Xiao, Yao Lu, Banghua Zhu, Mengyuan Yan, Joséphine Simon, Matthew Bennice, Chuyuan Fu, Cong Ma, Jiantao Jiao, et al. 2022. Jump-Start Reinforcement Learning. *arXiv preprint arXiv:2204.02372* (2022).

[83] Mel Vecerik, Todd Hester, Jonathan Scholz, Fumin Wang, Olivier Pietquin, Bilal Piot, Nicolas Heess, Thomas Rothörl, Thomas Lampe, and Martin Riedmiller. 2017. Leveraging demonstrations for deep reinforcement learning on robotics problems with sparse rewards. *arXiv preprint arXiv:1707.08817* (2017).

[84] Nolan C Wagener, Byron Boots, and Ching-An Cheng. 2021. Safe reinforcement learning using advantage-based intervention. In *International conference on machine learning*. PMLR, 10630–10640.

[85] Zhe Wang and Tianzhen Hong. 2020. Reinforcement learning for building controls: The opportunities and challenges. *Applied Energy* 269 (2020), 115036.

[86] Yifan Wu, George Tucker, and Ofir Nachum. 2019. Behavior regularized offline reinforcement learning. *arXiv preprint arXiv:1911.11361* (2019).

[87] Huang Xiao, Michael Herman, Joerg Wagner, Sebastian Ziesche, Jalal Etesami, and Thai Hong Linh. 2019. Wasserstein adversarial imitation learning. *arXiv preprint arXiv:1906.08113* (2019).

[88] Guanjie Zheng, Fuzheng Zhang, Zihan Zheng, Yang Xiang, Nicholas Jing Yuan, Xing Xie, and Zhenhui Li. 2018. DRN: A Deep Reinforcement Learning Framework for News Recommendation. In *WWW*. ACM, 167–176.

[89] Matthieu Zimmer, Paolo Viappiani, and Paul Weng. 2014. Teacher-student framework: a reinforcement learning approach. In *AAMAS Workshop Autonomous Robots and Multirobot Systems*.

[90] Samuele Zoboli, Vincent Andrieu, Daniele Astolfi, Giacomo Casadei, Jilles S Dibangoye, and Madiha Nadri. 2021. Reinforcement learning policies with local LQR guarantees for nonlinear discrete-time systems. In *2021 60th IEEE Conference on Decision and Control (CDC)*. IEEE, 2258–2263.