Machine Learning (2022) 111:1431–1521
https://doi.org/10.1007/s10994-022-06144-5

Lipschitzness is all you need to tame off-policy generative adversarial imitation learning

Lionel Blondé[1] · Pablo Strasser[1] · Alexandros Kalousis[1]

Received: 1 August 2020 / Revised: 18 January 2022 / Accepted: 27 January 2022 /
Published online: 4 April 2022

Abstract
Despite the recent success of reinforcement learning in various domains, these approaches remain, for the most part, deterringly sensitive to hyper-parameters and are often riddled with essential engineering feats allowing their success. We consider the case of off-policy generative adversarial imitation learning, and perform an in-depth review, qualitative and quantitative, of the method. We show that forcing the learned reward function to be local Lipschitz-continuous is a *sine qua non* condition for the method to perform well. We then study the effects of this necessary condition and provide several theoretical results involving the local Lipschitzness of the state-value function. We complement these guarantees with empirical evidence attesting to the strong positive effect that the consistent satisfaction of the Lipschitzness constraint on the reward has on imitation performance. Finally, we tackle a generic pessimistic reward preconditioning add-on spawning a large class of reward shaping methods, which makes the base method it is plugged into provably more robust, as shown in several additional theoretical guarantees. We then discuss these through a fine-grained lens and share our insights. Crucially, the guarantees derived and reported in this work are valid for *any* reward satisfying the Lipschitzness condition, nothing is specific to imitation. As such, these may be of independent interest.

Keywords Imitation learning · Reinforcement learning · Lipschitz-continuity · Generative adversarial networks · Deep learning

1 Introduction

Imitation learning (IL) (Bagnell 2015) sets out to design artificial agents able to adopt a behavior demonstrated via a set of expert-generated trajectories. Also referred to as *“teaching by showing”* (Schaal 1997), IL can replace tedious tasks such as manual hard-coded agent programming, or hand-crafted reward design *“reward shaping”* (Ng et al. 1999) for

Editor: Scott Sanner.

✉ Lionel Blondé
lionel.blonde@gmail.com

[1] University of Geneva, Geneva School of Business Administration - HES-SO, Geneva, Switzerland

the agent to be trained via reinforcement learning (RL) (Sutton and Barto 1998). Besides, in contrast with the latter, imitation learning does not necessarily involve agent-environment interactions. This feature is particularly appealing in real-world domains such as robotics (Atkeson and Schaal 1997; Schaal 1997; Ratliff et al. 2007; Billard et al. 2008), where the artificial agent is physically implemented with expensive hardware, and the environment contains enough external entities (e.g. humans, other artificial agents, other costly devices) to raise safety concerns (Ha et al. 2020; Kahn 2016; Ray et al. 2019; Held et al. 2017). When controls are provided in the demonstrations [or recovered via inverse dynamics from the available kinematics (Hanna and Stone 2017)], we can treat said controls as regression targets, and learn a mimicking policy with a simple, supervised approach. This interaction-free approach (simulated or physical, real-world interactions), called *behavioral cloning* (BC), has enabled the success of various endeavors in robotic manipulation and locomotion (Ratliff et al. 2007; Wang et al. 2017), in autonomous driving—with the first self-driving vehicle (Pomerleau 1989, 1990) thirty years ago and more recently with (Gu et al. 2020) using Waymo's open dataset (Sun et al. 2019)—and also in grand challenges like AlphaGo (Silver et al. 2016) and AlphaStar (Vinyals et al. 2019). Due to its conceptual simplicity, we expect BC to still be a part of the pipeline for the most ambitious enterprises going forward, especially as open datasets get slowly released.

Despite its practical advantages, BC is extremely data-hungry w.r.t. the amount of expert demonstrations it needs to yield robust, high-fidelity policies. Besides, unless corrective behavior is present in the dataset (e.g. in autonomous driving, how to drive back onto the road), the policy learned via BC will not be able to internalize this behavior. Once in a situation from which it can not recover, there will be a permanent *covariate shift* between its current observations and the demonstrated ones. The controls learned in a supervised manner on the expert dataset are therefore useless, due to the distributional shift. As a result, the agent's errors will compound, a phenomenon coined by Ross and Bagnell (2010) as *compounding errors*. In Sect. 6.2.3, we stress how the latter echoes the *compounding variations* phenomenon, exhibited as part of the theoretical contributions of this work. To address the shortcomings of BC, Abbeel and Ng (2004) proposes to harness the innate credit assignment (Sutton and Barto 1998) capabilities of RL, by first trying to learn the cost function underlying the demonstrated behavior [inverse RL (Ng et al. 2000)], before using this cost to optimize a policy via RL. The succession of inverse RL and RL is called apprenticeship learning (AL) (Abbeel and Ng 2004), and can, by design, yield policies that can recover from out-of-distribution situations thanks to RL's built-in temporal abstraction mechanisms. Cost learning however is incredibly tedious, and successful approaches end up requiring coarse relaxations to avoid being deterringly computationally-expensive (Abbeel and Ng 2004; Syed and Schapire 2008; Syed et al. 2008; Ho et al. 2016). Ultimately, as noted by Ziebart et al. (2008), setting out to recovering the cost signal under which the expert demonstrations are optimal (base assumption of inverse RL) is an ill-posed objective—echoing the reward shaping considerations from Ng et al. (1999). In line with this statement, generative adversarial imitation learning (GAIL) (Ho and Ermon 2016) departs from the typical AL pipeline, and replaces learning the optimal cost ("optimal" in the inverse RL sense) by learning a *surrogate* cost function. GAIL does so by leveraging generative adversarial networks (Goodfellow et al. 2014), as the name hints. The method is described in greater detail in Sect. 3. Due to the RL step it involves (like any AL method), GAIL suffers from poor sample-efficiency w.r.t. the amount of interactions it needs to perform with the environment. This caveat has since been addressed, notably by transposition to the off-policy setting, concurrently in SAM (Blondé and Kalousis 2019) and DAC (Kostrikov et al. 2019) (cf. Sect. 4). Both adversarial IL methods leverage actor-critic architectures, consequently suffering from a greater exposure to instabilities. These weaknesses are mitigated with various complementary techniques, and cautious hyper-parameter tuning.

In this work, we set out to first conduct a thorough theoretical and empirical investigation into off-policy generative adversarial imitation learning, to pinpoint which are the techniques that are instrumental in performing well, and shed light over which are ones that can be discarded or disregarded without decrease in performance. Ultimately, we would like to exhibit the techniques that are *sufficient* for the method to achieve peak performance. Virtually every algorithmic design choice made in this work is supported by an ablation study reported in the Appendix. We start by describing the base off-policy adversarial imitation learning method at the core of this work in Sect. 4. We then undertake diagnoses of the various issues that arise from the combination of bilevel optimization problems at the core of the investigated model in Sect. 5. A key contribution of our work consists in showing that enforcing a Lipschitzness constraint on the learned surrogate reward is a *necessary* condition for the method to even learn anything—in our consumer-grade, computationally affordable hardware setting. We study it closely, providing empirical evidence of the importance of this constraint through detailed ablation results in Sect. 5.5. We follow up on this empirical evidence with theoretical results in Sect. 6.1, characterizing the Lipschitzness of the state-action value function under said reward Lipschitzness condition, and discuss the obtained variation bounds subsequently. Crucially, we show that without variation bounds on the reward, a phenomenon we call *compounding variations* can cause the variations of the state-action value to explode. As such, the theoretical results reported in Sect. 6.1—and discussed in Sect. 6.2—corroborate the empirical evidence exhibited in Sect. 5.5. *Note, the theoretical results reported in this work are valid for any reward satisfying the condition, they readily transfer to the general RL setting and are not specific to imitation.* The theoretically-grounded Lipschitzness condition, implemented as a gradient penalty, is in practice a *local* Lipschitzness condition. We therefore investigate *where* (i.e. on which samples, on which input distribution) the local Lipschitzness regularization should be enforced. We propose a new interpretation of the regularization scheme through an RL perspective, make an intuitively grounded claim on where to enforce the constraint to get the best results, and corroborate our claim empirically (cf. Sect. 6.3). Crucially, we show that the consistent satisfaction of the Lipschitzness constraint on the reward is a strong predictor of how well the mimicking agent performs empirically (cf. Sect. 6.4). Finally, we introduce a generic pessimistic reward preconditioner which makes the base method it is plugged into provably more robust, as attested by its companion guarantees (cf. Sect. 6.5). *Again, these guarantees are not not specific to imitation and can be of independent interest for the RL community.* Among the reported insights, we give an illustrative example of how the simple technique can further increase the robustness of the method it is plugged into.

2 Related work

Off-policy generative adversarial imitation learning, which is the object of this work, involves learning a parametric surrogate reward function, from expert demonstrations. By design (Ho and Ermon 2016; Blondé and Kalousis 2019; Kostrikov et al. 2019), this signal is learned at the same time as the policy, and is therefore subject to non-stationarities (cf. Sect. 5.2). This reward regime is reminiscent of the *reward corruption* phenomenon (Everitt et al. 2017; Romoff et al. 2018), which posits that the real-world rewards are imperfect (e.g. uncontrolled task specification change, sensor defects, reward hacking) and must therefore be treated as such, i.e. non-stationary at the very least. Despite being learned and therefore liable to non-stationary behavior, our reward is internal—as opposed to outside the agent's and practitioner's scope—and is therefore fully observable, as well as controllable via the practitioner-specified algorithmic design. The reward corruption can consequently be acted upon, and more easily mitigated than if it originated from a *black box* reward originating from the unknown environment.

The demonstrations on the other hand are available from the very beginning, and do not change as the policy learns. In that respect, our approach differs from *observational learning* (Borsa et al. 2017), where the policy learns to imitate another by *observing* it itself learn in the environment—and therefore does not strictly qualify as an expert at the task. Observational learning draws clear parallels with the teacher-student scheme in policy distillation (Rusu et al. 2015). While our reward is changing since the policy changes and due to the inherent learning dynamics of function approximators, in observational learning, the reward would be changing also due to the expert still learning, causing a distributional drift.

Multi-armed bandits (Robbins 1952) have received a lot of attention in recent years to formalize and model problems of sequential decision making under uncertainty. In the context of this work, the most appropriate variants of bandits are *stateful* contextual multi-armed bandits. As the name hints, such models formalize decision making specific to given situations (i.e. contexts, states), in which the situations are *i.i.d.*-sampled. We consider the case of reinforcement learning, where the situations are entangled, along with the decisions themselves, in a Markov decision process (cf. Sect. 3). In particular, non-stationary reward channels in Markov decision processes have been studied extensively (cf. Sect. 5.2). Among these, adversarial bandits (Auer et al. 1995) can be seen as the archetype or worst-case reward corruption scenario, in which an adversary—possibly driven by malevolent intents—decides on the reward given to the agent. In these models, the common way to deal with non-stationary reward processes is to assume the reward variations in time are upper-bounded, either per-decision or over longer time periods. We give a comprehensive account of sequential decision making under uncertainty in non-stationary Markov decision processes in Appendix 2. By contrast, our theoretical guarantees are built on the premise that the reward function's variations are bounded *over the input space* by assuming that the reward function is locally Lipschitz-continuous over it. We make the same assumption on the dynamics of the multi-stage decision process, as well as on the control policy. While our theoretical results ultimately characterize the value function's robustness in terms of Lipschitz-continuity, (Fonteneau et al. 2010, 2013) start from the same assumptions, propose an estimator of the expected return, and derive bounds on its bias and variance. Derived in the offline RL setting, their bounds increase as the *"dispersion"* of the offline dataset increases. As such, our findings and dicussions carried out in Sect. 6.2 echo their work.

Several works have recently attempted to address the overfitting problem GAIL suffers from. This is due to the discriminator being able to trivially distinguish agent-generated samples from expert-generated ones, which occurs when the learning dynamics of the adversarial game are not properly balanced. As such, the gist of said techniques is to either weaken the discriminator directly or make its classification task harder, which unsurprisingly exactly coincides with the typical techniques used to cope with overfitting in (binary) classification. These techniques are, in no particular order: reducing the discriminator's capacity—by plugging the classifier on top of an independent perception stack (e.g. random features, state-action value convolutional layers) (Reed et al. 2018), smoothing the positive labels with uniform random noise (Blondé and Kalousis 2019), adopting a positive-unlabeled classification objective (instead of the traditional positive-negative one) (Xu and Denil 2019), using a gradient penalty [originally from (Gulrajani et al. 2017)] regularizer (Blondé and Kalousis 2019; Kostrikov et al. 2019), leveraging an adaptive information bottleneck in the discriminator network (Peng et al. 2018), enriching the expert dataset via task-specific data augmentation (Zolna et al. 2019). In this work, we do not propose a new regularization technique. Instead, we perform an in-depth analysis of the simplest techniques—in terms of conceptual simplicity, implementation time, number of parameters, and computational cost (Hernandez and Brown 2020)—and ultimately find that the gradient penalty regularizer achieves the best trade-off.

A large-scale empirical study of adversarial imitation learning (Orsini et al. 2021), released very recently, considers a wide range of hyper-parameter settings, reporting results for more than 500k trained agents. The authors conclude that their study adds nuances to ours (this work). In particular, they argue that while the regularization techniques that urge the reward to be Lipschitz-continuous indeed do improve the performance (hence corroborating what we show in the first investigation of our work; cf. Sect. 5.5), more traditional regularizers (e.g. weight decay, dropout) can often perform similarly. In this work, we align the notion of smoothness with the Lipschitz-continuity of a function approximator, and are therefore focusing, from Sect. 5.5 onward, on gradient penalization because it *explicitly* enforces the reward to be smooth. More importantly, reward Lipschitzness is among the premises of our theoretical guarantees. In the results reported in (Orsini et al. 2021), the discriminator regularization schemes that can perform on par with schemes enforcing Lipschitz-continuity explicitly [gradient penalization (Gulrajani et al. 2017), and spectral normalization (Miyato et al. 2018)], which are always the top performers, are: dropout (Srivastava et al. 2014), weight decay (Loshchilov and Hutter 2017), and mixup (Zhang et al. 2017) (performing data augmentation). Regularization schemes such as dropout, weight decay, and data augmentation are less often seen through the lens of smoothness regularization than through the lens of generalization, despite generalization being among the beneficial effects of smoothness (Rosca et al. 2020). Used in the last layer, weight decay (Loshchilov and Hutter 2017) punishes spikes in elements of the weight matrix by limiting its norm, hence not allowing the output of the network to change too much. Dropout (Srivastava et al. 2014) applies masks over hidden activations, making the network return similar outputs when inputs only differ slighly. When using data augmentation [e.g. in mixup (Zhang et al. 2017)], the network is forced to be close-to-invariant to purposely crafted variations of the input. These regularizers do not enforce Lipschitzness over the input space as explicitly as gradient penalties and spectral normalization do; nevertheless, they do encourage Lipschitzness implicitly, making the predictor more robust as a result. Specifically, as noted in Gouk et al. (2021), when a neural function approximator is trained with dropout, the Lipschitz constant of each layer is multiplied by $1 - r$, where r is the dropout rate. It is also noted in Cisse et al. (2017) that using weight decay regularization at the last layer controls the Lipschitz constant of the network. All in all, the methods reported by Orsini et al. (2021) as performing the best are the ones enforcing Lipschitz-continuity over the input space explicitly, and these can be matched by regularization schemes that encourage Lipschitzness over the input space implicitly. As such, these results are complementary to the ones we report in our first investigation in Sect. 5.5, where we found that direct, explicit gradient penalization exceeds the performance of other evaluated regularizers. As we report, not constraining the Lipschitzness of the discriminator yields the worst results among the evaluated alternatives. Keeping the Lipschitz constant of the discriminator in check seems essential. Perhaps more importantly, the empirical investigation we conduct in Sect. 5.5, and that is complemented by Orsini et al. (2021), motivates the derivation of our novel theoretical guarantees. Through these, we provide insights as to *why* keeping the Lipschitz constant of the reward in check seems to play such an important role in the stability of the value in off-policy adversarial IL. The considerable computational budget spent in Orsini et al. (2021) attests to how challenging the tackled problem is.

Hafner et al. (2011) advocate for the use of a *smooth* reward signal in RL. Lange et al. (2012) presents it as one key method to make learning values in offline RL less tedious. Sharp changes in reward value are hard to represent and internalize by the action-value neural function approximator. Using a smooth reward surrogate derived from the original *"jumpy"* reward signal such that the trends are preserved but the crispness is attenuated proved instrumental empirically. Our observation about reward Lipschitz-continuity being a crucial component of our off-policy imitation learning pipeline is in line with the suggestion of Hafner et al. (2011). On top of providing empirical evidence of its benefits, we also provide a number of theoretical results characterizing what the reward smoothness does on the value function smoothness.

Finally, we point out that local Lipschitz-continuity conditions are also found in the adversarial robustness literature. Notably, Finlay et al. (2018) encourages Lipschitzness via gradient regularization, as is done in our work. Similarly, Hardt et al. (2015) derives bounds under a Lipschitz-continuity assumption on the loss.

3 Background

Setting In this work, we address the problem of an agent whose goal is, in the absence of extrinsic reinforcement signal (Singh et al. 2009), to *imitate* the behavior demonstrated by an expert (Bagnell 2015), expressed to the agent via a pool of trajectories. The agent is never told how well she performs or what the optimal actions are, and is not allowed to query the expert for feedback.

Preliminaries The intrinsic behavior of the decision maker is represented by the *policy* π_θ, modeled by a neural network with parameter θ, mapping states to probability distributions over actions. Formally, the conditional probability density over actions that the agent concentrates at action a_t in state s_t is denoted by $\pi_\theta(a_t|s_t)$, for all discrete timestep $t \geq 0$. We model the environment the agent interacts with as an infinite-horizon, memoryless, and stationary *Markov Decision Process* (MDP) (Puterman 1994) formalized as the tuple $\mathbb{M} := (\mathcal{S}, \mathcal{A}, p, \rho_0, u, \gamma)$. $\mathcal{S} \subseteq \mathbb{R}^n$ and $\mathcal{A} \subseteq \mathbb{R}^m$ are respectively the state space and action space. p and ρ_0 define the *dynamics* of the world, where $p(s_{t+1}|s_t, a_t)$ denotes the stationary conditional probability density concentrated at the next state s_{t+1} when stochastically transitioning from state s_t upon executing action a_t, and ρ_0 denotes the initial state probability density. u denotes a stationary *reward process* that assigns, to any state-actions pairs, a real-valued reward r_t distributed as $r_t \sim u(\cdot|s_t, a_t)$. Finally, $\gamma \in [0, 1)$ is the discount factor. We make the MDP *episodic* by positing the existence of an absorbing state in every trace of interaction and enforcing $\gamma = 0$ to formally trigger episode termination once the absorbing state is reached. Since our agent does not receive rewards from the environment, she is in effect interacting with an MDP lacking a reward process r. Our method however encompasses learning a surrogate reward parameterized by a deterministic function approximator such as a neural network with parameter φ, denoted by r_φ, and whose learning procedure will be reported subsequently. Consequently, our agent effectively interacts with the augmentation of the previous MDP defined as $\mathbb{M}^* := (\mathcal{S}, \mathcal{A}, p, \rho_0, r_\varphi, \gamma)$. A *trajectory* τ_θ is a trace of π_θ in $\mathbb{M}^*$, succession of consecutive *transitions* (s_t, a_t, r_t, s_{t+1}), where $r_t := r_\varphi(s_t, a_t)$. A *demonstration* is the set of state-actions pairs (s_t, a_t) extracted from a trajectory collected by the expert policy π_e in $\mathbb{M}$. The *demonstration dataset* $\mathcal{D}$ is a set of demonstrations.

Objective Building on the reward hypothesis at the core of reinforcement learning (any task can be defined as the maximization of a reward), to act optimally, our agents must be able to deal with delayed signals and maximize the long-term cumulative reward. To address credit assignment, we use the concept of *return*, the discounted sum of rewards from timestep t onwards, defined as $R_t^\gamma := \sum_{k=0}^{+\infty} \gamma^k r_{t+k} := \sum_{k=0}^{+\infty} \gamma^k r_\varphi(s_{t+k}, a_{t+k})$ in the infinite-horizon regime. By taking the expectation of the return with respect to all the future states and actions in $\mathbb{M}^*$, after selecting a_t in s_t and following π_θ thereafter, we obtain the state-action value (Q-value) of the policy π_θ at (s_t, a_t): $Q^{\pi_\theta}(s_t, a_t) := \mathbb{E}_{s_{t+1} \sim p(\cdot|s_t, a_t), a_{t+1} \sim \pi_\theta(\cdot|s_{t+1}), \ldots}[R_t^\gamma]$ (*abbrv.* $\mathbb{E}_{\pi_\theta}^{>t}[R_t^\gamma]$). At state s_t, a policy π_θ that picks a_t verifying:

$$a_t = \operatorname*{argmax}_{a \in \mathcal{A}} Q^{\pi_\theta}(s_t, a)$$

therefore acts optimally looking onwards from s_t. Ultimately, an agent acting optimally at all times maximizes $V^{\pi_\theta}(s_0) := \mathbb{E}_{a_0 \sim \pi_\theta(\cdot|s_0)}[Q^{\pi_\theta}(s_0, a_0)]$ for any given start state $s_0 \sim \rho_0$. *In fine*, we can now define the *utility function* [also called *performance objective* (Silver et al. 2014)] to which our agent's policy π_θ must be solution of: $\pi_\theta = \operatorname*{argmax}_{\pi \in \Pi} U_0(\pi)$ where $U_t(\pi) := V^\pi(s_t)$ and Π is the search space of parametric function approximators, i.e. deep neural networks.

Generative adversarial imitation learning GAIL (Ho and Ermon 2016) trains a binary classifier D_φ, called *discriminator*, where samples from π_e are positive-labeled, and those from π_θ are negative-labeled. It borrows its name from *Generative Adversarial Networks* (Goodfellow et al. 2014): the policy π_θ plays the role of generator and is optimized to fool the discriminator D_φ into classifying its generated samples (negatives), as positives. As such, the prediction value indicates to what extent D_φ believes π_θ's generations are coming from the expert, and therefore constitutes a good measure of mimicking success. GAIL does not try to recover the reward function that underlies the expert's behavior. Rather, it learns a similarity measure between π_e and π_θ, and uses it as a *surrogate* reward function. We say that π_θ and D_φ are *"trained adversarially"* to denote the two-player game they are intricately tied in: D_φ is trained to assert with confidence whether a sample has been generated by π_θ, while π_θ receives increasingly greater rewards as D_φ's confidence in said assertion lowers. *In fine*, the surrogate reward measures the confusion of D_φ. In this work, the neural network function approximator modeling D_φ uses a sigmoid as output layer activation, i.e. $D_\varphi \in [0, 1]$. The exact zero case is bypassed numerically for $\log \circ D_\varphi$ to always exist, by adding an infinitesimal value $\epsilon > 0$ to D_φ inside the logarithm. The same numerical stability trick is used for $\log \circ (1 - D_\varphi)$ to avoid the exact one case (cf. reward formulations in Sect. 4).

4 Comprehensive refresher on the sample-efficient adversarial mimic

Building on TRPO (Schulman et al. 2015), GAIL (Ho and Ermon 2016) inherits its policy evaluation subroutine, consisting in learning a parametric estimate of the state-value function $V_\omega \approx V^{\pi_\theta}$ via Monte-Carlo estimation over samples collected by π_θ. While it uses function approximation to estimate V^{π_θ}, hoping it generalizes better than a straight-forward non-parametric Monte-Carlo estimate (discounted sum), we will reserve the term *actor-critic* for architectures in which the state-value $V^{\pi_\theta}(\cdot)$ or Q-value $Q^{\pi_\theta}(\cdot,\cdot)$ is learned via Temporal-Difference (TD) (Sutton 1988). This terminology choice is adopted from Sutton and Barto (1998) (cf. Chapter 13.5). A *critic* is used for bootstrapping, as in the TD update rule (whatever the bootstrapping degree is). As such, TRPO is not an actor-critic, while algorithms learning their value via TD, such as DDPG (Silver et al. 2014; Lillicrap et al. 2016), are actor-critic architectures. Albeit hindered from various weaknesses (cf. Sect. 5.1), and forgetting for a moment that it is combined with function approximation (Sutton et al. 1999; Silver et al. 2014), the TD update is able to propagate information quicker as the backups are shorter and therefore do not need to reach episode termination to learn, in contrast with Monte-Carlo estimation. That is without even involving fictitious, memory, or experience replay mechanisms (Lin 1992). By design, TD learning is less data-hungry (w.r.t. interactions in the environment), and involving replay mechanisms (Lin 1992; Lillicrap et al. 2016; Wang et al. 2016) significantly adds on to its inherent sample-efficiency. Based on this line of reasoning, SAM (Blondé and Kalousis 2019) and DAC (Kostrikov et al. 2019) addressed the deterring sample-complexity of GAIL by, among other improvements [cf. (Blondé and Kalousis 2019; Kostrikov et al. 2019)], using an actor-critic architecture to replace TRPO for policy evaluation and improvement. SAM (Blondé and Kalousis 2019) uses DDPG (Lillicrap et al. 2016), whereas DAC (Kostrikov et al. 2019) uses TD3 (Fujimoto et al. 2018). Both were released concurrently, and both report significant improvements in sample-efficiency (up to two orders of magnitude). Standing as the stripped-down model that brought sample-efficiency to GAIL, we take SAM as base. Albeit described momentarily in the body of this work, we urge the reader eager to understand every single aspect of the laid out algorithm to also refer to the section in which we describe the experimental setting, cf. Sect. 5.5.

We now lay out the constituents of SAM (Blondé and Kalousis 2019), and how their learning procedures are orchestrated. The agent's behavior is dictated by a *deterministic* policy μ_θ, the critic Q_ω assigns Q-values to actions picked by the agent, and the reward r_φ assesses to what degree the agent behaves like the expert. As usual, θ, ω, and φ denote the respective parameters of these neural function approximators. To explore when carrying out rollouts in the environment, μ_θ is perturbed both in parameter space by adaptive noise injection in θ (Plappert et al. 2018; Fortunato et al. 2017), and action space by adding the temporally-correlated response of an Ornstein–Uhlenbeck noise process (Uhlenbeck and Ornstein 1930; Lillicrap et al. 2016) to the action returned by μ_θ. Formally, in state s_t, action a_t is sampled from $\pi_\theta(\cdot|s_t) := \mu_{\theta+\epsilon}(s_t) + \eta_t$, where $\epsilon \sim \mathcal{N}(0, \sigma_a^2)$ (σ_a adapts conservatively such that $|\mu_{\theta+\epsilon}(s_t) - \mu_\theta(s_t)|$ remains below a certain threshold), and where η_t is the response of the Ornstein-Uhlenbeck process (Uhlenbeck and Ornstein 1930) $\mathfrak{N}_{OU}$ at timestep t in the episode, such that $\eta_t := \mathfrak{N}_{OU}(t, \sigma_b)$. Note, $\mathfrak{N}_{OU}$ is reset upon episode termination. As a first minor contribution, we carried out an ablation study on exploration strategies, and report the results in Appendix 9. While the utility of temporally-correlated noise is somewhat limited to dynamical systems, both parameter noise and input noise injections have proved beneficial in generative modeling with GANs [(Zhao et al. 2017) and (Arjovsky and Bottou 2017), respectively]. As in GAIL (Ho and Ermon 2016) (described earlier in Sect. 3), the discriminator D_φ is trained via an adversarial training procedure (Goodfellow et al. 2014) against the policy π_θ. The surrogate reward r_φ used to augment MDP $\mathbb{M}$ into $\mathbb{M}^*$ is derived from D_φ to reflect the incentive that the agent needs to complete the task at hand. In the tasks we consider in this work (simulated robotics environments (Brockman et al. 2016), based on the MuJoCo (Todorov et al. 2012) physics engine, and described in Table 1) an episode terminates either (a) when the agent *fails* to

Table 1 State and action dimensions, n and m, of the studied environments from the MuJoCo (Todorov et al. 2012) simulated robotics benchmark from OpenAI Gym (Brockman et al. 2016)

Environment	State dim. n	Action dim. m	Expert return $\mu(\sigma)$
`IDP`	11	1	9339.966(1.041)
`Hopper`	11	3	4111.823(56.81)
`Walker2d`	17	6	6046.116(13.76)
`HalfCheetah`	17	6	7613.154(36.25)
`Ant`	111	8	6688.696(48.83)
`Humanoid`	376	17	9175.152(98.94)

abbrv. `IDP` for `InvertedDoublePendulum`, the continuous control counterpart of `Acrobot`. In the last column, we report both the mean μ and standard deviation σ (formatted as $\mu(\sigma)$ in the table) of the expert's returns, aggregated across the set of 10 demonstrations used in this work

complete the task according to an task-specific criterion hard-coded in the environment, or *(b)* when the agent has performed a number of steps in the environments that exceeds a predefined hard-coded *timeout*, which we left to its default value—with the exception of `HalfCheetah`, in which *(a)* does not apply. Due to *(a)*, the agent can decide to truncate its return by triggering its own failure, and decide to "cut its losses" when it is penalized too heavily for not succeeding according to the task criterion. Always-negative rewards [e.g. per-step "−1" reward to urge to agent to complete the task quickly (Kaelbling 1993)] can therefore make the agent give up and trigger termination the earliest possible, as this would maximize its return. On the other hand, always-positive rewards can make the agent content with its sub-optimal actions which would prevent it from pursuing higher rewards, as long as it remains alive. This phenomenon has been dubbed *survival bias* in (Kostrikov et al. 2019). Notably, this discussion highlights the tedious challenge that reward shaping (Ng et al. 1999) usually represents to practitioners when designing a new task. Stemming from their generator loss counterparts in the GAN literature, the *minimax (saturating)* reward variant is $r_\varphi := -\log(1 - D_\varphi)$, and the *non-saturating* reward variant is $\log(D_\varphi)$. The minimax reward is always positive, the non-saturating reward is always negative, and the sum of the two can take positive and negative values. We found empirically that using the minimax reward, despite being always positive, yielded by far the best results compared to the sum of the two variants. The performance gap is reduced in the `HalfCheetah` task which was expected since it is the only task in which the agent can not trigger an early termination. We report these comparative results in Appendix 6. Crucially, these results show that the base method considered in this work can already successfully mitigate survival bias, without requiring additional reward shaping. In summary, we use the formulation $r_\varphi := -\log(1 - D_\varphi)$, unless stated otherwise explicitly.

We also adopt the mechanism introduced in Kostrikov et al. (2019) that wraps the absorbing transitions (agent-generated *and* expert-generated) to enable the discriminator to distinguish between terminations caused by failure and terminations triggered by the

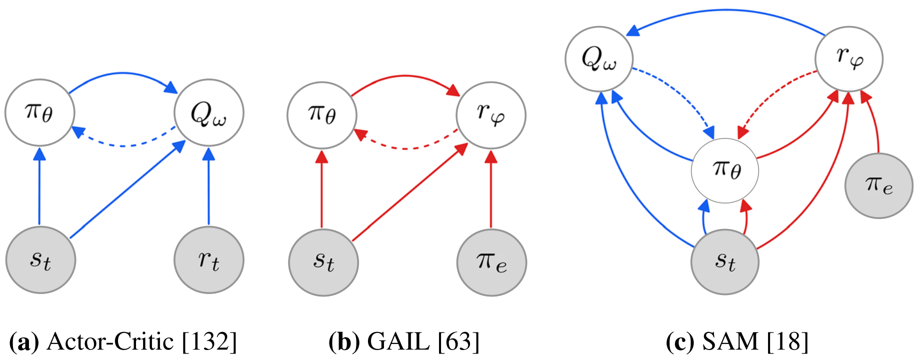


Fig. 1 Information flows (plain arrows) and gradient flows (dotted arrows) between modules. Best seen in color (Color figure online)

artificially hard-coded timeout. The method enables the discriminator to penalize the agent for terminating by failure when the expert would, with the same action and in the same state, terminate by reaching the episode timeout without failing. In such a scenario, without wrapping the absorbing transitions, the agent perfectly imitates the expert in the eyes of the discriminator, which is not the case. We use the wrapping mechanism in every experiment. Nonetheless, we omit it from the equations and algorithms for legibility. Giving the agent the ability to differentiate between terminations that are due to time limits and those caused by the environment had proved crucial for the decision maker to continue beyond the time limit. The significant role played by the explicit inclusion of the notion of time in RL has been established by Harada Harada (1997), yet without much follow-up, until being revived in Pardo et al. (2018) where the authors demonstrate that a careful inclusion of the notion of time in RL can meaningfully impact performance.

By assuming the roles of opponents in a GAN, θ and φ are tied in a *bilevel* optimization problem (as highlighted in Pfau and Vinyals (2016)). Similarly, by defining an actor-critic architecture, θ and ω are also tied in a bilevel optimization problem. We notice the dual role of θ, which is intricately tied in both bilevel problems. As such, what SAM (Blondé and Kalousis 2019) sets out to solve can be dubbed a *θ-coupled twin bilevel* optimization problem. Note, Q_ω uses the parametric reward r_φ as a scalar detached from the computational graph of the (θ, ω) bilevel problem, as having gradients flow back from Q_ω to φ would prevent D_φ from being learned as intended, i.e. adversarially in the (θ, φ) bilevel problem. The information and gradient flows occurring between the components are illustrated in Fig. 1. As we show via numerous ablation studies in this work, training this *θ-coupled twin bilevel* system to completion is severely prone to instabilities and highly sensitive to hyper-parameters. Ultimately, we show that r_φ's Lipschitzness is a *sine qua non* condition for the method to perform well, and study the effects of this necessary condition in several theoretical results in Sect. 6.1.

Sample-efficiency is achieved through the use of a replay mechanism (Lin 1992): every component (every neural network, θ, ω, and φ) is trained using samples from the replay buffer $\mathcal{R}$ (Mnih et al. 2013, 2015), a *“first in, first out”* queue of fixed retention window, to which new rollout samples (transitions) are sequentially added, and from which old rollout samples are sequentially removed. Note however that when a transition is sampled from $\mathcal{R}$, its reward component is re-computed using the most recent r_φ update. Blondé and Kalousis (2019) and Kostrikov et al. (2019) were the first to train D_φ with experience replay, in a non-*i.i.d.* context (Markovian), for increased learning stability. Borrowing the common terminology, the reward is therefore effectively *“learned off-policy”*. Let β be the off-policy distribution that corresponds to uniform sampling over $\mathcal{R}$. β is therefore effectively a mixture of past policy updates $[\theta_{i-\Delta+1}, \dots, \theta_{i-1}, \theta_i]$, where the mixing depends on $\mathcal{R}$'s retention window, and the number of collected samples per iteration.

We introduce $\rho^\pi_{\mathbb{M}^*}$, which denotes the discounted state visitation frequency of an arbitrary policy π in $\mathbb{M}^*$. Formally, $\rho^\pi_{\mathbb{M}^*}(s) := \sum_{t=0}^{+\infty} \gamma^t \mathbb{P}^\pi_{\mathbb{M}^*}[S_t = s]$, where $\mathbb{P}^\pi_{\mathbb{M}^*}[S_t = s]$ is the probability of reaching state s at timestep t when interacting with the MDP $\mathbb{M}^*$ by acting according to π. Since $\sum_{s\in\mathcal{S}} \rho^\pi_{\mathbb{M}}(s) = 1/(1-\gamma)$, $\rho^\pi_{\mathbb{M}}$ can be seen as a probability distribution over states up to a constant factor. Due to the presence of the discount factor γ, $\rho^\pi_{\mathbb{M}^*}(s)$ has higher value if s is visited earlier than later in the infinite-horizon trajectory. In practice, we relax the definition to its non-discounted counterpart and to the episodic regime case, as is usually done. Plus, since every interaction is done in MDP $\mathbb{M}^*$, we use the shorthand ρ^π. From this point forward, when states s_t are sampled uniformly from the replay buffer $\mathcal{R}$—in effect, following policy β—the expectation over said samples will be denoted as $\mathbb{E}_{s_t\sim\rho^\beta}[\cdot]$.

We now go over how each module (θ, ω, and φ) is optimized in this work. We optimize φ with the binary cross-entropy loss, where positive-labeled samples are from π_e, and negative-labeled samples are from β:

$$\ell_\varphi := \mathbb{E}_{s_t\sim\rho^{\pi_e}, a_t\sim\pi_e}[-\log(1 - D_\varphi(s_t, a_t))] + \mathbb{E}_{s_t\sim\rho^\beta, a_t\sim\beta}[-\log(D_\varphi(s_t, a_t))] \tag{1}$$

In this work, unless stated otherwise, φ is regularized with gradient penalization $\mathfrak{R}^\zeta_\varphi(k)$, subsuming the original formulation proposed in Gulrajani et al. (2017), which was used in SAM (Blondé and Kalousis 2019) and DAC (Kostrikov et al. 2019):

$$\ell^{\text{GP}}_\varphi := \ell_\varphi + \lambda\, \mathfrak{R}^\zeta_\varphi(k) := \ell_\varphi + \lambda\, \mathbb{E}_{s_t\sim\rho^\zeta, a_t\sim\zeta}[(\|\nabla_{s_t,a_t} D_\varphi(s_t, a_t)\| - k)^2] \tag{2}$$

The regularizer will be the object of several downstream analyses and discussions (cf. Sects. 5.4 and 6.3). The meaning of λ, k and ζ will be given in Sect. 5.4.

The critic's parameters ω are updated by gradient decent on the TD loss (Sutton 1988), using the multi-step version (Peng et al. 1996) (“*n-step*”) of the Bellman target (R.H.S. of the expected Bellman equation), which has proven beneficial for policy evaluation (Hessel et al. 2017; Fernando Hernandez-Garcia and Sutton 2019). The loss optimized by the critic is:

$$\ell_\omega := \mathbb{E}_{s_t\sim\rho^\beta, a_t\sim\beta}[(Q_\omega(s_t, a_t) - Q^{\text{targ}})^2] \tag{3}$$

where the target Q^{targ} uses *softly*-updated (Lillicrap et al. 2016) target networks (Mnih et al. 2013, 2015), θ' and ω', and is defined as:

$$Q^{\text{targ}} := \sum_{k=0}^{n-1} \gamma^k r_\varphi(s_{t+k}, a_{t+k}) + \gamma^n Q_{\omega'}(s_{t+n}, \mu_{\theta'}(s_{t+n})) \qquad \blacktriangleright \textit{Bellman target} \tag{4}$$

$$(\theta', \omega') \leftarrow (1-\tau)(\theta', \omega') + \tau(\theta, \omega) \quad 0 \leq \tau \leq 1 \qquad \blacktriangleright \textit{target networks update} \tag{5}$$

Finally, since μ_θ is deterministic, its utility value at timestep t is $U_t(\mu_\theta) = V^{\mu_\theta}(s_t) = Q^{\mu_\theta}(s_t, \mu_\theta(s_t)) \approx \mathbb{E}_{s_t \sim \rho^\beta}[Q_\omega(s_t, \mu_\theta(s_t))] =: \mathcal{U}_\theta$, where the approximation is due to the actor-critic design involving the use of function approximators. To maximize its utility at t, θ must take a gradient step in the ascending direction, derived according to the *deterministic policy gradient theorem* (Silver et al. 2014):

$$\nabla_\theta \, U_t(\mu_\theta) \approx \nabla_\theta \, \mathcal{U}_\theta \tag{6}$$

$$= \nabla_\theta \, \mathbb{E}_{s_t \sim \rho^\beta}[Q_\omega(s_t, \mu_\theta(s_t))] \tag{7}$$

$$= \mathbb{E}_{s_t \sim \rho^\beta}[\nabla_\theta \mu_\theta(s_t) \nabla_a Q_\omega(s_t, a)|_{a=\mu_\theta(s_t)}] \tag{8}$$

This last step [Eq. (8)] emerges from the natural assumption that $\forall s \; \nabla_\theta \, s = 0$, since the analytical form of $\mathbb{M}$'s dynamics, p, is unknown. To overcome the inherent *overestimation bias* (Thrun and Schwartz 1993) hindering Q-Learning and actor-critic methods based on greedy action selection [e.g. DDPG (Lillicrap et al. 2016)], and therefore suffered by our critic Q_ω, we apply the actor-critic counterpart of double-Q learning (van Hasselt 2010)—analogously, Double-DQN (van Hasselt et al. 2015) for DQN—proposed in Twin-Delayed DDPG (*abbrv.* TD3) (Fujimoto et al. 2018). This add-on method, simply called *clipped double-Q learning* (*abbrv.* CD), consists in learning an additional (or *"twin"*) critic, and using the smaller of the two associated Q-values in the Bellman target, used in the temporal-difference error of both critics. For its reported benefits at minimal cost, we also use the other main add-on proposed in TD3 (Fujimoto et al. 2018) called *target policy smoothing*. The latter adds noise to the target action in order for the deterministic policy not to pick actions with erroneously high Q-values, as such input noise injection effectively smooths out the Q landscape along changes in action. Target policy smoothing (or target smoothing, *abbrv.* TS) draws strong inspiration from the SARSA (Sutton and Barto 1998) learning update since it uses a perturbation of the greedy next-action in the learning update rule, which makes the method more robust against noisy inputs and therefore potentially safer in a safety-critical scenario. Note, while value overfitting primarily impedes policies that are deterministic by design, stochastic policies that prematurely collapse to their mode (Schulman et al. 2015) are deterministic in effect and as such are impeded too. In particular, fitting the value estimate against an expectation of *similar* bootstrapped target value estimates forces similar actions to have similar values, which corresponds—by definition—to making the Q-function locally Lipschitz-continuous. As such, the induced smoothness over Q is to be understood in terms of *local Lipschitz-continuity* (or equivalently, *local Lipschitzness*), which we define in Definition 1. More generally, the concept of smoothness that is at the core of the analyses laid out in this work is the concept of Lipschitz-continuity. Interestingly, we show later in Sect. 6.2.4, formally and from first principles, that target policy smoothing is equivalent to applying a regularizer on Q that induces Lipschitz-continuity *w.r.t.* the action input. In addition, we align the notion of *robustness* of a function approximator with the value of its *Lipschitz constant* (cf. Definition 1): a k_1-Lipschitz-continuous function approximator will be characterized as *more robust* than another k_2-Lipschitz-continuous function approximator if and only if $k_1 \leq k_2$. As such, in this work, the notions of smoothness and robustness are both aligned with the notion of Lipschitz-continuity.

Definition 1 (*Local k-Lipschitz-continuity*) Let f be a function $\mathcal{X} \subseteq \mathbb{R}^n \rightarrow \mathcal{Y} \subseteq \mathbb{R}^m$, $x \mapsto f(x)$, and C^0 (continuous) over $\mathcal{X}$. We denote the euclidean norms of $\mathcal{X}$ and $\mathcal{Y}$ by $\| \cdot \|_\mathcal{X}$ and $\| \cdot \|_\mathcal{Y}$ respectively, and the Frobenius norm of the $\mathbb{R}^{m \times n}$ matrix space by $\| \cdot \|_F$. Lastly, let k be a non-negative real, $k \geq 0$.

(a) f is k-Lipschitz-continuous over $\mathcal{X}$ iff, $\forall x, x' \in \mathcal{X}$,

$$\|f(x) - f(x')\|_\mathcal{Y} \leq k \, \|x - x'\|_\mathcal{X}$$

(b) If f is also differentiable, then f is k-Lipschitz-continuous over $\mathcal{X}$ iff, $\forall x, x' \in \mathcal{X}$,

$$\|\nabla f(x)\|_F \leq k$$

In either case, if the inequality is verified, k is called the *Lipschitz constant* of f. The symbol ∇, historically reserved to denote the gradient operator, is here used to denote the Jacobian operator of the vector function f, to maintain symmetry with the notations and appellations used in previous works.

(c) Let X be a subspace of $\mathcal{X}$, $X \subseteq \mathcal{X}$. f is said *locally* k-Lipschitz-continuous over $X \subseteq \mathcal{X}$ iff, for all $x \in X$, there exists a neighborhood U_x of x such that f is k-Lipschitz-continuous over U_x.

Based on Definition 1(b) the gradient penalty in Eq. (2), effectively enforces local Lipschitz-continuity over the support of the ζ distribution (described later in cf. Sect. 5.4), a subspace of the state-action joint space.

Unless specified otherwise, we use both the clipped double-Q learning and target policy smoothing add-on techniques in all the experiments reported in this work. We ran an ablation study on both techniques to illustrate their respective benefits, and support our algorithmic design choice to use them. We report said ablations in Appendix 4.

We describe the inner workings of SAM in Algorithm 1.[1]

Since our agent learns a parametric reward—differentiable by design—along with a deterministic policy, we *could*, in principle, use the gradient $\mathbb{E}_{s_t \sim \rho^\beta}[\nabla_\theta \mu_\theta(s_t) \nabla_a r_\varphi(s_t, a)|_{a=\mu_\theta(s_t)}]$ [constructed by analogy with Eq. (8)] to update the policy. (Blondé and Kalousis 2019)

[1] The symbols "◇" and "◇" appearing in front of line numbers in Algorithm 1 are related to the distributed learning scheme used in this work, which we describe in Sect. 5.5.

raised the question of whether one *should* use this gradient and answered in the negative: while the gradient in Eq. (8) guides the policy towards behaviors that maximize the long-term return of the agent, effectively trying to address the credit assignment problem, the gradient involving r_φ in place of Q_ω is myopic, and does not encourage the policy to think more than one step ahead. It is obvious that back-propagating through Q_ω, literally designed to enable the policy to reason across longer time ranges, will be more helpful to the policy towards solving the task. The authors therefore discard the gradient involving r_φ. Nonetheless, we set out to investigate whether the latter can favorably assist the gradient in Eq. (8) in solving the task, when both gradients are used *in conjunction*. Drawing a parallel with the line of work using unsupervised auxiliary tasks to improve representation learning in visual tasks (Jaderberg et al. 2016; Shelhamer et al. 2016; Mirowski et al. 2016; Doersch et al. 2015), we define the gradient $\mathbb{E}_{s_t \sim \rho^\beta}[\nabla_\theta \mu_\theta(s_t) \nabla_a Q_\omega(s_t, a)|_{a=\mu_\theta(s_t)}]$ as the *main* gradient, and $\mathbb{E}_{s_t \sim \rho^\beta}[\nabla_\theta \mu_\theta(s_t) \nabla_a r_\varphi(s_t, a)|_{a=\mu_\theta(s_t)}]$ as the *auxiliary* gradient, which we denote by g_m and g_a respectively. Based on our previous argumentation, allowing the myopic g_a to take the upper hand over g_m could have a disastrous impact on solving the task: combining the g_m and g_a must be done conservatively. As such, we use the auxiliary gradient only if it amplifies the main gradient. We measure the complementarity of the main and auxiliary tasks by the cosine similarity between their respective gradients, $\mathfrak{S}(g_m, g_a))$, as done in Du et al. (2018), and assemble the new composite gradient $g_c := g_m + \max(0, \mathfrak{S}(g_m, g_a))\, g_a$. By design, g_a is added to g_m only if the cosine similarity between them, $\mathfrak{S}(g_m, g_a))$, is positive, and will, in that case, be scaled by said cosine similarity. If the gradients are collinear, they are summed: $g_c = g_m + g_a$. If they are orthogonal or if the similarity is negative, g_a is discarded: $g_c = g_m$. Our experiments comparing the usage of g_c and g_m (cf. Fig. 12 in Appendix 3) show that using the composite gradient g_c does not yield any improvement over using only g_m. By monitoring the values taken by $\mathfrak{S}(g_m, g_a))$, we noticed that the cosine similarity was almost always negative, yet close to 0, hence $g_c = g_m$, which trivially explains why the results are almost identical.

5 Lipschitzness is all you need

This section aims to put the emphasis on what makes off-policy generative adversarial imitation learning challenging. When applicable, we propose solutions to these challenges, supported by intuitive and empirical evidence. *In fine*, as the section name hints, we found that—in our experimental and computational setting, described at the beginning of Sect. 5.5—forcing the local Lipschitzness of the reward is a *sine qua non* condition for good performance, while also being *sufficient* to achieve peak performance.

Algorithm 1: SAM: Sample-efficient Adversarial Mimic

```
init: initialize the random seeds of each framework used for sampling, the random seed of the
      environment $\mathbb{M}$, the neural function approximators' parameters ($\theta$, $\varphi$, $\omega$), their target
      networks as exact frozen copies, the rollout cache $\mathscr{C}$, the replay buffer $\mathscr{R}$.
while no stopping criterion is met do
    /* Interact with the world to collect new samples */
    repeat
        Perform action $a_t \sim \pi_\theta(\cdot|s_t)$ in state $s_t$ and receive the next state $s_{t+1}$ and termination
          indicator $d$ returned by the environment $\mathbb{M}^* - \{r_\varphi\}$;
        Store the reward-less transition $(s_t, a_t, s_{t+1})$ in the rollout cache $\mathscr{C}$;
    until the rollout cache $\mathscr{C}$ is full;
    Dump the content of the rollout cache $\mathscr{C}$ into the replay buffer $\mathscr{R}$, then flush $\mathscr{C}$;
    /* Train every modules */
    foreach training step per iteration do
        foreach reward training step per iteration do
            Get a mini-batch of samples from the replay buffer $\mathscr{R}$;
            Get a mini-batch of samples from the expert demonstration dataset $\mathscr{D}$;
            Perform a gradient descent step along $\nabla_\varphi \ell_\varphi^{\text{GP}}$ (cf. EQ 1) using both mini-batches:

              $\ell_\varphi^{\text{GP}} := \mathbb{E}_{s_t \sim \rho^{\pi_e}, a_t \sim \pi_e}[-\log(1 - D_\varphi(s_t, a_t))] + \mathbb{E}_{s_t \sim \rho^\beta, a_t \sim \beta}[-\log(D_\varphi(s_t, a_t))] + \lambda \mathfrak{R}_\varphi^\zeta(k)$

              where $\mathfrak{R}_\varphi^\zeta(k) := \mathbb{E}_{s_t \sim \rho^\zeta, a_t \sim \zeta}[(\|\nabla_{s_t, a_t} D_\varphi(s_t, a_t)\| - k)^2]$ is a gradient penalty
              regularizer;
        end
        foreach agent training step per iteration do
            Get a mini-batch of samples from the replay buffer $\mathscr{R}$;
            Augment every reward-less transition sampled from $\mathscr{R}$ with the learned reward
              surrogate $r_\varphi$: $(s_t, a_t, s_{t+1}) \rightarrow (s_t, a_t, r_\varphi(s_t, a_t), s_{t+1})$ (omitting here the use of
              $n$-step returns for simplicity);
            Perform a gradient descent step along $\nabla_\omega \ell_\omega$ (cf. EQ 3) using the mini-batch:

                          $\ell_\omega := \mathbb{E}_{s_t \sim \rho^\beta, a_t \sim \beta}[(Q_\omega(s_t, a_t) - Q^{\text{targ}})^2]$

              where $Q^{\text{targ}} := \sum_{k=0}^{n-1} \gamma^k r_\varphi(s_{t+k}, a_{t+k}) + \gamma^n Q_{\omega'}(s_{t+n}, \mu_{\theta'}(s_{t+n}))$ is the $n$-step
              Bellman target;
            Perform a gradient ascent step along $\nabla_\theta \mathscr{U}_\theta$ (cf. EQ 6) using the mini-batch:

                          $\mathscr{U}_\theta := \mathbb{E}_{s_t \sim \rho^\beta}[Q_\omega(s_t, \mu_\theta(s_t))]$

              ;
            Update the target networks using the new $\omega$ and $\theta$;
        end
    end
    Adapt parameter noise standard deviation $\sigma$ used to define $\pi_\theta$ from $\mu_\theta$ (cf. SECTION 4);
    /* Evaluate the trained policy */
    foreach evaluation step per iteration do
        Evaluate the empirical return of $\mu_\theta$ in $\mathbb{M}$, using the task reward $r$ (cf. SECTION 4);
    end
end
```

5.1 A deadlier triad

In recent years, several works (Fujimoto et al. 2018; Fu et al. 2019; Achiam et al. 2019) have carried out in-depth diagnoses of the inherent problems of Q-learning (Watkins 1989; Watkins and Dayan 1992)—and bootstrapping-based actor-critic architectures by extension—in the function approximation regime. Note, while the following issues directly apply to DQN (Mnih et al. 2013, 2015), which even introduces additional difficulties (e.g. target networks, replay buffer), we limit the scope of this section to Q-learning, to eventually make our point. Q-learning under function approximation possesses properties that, when used in conjunction, make the algorithm brittle, prone to unstable behavior, as well as tedious to bring to convergence. Without caution, the algorithm is bound to diverge. These properties constitute the *deadly triad* (Sutton and Barto 1998; van Hasselt et al. 2018): function approximation, bootstrapping, and off-policy learning.

Since the method we consider in this work *per se* follows an actor-critic architecture, it possesses all three properties, and is therefore inclined to diverge and suffer from instabilities. Additionally, since the learned reward r_φ is: (a) defined from binary classifier predictions—discriminator's predicted probabilities of being expert-generated—estimated via function approximation, (b) learned at the same time as the policy, and (c) learned off-policy—with the negative samples coming from the replay distribution β, the method we study consequently introduces an extra layer of complication in the deadly triad. We now go over the three points and explain to what extent they each exacerbate the divergence-inducing properties that form the deadly triad.

To tackle point (a), we introduce explicit residuals to represent the various sources of error involved in temporal-difference learning, and illustrate how these residuals accumulate over the course of an episode. We will use the shorthand $\mathbb{E}[\cdot]$ for expectations for the sake of legibility. We take inspiration from Eq. (12) in Fujimoto et al. (2018), where a bias term is introduced in the TD error due to the function approximation of the Q-value, as the Bellman equation is never exactly satisfied in this regime. Borrowing the terminology from the statistical risk minimization literature, while the original bias suffered by the TD error was due to the *estimation error* caused by bootstrapping, function approximation is responsible for an extra *approximation error* contribution. The sum of these two errors is represented with the residual δ_ω. Let us now consider $D_\varphi(s, a)$, the estimated probability that a sample (s, a) is coming from expert demonstrations. Formally, $D_\varphi(s, a) = \mathbb{P}_\varphi[\textsc{Expert}(s, a)]$, where the event is defined as $\textsc{Expert}(s, a) := \text{“}s \sim \rho^{\pi_e} \wedge a \sim \pi_e\text{”}$, and where $\mathbb{P}_\varphi$ denotes the probability estimated with the approximator φ. In the same vein, we distinguish the error contributions: the approximation error is caused by the choice of function approximatior class (e.g. two-layer neural networks with hyperbolic tangent activations), and the estimation error is due to the gap between the estimations of our classifier and the predictions of the *Bayes classifier*—the classifier with the lowest misclassification rate in the chosen class. This gap can be written as $|D_\varphi(s_t, a_t) - \textsc{Bayes}(s_t, a_t)|$, where $\textsc{Bayes}(s, a) = \mathbb{P}_{\textsc{Bayes}}[\textsc{Expert}(s, a)]$, by analogy with the previous notations. *In fine*, we introduce the residual δ_φ that represents the contribution of both errors in the learned reward r_φ, hence:

$$Q_\omega(s_t, a_t) = r_\varphi(s_t, a_t) - \delta_\varphi(s_t, a_t) + \gamma\mathbb{E}[Q_\omega(s_{t+1}, a_{t+1})] - \delta_\omega(s_t, a_t) \tag{9}$$

$$= [r_\varphi(s_t, a_t) - \delta_\varphi(s_t, a_t) - \delta_\omega(s_t, a_t)] + \gamma\mathbb{E}[Q_\omega(s_{t+1}, a_{t+1})] \tag{10}$$

$$= \Delta_{\varphi,\omega}(s_t, a_t) + \gamma \mathbb{E}[Q_\omega(s_{t+1}, a_{t+1})] \tag{11}$$

$$= \Delta_{\varphi,\omega}(s_t, a_t) + \gamma \mathbb{E}[\Delta_{\varphi,\omega}(s_{t+1}, a_{t+1}) + \gamma \mathbb{E}[Q_\omega(s_{t+2}, a_{t+2})]] \tag{12}$$

$$= \mathbb{E}\left[\sum_{k=0}^{+\infty} \gamma^k \, \Delta_{\varphi,\omega}(s_{t+k}, a_{t+k})\right] \tag{13}$$

where $\Delta_{\varphi,\omega}(s_t, a_t) := r_\varphi(s_t, a_t) - \delta_\varphi(s_t, a_t) - \delta_\omega(s_t, a_t)$.

As observed in Fujimoto et al. (2018) when estimating the accumulation of error due to function approximation in the standard RL setting, the variance of the state-action value is proportional to the variance of both the return and the Bellman residual δ_ω. Crucially, in our setting involving the learned imitation reward r_φ, it is *also* proportional to the variance of the residual δ_φ, containing contributions of both the approximation error *and* estimation error of r_φ. As a result, the variance of the estimate also suffers from a critically stronger dependence on γ (cf. ablation study in Appendix 7). Intuitively, as we propagate rewards further (higher γ^k value), their induced residual error triggers a greater increase in the variance of the Q-value estimate. In addition to its effect on the variance, the additional residual also clearly impacts the overestimation bias (Thrun and Schwartz 1993) it is afflicted by, which further advocates the use of dedicated techniques such as Double Q-learning (Fujimoto et al. 2018; van Hasselt 2010), as we do in this work (cf. Sect. 4). All in all, by introducing an extra source of approximation and estimation error, we further burden TD-learning.

Moving on to points (b)—the reward is learned at the same time as the policy—and (c)—the reward is learned off-policy using samples from the replay policy β—we see that each statement allow us to qualify the reward r_φ as a *non-stationary* process. Conceptually, by considering a additive decomposition of the reward r_φ into a stationary r_φ^{STAT} and a non-stationary contribution $r_\varphi^{\text{NON-STAT}}$, we see that following an accumulation analysis similar to the previous one shows that the variance of the state-action value is proportional to the variances of each contribution. While the variance of r_φ^{STAT} can be important and therefore can have a considerable impact on the variance of the Q-value estimate, it can usually be somewhat tamed with online normalization techniques and mitigated with techniques enabling the agent to cope with rewards of vastly different scales [e.g. Pop-art (van Hasselt et al. 2016)]. We show later that such methods do not help when the underlying reward is non-stationary (cf. Sect. 5.2 for empirical results). The variance of the non-stationary contribution $r_\varphi^{\text{NON-STAT}}$, indeed is, due to its continually-changing nature, untameable with these regular techniques relying on the usual stationarity assumption—unless additional dedicated mechanisms are integrated (e.g. change point detection techniques). Naturally, the non-stationary contribution also has an effect on the bias of the estimation, and *a fortiori* on its overestimation bias [as with (a)]. We note that the argument made in the context of Q-learning by Fu et al. (2019) naturally transfers to the TD-learning objective optimized in this work: the objective is non-stationary, due to (i) the *moving target* problem—caused by using bootstrapping to learn an estimate that is updated every iteration and (ii) the *distribution shift* problem—caused by learning the Q-value estimate off-policy using β, effectively being a mixture of past policies, which changes every iteration. Point (i) is a source of non-stationarity since the target of the supervised objective is moving with the prediction as iterations go by, due to using bootstrapping. Fitting the current estimate against the target defined from this very estimate is an ordeal, and (b) makes the task even harder by having the reward move too, given it is also learned, at the same time. The target of the TD objective therefore now has two moving pieces, one from bootstrapping (i), one from reward learning (b). The distribution shift problem (ii), stemming from the Q-value being learned off-policy, is naturally worsened by the reward being estimated off-policy (c). Note, although both the reward and Q-value are learned with samples from β, the actual mini-batches used to perform the gradient update of each estimate might be different in practice. As such, the TD error would be optimized using samples from a mixture of past policies that is different from the mixture under which the reward is learned, and then use this reward trained under a different effective distribution in the Bellman target. All in all, by introducing a extra sources of non-stationarity (b) and (c), we further burden the non-stationarity of TD-learning (i) and (ii).

5.2 Continually changing rewards

In a non-stationary MDP, the non-stationarities can manifest in the dynamics (Nilim and El Ghaoui 2005; Da Silva et al. 2006; Xu and Mannor 2007; Lim et al. 2013; Abdallah and Kaisers 2016), in the reward process (Even-dar et al. 2005; Dick et al. 2014), or in both conjointly (Yu and Mannor 2009a, b; Abbasi-Yadkori et al. 2013; Gajane et al. 2018; Padakandla et al. 2019; Yu and Sra 2019; Lecarpentier et al. 2019) (cf. Appendix 2 for a review of sequential decision making under uncertainty in non-stationary MDPs). In this work, we focus on the MDP $\mathbb{M}^*$ whose transition distribution p is stationary i.e. not changing over time. As discussed in Sect. 5.1, the reward process defined by r_φ is however non-stationary. In particular, r_φ is *drifting*, i.e. gradually changes at an unknown rate, due to the reward being learned at the same time as the policy, but also due to it being estimated off-policy. While the former reason is true in the on-policy setting as well, the latter is specific to the off-policy setting, on which we focus in this work. Indeed, in *on-policy* generative adversarial imitation learning, the parameter sets φ and θ are involved in a bilevel optimization problem (cf. Sect. 3) and consequently are intricately tied. φ is trained via an adversarial procedure opposing it to θ in a zero-sum two-player game. At the same time, θ is trained by policy gradients to optimize π_θ's episodic accumulation of rewards generated by r_φ. The synthetically generated rewards perceived by the agent are, in effect, sampled from a stochastic process that incrementally changes over the course of the policy updates, effectively qualifying r_φ as a drifting non-stationary reward process.

By moving to the off-policy setting—for reasons laid out earlier in Sect. 4—the zero-sum two-player game is not opposing r_φ and π_θ, but r_φ and β, where β is the off-policy distribution stemming from experience replay. As the parameter set θ go through gradient updates, the new policies π_θ are added to the mixture of past policies β. Crucially, to perform its parameter update at a given iteration, the policy π_θ uses transitions augmented with rewards generated by r_φ, whose latest update was trying to distinguish between samples from π_e and β (as opposed to π_e and π_θ in the on-policy setting). Since π_θ is drifting, β is also drifting based on how experience replay operates. Nevertheless, by being a mixture of previous policy updates, β potentially drifts less that π_θ, since, in effect, two consecutive β distributions are mixing over a wide overlap of the same past policies. In reality however, β corresponds to uniformly sampling a mini-batch from the replay buffer. Consecutive β can therefore be uncontrollably distant from each other in practice, making the distributional drift of the reward more tedious to deal with than in the on-policy setting. Using large mini-batches and distributed multi-core architectures somewhat levels the playing field though.

The adversarial bilevel optimization problem guiding the adaptive tuning of r_φ for every π_θ update is reminiscent of the stream of research pioneered by Auer et al. (1995) in which the reward is generated by an omniscient *adversary*, either arbitrarily or adaptively with potentially malevolent drive (Yu and Mannor 2009a, b; Lim et al. 2013; Gajane et al. 2018; Yu and Sra 2019). Non-stationary environments are almost exclusively tackled from a theoretical perspective in the literature (cf. previous references). Specifically, in the *drifting* case, the non-stationarities are traditionally dealt with via the use of sliding windows. The accompanying (dynamic) regret analyses all rely on strict assumptions. In the switching case, one needs to know the number of occurring switches beforehand, while in the drifting case, the change variation need be upper-bounded. Specifically, (Besbes et al. 2014; Cheung et al. 2019a) assume the total change to be upper-bounded by some preset variation budget, while (Cheung et al. 2019b) assumes the variations are uniformly bounded in time. Ortner et al. (2019) assumes that the *incremental* variation [as opposed to *total* in (Besbes et al. 2014; Cheung et al. 2019a)] is upper-bounded by a *per-change* threshold. Finally, in the same vein, (Lecarpentier et al. 2019) posits *regular evolution*, by making the assumption that both the transition and reward functions are Lipschitz-continuous *w.r.t.* time. By contrast, our approach relies on imposing local Lipschitz-continuity of the reward over the input space, which will be described later in Sect. 5.4.

Online return normalization methods—using statistics computed over the entire return history (reminiscent of sliding window methods) to whiten the current return estimate—are the usual go-to solution to deal with rewards (and *a fortiori* returns) whose scale can vary a lot, albeit still under stationarity assumption. We investigate whether online return normalization methods and Pop-Art (van Hasselt et al. 2016) can have a positive impact on learning performance, when the process underlying the reward is learned at the same time as the policy, via experience replay. Given that the reward distribution can drift at an unknown rate (although influenced by the learning rate used to train φ), it is fair to assume that we might benefit from such methods, especially considering how unstable a twin bilevel optimization problem can be. On the other hand, as learning progresses, older rewards are – especially in early training—*stale*, which can potentially pollute the running statistics accumulated by these normalization techniques. The results obtained in this ablation study are reported in Appendix 8.

We observe that neither return normalization nor Pop-Art provide an improvement over the baseline. On the contrary, in `Hopper` and `Walker2d`, we see that they even yield significantly poorer performance within the allowed runtime, compared to the base method using neither return normalization nor Pop-Art (cf. Fig. 20). We propose an explanation of this phenomenon based on the *stability-plasticity dilemma* (Carpenter and Grossberg 1987). In early training, the policy π_θ changes at a fast rate and with a high amplitude when going through gradient updates, due to being a randomly initialized neural function approximator. The reward r_φ is in a symmetric situation, but is also influenced by the rate of change of θ, being trained in an adversarial game. In order to keep up with this fast pace of change in early training, the critic Q_ω—using the reward r_φ in its own learning objective—needs to be sufficiently flexible to accommodate and adapt quickly to these frequent changes. In other words, the critic's *plasticity* must be high. Since reward estimates from r_φ become stale after a few φ updates, we also want our critic to avoid using stale reward to prevent the degradation of ω. This property is referred to as *stability* in Carpenter and Grossberg (1987). *In fine*, the critic must be plastic and stable. Note, using the current reward update to augment the sample transitions with their reward, as done in this work, provides the critic with such stability. However, return normalization and Pop-Art use stale running statistics estimates to whiten the state-action values returned by the critic, which prevents both plasticity (values need to change fast with the reward, normalization slows down this process) and harms stability due to the staleness of the obsolete reward that are *"baked in"* the running statistics. The obtained results corroborate the previous analysis (cf. Appendix 8).

We conclude this section by discussing the reward learning dynamics. While in the transient regime, the reward process is effectively non-stationary, it gradually becomes stationary as it reaches a steady-state regime. Nonetheless, the presence of such stabilization does not guarantee that the desired equilibrium has been reached. Indeed, as we will discuss in the next section, adversarial imitation learning has proved to be prone to overfitting. We now address it.

5.3 Overfitting cascade

Being based on a binary classifier, the synthetic reward process r_φ is inherently susceptible to overfitting, and it has been shown (cf. subsequent references) that it indeed does. As exhibited in Sect. 2, several endeavors have proposed techniques to prevent the learned reward from overfitting, individually building on traditional regularization methods aimed to address overfitting in classification. These techniques either make the discriminator model weaker (Reed et al. 2018; Blondé and Kalousis 2019; Kostrikov et al. 2019; Peng et al. 2018), or make the classification task harder (Blondé and Kalousis 2019; Xu and Denil 2019; Zolna et al. 2019), to deter the discriminator from relying on non-salient features to trivially distinguish between samples from π_e and π_θ (π_e and β in our off-policy setting, cf. Sect. 5.2).

On a more fundamental level, the ability of deep neural networks to generalize (and *a fortiori* to circumvent overfitting) had been attributed to the flatness of the loss landscape in the neighborhoods of minima of the loss function (Hochreiter and Schmidhuber 1997; Keskar et al. 2017)—provided the optimization method is a variant of stochastic gradient descent. While it has more recently been shown that sharp minima *can* generalize (Dinh et al. 2017), we argue and show both empirically and analytically that, in the off-policy setting tackled in this work, flatness of the reward function around the maxima—corresponding to the positive samples, i.e. the expert data—is paramount for good empirical performance. In other words, we argue that the presence of peaks in the reward function caused by the discriminator overfitting on the expert data (non-salient features in the worst case) is the major source of optimization issues occuring in off-policy GAIL. As such, we focus on methods that address overfitting by inducing flatness in the learned reward function around expert samples, subject to being peaked on the reward landscape. An obvious candidate to enforce this desired flatness property is gradient penalty regularization, inducing Lipschitz-continuity on the reward function r_φ, over its input space $\mathcal{S} \times \mathcal{A}$, which has been described earlier in Sect. 4, and will be the object of Sects. 5.4 and 6.3.

Simply put, reward overfitting translates to the presence of peaks on the reward landscape. Even in the case where these peaks exactly coincide with the expert data (perfect classification, the discriminator coincides with the Bayes classifier of the function class), peaked reward landscapes (i.e. sparse reward setting) can be tedious to optimize over. Crucially, peaks in r_φ *can potentially* cause peaks in the state-action value landscape Q_ω. When policy evaluation is done via Monte-Carlo estimation, the length of the rollouts likely attenuates the contribution of individual peaked rewards aggregated during the rollout into a discounted sum. If the peaks were not predominant in the rollout, the associated empirical estimate of the value will not be peaked (relative to its neighboring values). By contrast, the TD's bootstrapping-based objective does not attenuate peaks in r_φ, which consequently causes peaks in Q_ω. Note, using multi-steps returns (Peng et al. 1996) can help mitigate the phenomenon and benefit from the attenuation effect witnessed in the Monte-Carlo estimation described above, hence our usage of multi-step returns in this work (cf. Sect. 4).

Narrow peaks in the state-action value estimate Q_ω can cause the deterministic policy μ_θ to itself overfit to these peaks on the Q_ω landscape. As such overfitting *cascades* from rewards to the policy, and hampers policy optimization [cf. Eq. (8)]. Furthermore, peaks in Q-values can severely hinder temporal-difference optimization since, by design, these outlying values can appear in either the predicted Q-value or the target Q-value. As such, echoing the observations and analyses made in Sects. 5.1 and 5.2, bootstrapping makes the optimization more tedious, when bringing sampled-efficiency to GAIL. These irregularities naturally transfer to the loss landscape, exacerbating the innate irregularity of loss landscapes when using neural networks as function approximators (Li et al. 2018), making it harder to optimize over Eq. (3). *In fine*, peaks on the reward landscape can cascade and impede both policy improvement and evaluation.

In the next section (Sect. 5.4), we discuss how to enforce Lipschitz-continuity in usual neural architectures, before going over empirical results corroborating our previous analyses (Sect. 5.5). Ultimately, we show that *not* forcing Lipschitz-continuity on the learned surrogate reward yields poor results, making it a *sine qua non* condition for success.

5.4 Enforcing Lipschitz-continuity in deep neural networks

Designed to address the shortcomings of the original GAN (Goodfellow et al. 2014), whose training effectively minimizes a Jensen-Shannon divergence between generated and real distributions, the Wasserstein GAN (WGAN) (Arjovsky et al. 2017) leverages the Wasserstein metric. Specifically, the authors of Arjovsky et al. (2017) use the dual representation of the *Wasserstein-1* metric under a *1-Lipschitz-continuity* (cf. Definition 1) assumption over the discriminator, which allow them to employ the Kantorovich-Rubinstein duality theorem, to eventually arrive at a tractable loss one can optimize over.

In the Wasserstein GAN (Arjovsky et al. 2017), the weights of the discriminator—called *critic* to emphasize that it is no longer a classifier—are *clipped*. While not equivalent to enforcing the 1-Lipschitz constraint their model is theoretically built on, clipping the weights *does* loosely enforce Lipschitz-continuity, with a Lipschitz constant depending on the clipping boundaries. This simple technique however disrupts, by its design, the optimization dynamics. As emphasized in Gulrajani et al. (2017), clipping the weights of the Wasserstein critic can result in a pathological optimization landscape, echoing the analysis carried out in Sect. 5.3.

In an attempt to address this issue, the authors of Gulrajani et al. (2017) propose to impose the underlying 1-Lipschitz constraint via another method, fully integrated into the bilevel optimization problem as a gradient penalty regularization. When augmented with this gradient penalization technique, WGAN—dubbed WGAN-*GP*—is shown to yield consistently better results, enjoys more stable learning dynamics, and displays a smoother loss landscape (Gulrajani et al. 2017). Interestingly, the regularization technique has proved to yield better results even in the original GAN (Lucic et al. 2017), despite it not being grounded on the Lipschitzness footing like WGAN (Arjovsky et al. 2017). In addition, following in the footsteps of the comprehensive study proposed in Lucic et al. (2017) and Kurach et al. (2018) shows empirically that the WGAN loss does not outperform the original GAN consistently across various hyper-parameter settings, and advocates for the use of the original GAN loss, along with the use of spectral normalization (Miyato et al. 2018), and gradient penalty regularization (Gulrajani et al. 2017) to achieve the best results (albeit at an increased cost in computation in visual domains). In line with these works (Lucic et al. 2017; Kurach et al. 2018), we therefore commit to the archetype GAN loss formulation (Goodfellow et al. 2014), as has been laid out earlier in Sect. 4 when describing the discriminator objective in Eq. (1). We now remind the objective optimized by the discriminator [cf. Eq. (2)], where the generalized form of the gradient penalty, $\mathfrak{R}^{\zeta}_{\varphi}(k)$, subsumes the original penalty (Gulrajani et al. 2017) as well as variants that will be studied later in Sect. 6.3:

$$\ell^{\text{GP}}_{\varphi} := \ell_{\varphi} + \lambda\, \mathfrak{R}^{\zeta}_{\varphi}(k) := \ell_{\varphi} + \lambda\, \mathbb{E}_{s_t \sim \rho^{\zeta}, a_t \sim \zeta}[(\|\nabla_{s_t, a_t} D_{\varphi}(s_t, a_t)\| - k)^2] \tag{14}$$

In Eq. (14), λ corresponds to the weight attributed to the regularizer in the objective (cf. ablation in Sect. 6.3), and $\|\cdot\|$ depicts the euclidean norm in the appropriate vector space. ζ is the distribution defining *where* in the input space $\mathcal{S} \times \mathcal{A}$ the Lipschitzness constraint should be enforced. ζ is defined from π_e and β. In the original gradient penalty formulation (Gulrajani et al. 2017), ζ corresponds to sampling points uniformly in segments[2] joining points from the generated data and real data, grounded on the derived theoretical results (cf. Proposition 1 in Gulrajani et al. (2017)) that the optimal discriminator is 1-Lipschitz along these segments. While it does not mean that enforcing such constraint will make the discriminator optimal, it yields good results in practice. We discuss several formulations of ζ in Sect. 6.3, evaluate them empirically and propose intuitive arguments explaining the obtained results. In particular, we adopt an *RL viewpoint* and propose an alternate ground as to why the regularizer has enabled successes in control and search tasks, as reported in Blondé and Kalousis (2019); Kostrikov et al. (2019). In particular, in Gulrajani et al. (2017), the 1-Lipschitz-continuity is encouraged by using $\mathfrak{R}^{\zeta}_{\varphi}(1)$ as regularizer.

Additionally, in line with the observations done in Gulrajani et al. (2017), we investigated with (a) replacing $\mathfrak{R}^{\zeta}_{\varphi}(k)$ with a *one-sided* alternative defined as $\mathbb{E}_{s_t \sim \rho^{\zeta}, a_t \sim \zeta}[\max(0, \|\nabla_{s_t, a_t} D_{\varphi}(s_t, a_t)\| - k)^2]$, and (b) ablating online batch normalization of the state input from the discriminator. The alternative regularizer of (a) encourages the norm to be *lower* than k (formally, $\|\nabla_{s_t, a_t} D_{\varphi}(s_t, a_t)\| \leq k$) in contrast to the original regularizer that enforces it to be *close* to k. While the one-sided version describes the notion of k-Lipschitzness more accurately (cf. Definition 1), it yields similar results overall, as shown in Appendix 5.1. Crucially, we conclude from these experiments that it is *sufficient* to have the norm remain upper-bounded by k, or equivalently, to have D_{φ} be Lipschitz-continuous. In other words, we do not need to impose a stronger constraint than k-Lipschitz-continuity on the discriminator to achieve peak performance, in the context of this ablation study. As for (b), online batch normalization of the state input is mostly hurting performance. as reported in Appendix 5.2. We therefore arrive at the same conclusions as Gulrajani et al. (2017): (a) we use the *two-sided* formulation of $\mathfrak{R}^{\zeta}_{\varphi}(k)$ described in Eq. (14) since using the once-sided variant yields no improvement, and (b) we omit the online batch normalization of the state input in the discriminator since it hurts performance, while still using this normalization scheme in the policy and critic (more details about the technique will be given when we describe our experimental setting in the next section, Sect. 5.5).

[2] The segment joining the arbitrary points x and y in $\mathbb{R}^d$ is the set of points defined as $S := \{(1-\alpha)x + \alpha y \mid \alpha \in [0,1]\}$. Sampling a point $z \in \mathbb{R}^d$ uniformly from S corresponds to sampling $\alpha \sim \text{unif}(0,1)$, before assembling $z := (1-\alpha)x + \alpha y$.

5.5 Diagnosing the importance of Lipschitzness empirically in off-policy adversarial imitation learning

Before going over the empirical results reported in this section, we describe our experimental setting. Unless explicitly stated otherwise, every experiment—reported in both this section and Sect. 6.5—is run in the same base setting. In addition, the used hyperparameters are made available in Appendix 1.

5.5.1 Environments

In this work, we consider the simulated robotics, continuous control environments built with the MuJoCo (Todorov et al. 2012) physics engine, and provided to the community through the OpenAI Gym API (Brockman et al. 2016). We use the following versions of the environments: `v3` for `Hopper`, `Walker2d`, `HalfCheetah`, `Ant`, `Humanoid`, and `v2` for `InvertedDoublePendulum`. For each of these, the dimension n of a given state $s \in \mathcal{S} \subseteq \mathbb{R}^n$ and the dimension m of a given action $a \in \mathcal{A} \subseteq \mathbb{R}^m$ scale as the degrees of freedom (DoFs) associated with the environment's underlying MuJoCo model. As a rule of thumb, the more complex the articulated physics-bound model is (i.e. more limbs, joints with greater DoFs), the larger both n and m are. The intrinsic difficulty of the simulated robotics task scales super-linearly with n and m, albeit considerably faster with m (policy's output) than with n (policy's input).

Omitting their respective versions, Table 1 reports the state and action dimensions (n and m respectively) for all the environments tackled in this work, and are ordered, from left to right, by increasing state and action dimensions, `Humanoid-v3` being the most challenging. Since we consider, in our experiments, expert datasets composed of at most 10 demonstrations (10 is the default number; when we use 5, we specify it in the caption), we report return statistics (mean μ and standard deviation σ, formatted as $\mu(\sigma)$ in Table 1) aggregated over the set of 10 deterministically-selected demonstrations (the 10 first in our fixed pool) that every method requesting for 10 demonstrations will receive. To reiterate: in this work, every single method and variant will receive exactly the same demonstrations, due to an explicit seeding mechanism in every experiment. The reported statistics therefore identically apply to every method or variant using 10 demonstrations. By design, this reproducibility asset naturally extends to settings requesting fewer.

5.5.2 Demonstrations

As in Ho and Ermon (2016), we subsampled every demonstration with a $1/u$ ratio—an operation called *temporal dropout* in Duan et al. (2017). For a given demonstration, we sample an index i_0 from the discrete uniform distribution $\text{unif}\{0, u-1\}$ to determine the first subsampled transition. We then take one transition every u transition from the initial index i_0. *In fine*, the subsampled demonstration is extracted from the original one of length l by only preserving the transitions of indices $\{i_0 + ku \,|\, 0 \leq k < \lfloor l/u \rfloor\}$. Since the experts achieve very high performance in the MuJoCo benchmark (cf. last column of Table 1) they never fail their task and live until the *"timeout"* episode termination triggered by OpenAI Gym API, triggered once the horizon of 1000 timesteps is reached, in every environments considered in this work. As such, most demonstrations have a length

$l \approx 1000$ transitions (sometimes less but always above 950). Since we use the sub-sampling rate $u = 20$, as in Ho and Ermon (2016), the subsampled demonstrations have a length of $|\{i_0 + ku \mid 0 \leq k < \lfloor l/u \rfloor\}| = \lfloor l/u \rfloor \approx 50$ transitions.

We wrap the absorbing states in both the expert trajectories beforehand and agent-generated trajectories at training time, as introduced in Kostrikov et al. (2019). Note, this assumes knowledge about the nature—organic (e.g. falling down) and triggered (e.g. time-out flag set at a fixed episode horizon)—of the episode terminations (if any) occurring in the expert trajectories. Considering the benchmark, it is trivial to individually determine their natures in our work, which makes said assumption of knowledge weak. We trained the experts from which the demonstrations were then extracted using the on-policy state-of-the-art PPO (Schulman et al. 2017) algorithm. We used early stopping to halt the expert training processes when a phenomenon of diminishing returns is observed in its empirical return, typically attained by the 20 million interactions mark. We used our own parallel PPO implementation, written in PyTorch (Paszke et al. 2019), and will share the code upon acceptance. The IL endeavors presented in this work have also been implemented with this framework.

5.5.3 Distributed training

The distributed training scheme employed to obtain every empirical imitation learning result exhibited in this work uses the MPI message-passing standard. Upon launch, an experiment spins n workers, each assigned with an identifying unique rank $0 \leq r < n$. They all have symmetric roles, except the rank 0 worker, which will be referred to as the *"zero-rank"* worker. The role of each worker is to follow the studied algorithm—SAM (cf. Algorithm 1) in the experiments reported in this section, and the proposed extension PURPLE in the experiments reported later in Sect. 6.5. The zero-rank worker exactly follows the algorithm, while the $n-1$ other workers omit the evaluation phase (denoted by the symbol "◇" appearing in front of the line number). The random seed of each worker is defined deterministically from its rank and the *base* random seed given as a hyper-parameter by the practitioner, and is used to (a) determine the behavior of every stochastic entity involved in the worker's training process, and (b) determine the stochasticity of the environment it interacts with.

Before every gradient-based parameter update step—denoted in Algorithm 1 by the symbol "◇" appearing in front of the line number—the zero-rank worker gathers the gradients across the $n-1$ other workers, and aggregates them via an averaging operation, and sends the aggregate to every worker. Upon receipt, every worker of the pool then uses the aggregated gradient in its own learning update. Since the parameters are synced across workers before the learning process kicks off, this *synchronous* gradient-averaging scheme ensures that the workers all have the same parameters throughout the entire learning process (same initial parameters, then same updates). This distributed training scheme leverages learners seeded differently in their own environments, also seeded differently, to accelerate exploration, and above all provide the model with greater robustness.

Every imitation learning experiment whose results are reported in this work has been run for a fixed wall-clock duration—12 or 48 h, as indicated in their respective captions—due to hardware and computational infrastructure constraints. While the effective running time appears in the caption of every plot, the latter still depict the temporal progression of the methods in terms of *timesteps*, the number of interactions carried out with the environment. The reported performance corresponds to the undiscounted empirical return, computed using the reward returned by the environment (available at evaluation time), gathered by the non-perturbed policy μ_θ (deterministic) of the zero-rank worker. Every experiment uses 16 workers, and can therefore be executed on most desktop consumer-grade computers. Lastly, we monitored every experiment with the Weights & Biases (Biewald 2020) tracking and visualization tool.

Additionally, we run each experiment with 5 different *base* random seeds (0–4), raising the effective seed count per experiment to 80. Each presented plot depicts the mean across them with a solid line, and the standard deviation envelope (half a standard deviation on either side of the mean) with a shaded area.

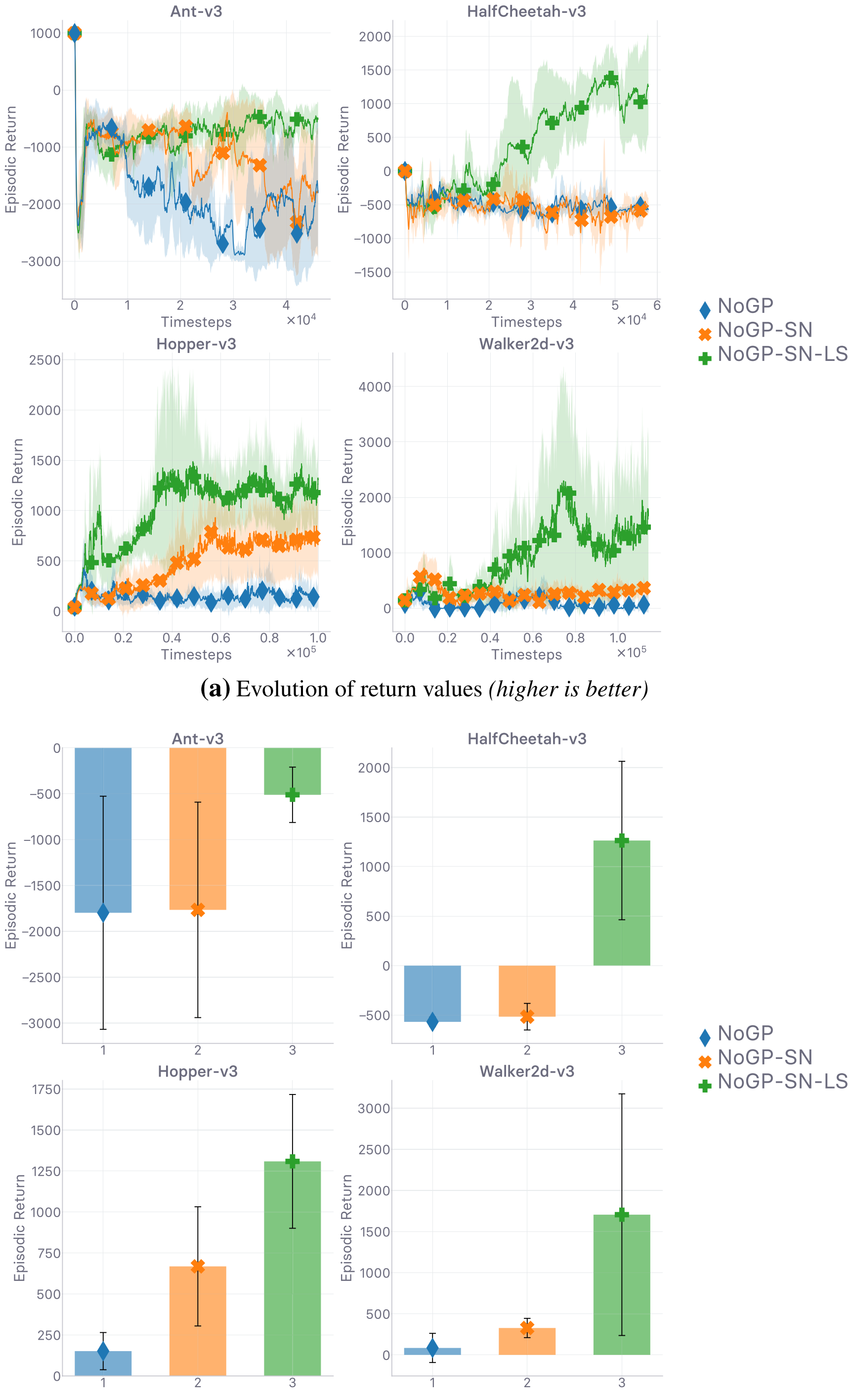
Finally, we use an *online* observation normalization scheme, instrumental in performing well in continuous control tasks. The running mean and standard deviation used to standardize the observations are computed using an online method to represent the statistics of the entire history of observation. These statistics are updated with the mean and standard deviation computed over the concatenation of latest rollouts collected by each parallel worker, making is effectively an *online distributed* batch normalization (Ioffe and Szegedy 2015) variant.

5.5.4 Empirical results

We now go over our first set of empirical results, whose goal is to show to what extent gradient penalty regularization is needed. The compared methods all use SAM (cf. Sect. 4) as base.

First, Fig. 2 compares several modular configurations, which are described using the following handles in the legend. `GP` means that gradient penalization (GP) (cf. Sect. 5.4) is used. `NoGP` means that GP is not used (using ℓ_φ instead of ℓ_φ^{GP}). Note, `NoGP` is the only negative handle that we use, since it it central to our analyses. When any other technique is not in use, it is simply absent from the handle in the legend. `SN` means that spectral normalization (SN) (Miyato et al. 2018) is used. SN normalizes the discriminator's weights to have a norm close to 1, drawing a direct parallel with GP. In line with what the large-scale ablation studies on GAN add-ons advocate (Lucic et al. 2017; Kurach et al. 2018), SN is used in most modern GAN architectures for its simplicity. We here investigate if SN is enough to keep the gradient in check, or if GP is necessary. `LS` denotes one-sided uniform label smoothing, consisting in replacing the positive labels only (hence *one-sided*), which are normally equal to 1 (expert, real), by a *soft label* u, distributed as $u \sim \text{unif}(0.7, 1.2)$. We do not consider Variational Discriminator Bottleneck (VDB) (Peng et al. 2018) in our comparisons since (a) we prefer to focus on stripped-down canonical methods, and (b) the information bottleneck forced on the discriminator's hidden representation boils down to smoothing the labels anyway, as shown recently in Müller et al. (2019).

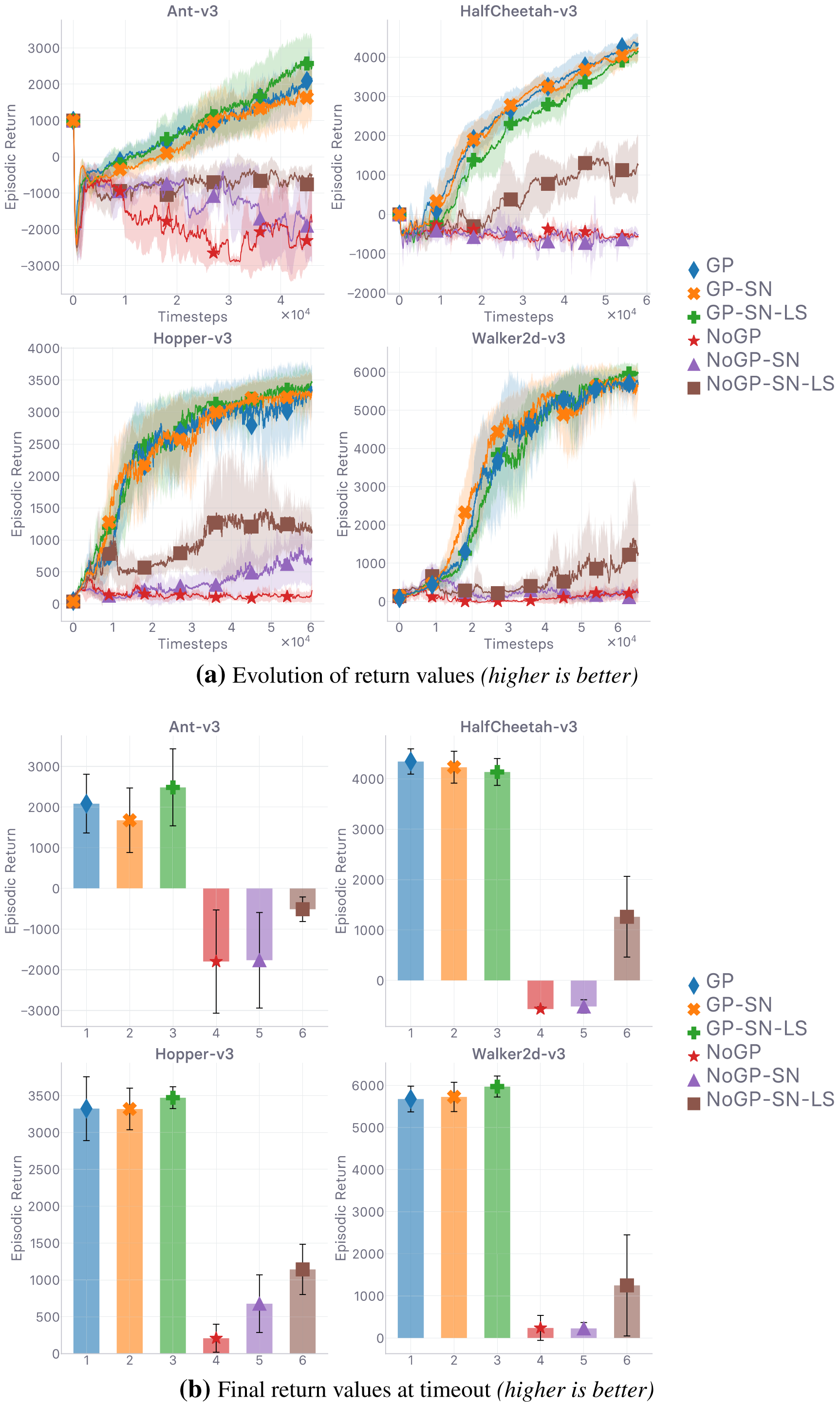
In Fig. 2, we see that *not using GP* (`NoGP`) prevents the agent from learning anything valuable: the agent barely collects *any reward at all*. While using SN can improve performance slightly (`NoGP-SN`), the addition of LS (`NoGP-SN-LS`) *considerably* improves performance over the two previous candidates. Nonetheless, despite the sizable runtime, all three perform poorly and are a far cry from achieving the same empirical return as the expert (cf. Table 1). In contrast with Figs. 2, 3 and 4 show to what extent introducing GP in the off-policy imitation learning algorithm considered in this work impacts performance positively. The performance gap is *substantial*—in every environment except the easiest one considered, `InvertedDoublePendulum-v2`, as described in Table 1. As soon as GP is in use, the agent achieves near-expert performance (cf. Table 1). *In fine*, Fig. 2 shows



(a) Evolution of return values *(higher is better)*

(b) Final return values at timeout *(higher is better)*

Fig. 2 Evaluation of several methods while *not* using GP. Legend described in text. Runtime is 12 h



(a) Evolution of return values *(higher is better)*

(b) Final return values at timeout *(higher is better)*

Fig. 3 Evaluation of several methods showing the necessity of GP. Legend described in text. Runtime is *12 h*

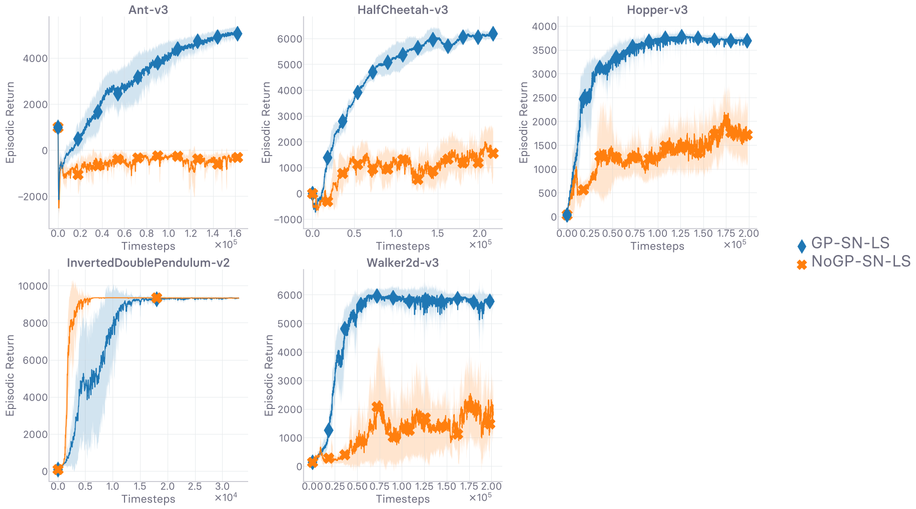


Fig. 4 Evaluation of several methods showing the necessity of GP. Legend described in text. Runtime is *48 h*

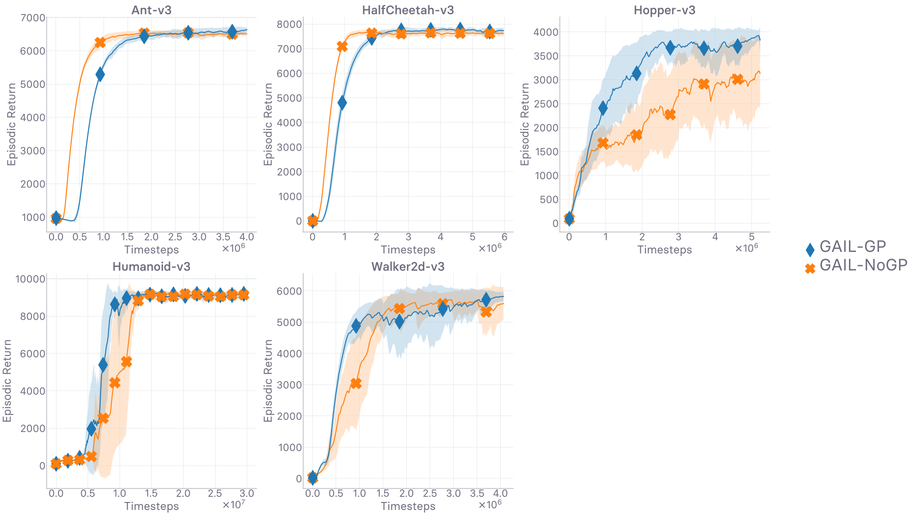
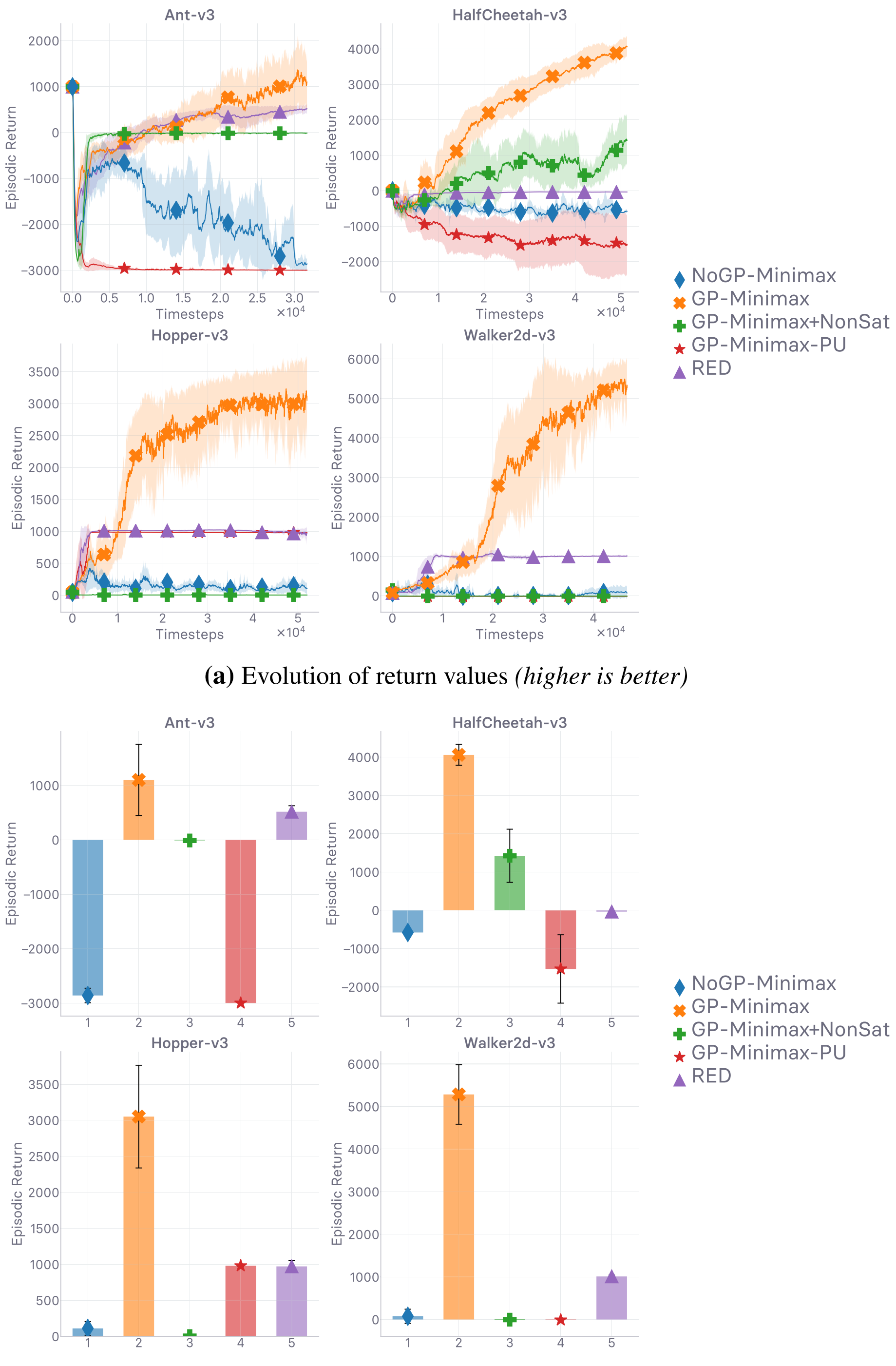


Fig. 5 Ablation study on GP in *on-policy* GAIL. We see that the agent is still able to learn policies achieving peak performance even without GP, in contrast to the off-policy version of the algorithm. In the most difficult environment of the MuJoCo suite (cf. Table 1), `Humanoid`, GP achieves best performance. Runtime is 12 hours

that *without* GP, neither SN nor LS are enough to enable the agent to mimic the expert with high fidelity, while Fig. 3 and Fig. 4 show that *with* GP, extra methods such as LS barely improve performance. These results support our claim: gradient penalty is, (*empirically*) *necessary* and *sufficient* to ensure near-expert performance in *off-policy* generative adversarial imitation learning, in our computational setting.



(a) Evolution of return values *(higher is better)*

(b) Final return values at timeout *(higher is better)*

Fig. 6 Evaluation of several alternate reward formulations. Legend described in text. Runtime is 12 hours

We also conducted an ablation of GP in the *on-policy* setting, reported in Fig. 5. We see that across the range of environments, GP does not assume the same decisive role as in the off-policy setting. In fact, the agent reaches peak performance earlier *without* GP in two challenging environments, `Ant` and `HalfCheetah`, out of the five considered. Nevertheless, it still allows the agent to attain peak empirical return faster in `Hopper`, `Walker2d`, and perhaps most strikingly, in the extremely complex `Humanoid` environment. All in all, while GP can help in the on-policy setting, in is not *necessary* as in the off-policy setting studied in this work. In line with the analyses led in Sects. 5.1–5.3, the results of Fig. 5 somewhat corroborate our claim that the presence of bootstrapping in the policy evaluation objective creates a *bottleneck*, that can be addressed by enforcing a Lipschitz-continuity constraint—GP—on the reward learned for imitation.

Figure 6 compares SAM, with and without GP, against several alternate versions of the objective used to train the surrogate reward for imitation. We introduce the following new handles to denote these methods. *"RED"* means that the random expert distillation (RED) (Wang et al. 2019) method is used to learn the imitation reward, replacing the adversarial one in SAM. RED is based on random network distillation (RND) (Burda et al. 2018), an exploration method using the prediction error of a learned network against a random fixed target as a measure of novelty, and use it to craft a reward bonus. Instead of updating the network while training to keep the novelty estimate tuned to the current exploration level of the agent, RED trains the RND predictor network to predict the random fixed target on the expert dataset *before* training the policy. RED then uses the prediction error to assemble a reward signal for the imitation agent, who is rewarded *more* if the actions it picks are deemed *not novel*, as that means the agent's occupancy measure matches the occupancy of what has been seen before, i.e. the expert dataset. As such, RED is a technique that rewards the agent for matching the distribution support of the expert policy π_e. Note, as opposed to adversarial imitation, the RED reward is not updated during training, which technically protects it from overfitting. *"PU"* means that we learn the reward via adversarial imitation, but using the discriminator objective recently proposed in positive-unlabeled (PU) GAIL (Xu and Denil 2019). Briefly, the method considers that while the expert-generated samples are positive-labels, the agent-generated ones are unlabeled (as opposed to negative-labeled). Intuitively, it should prevent the discriminator overfitting on irrelevant features when it becomes difficult for the discriminator to tell agent and expert apart.

The wrapping mechanism—consisting in wrapping the *absorbing* transitions, which we described in Sect. 4—is used in every experiment reported in Fig. 6, *including RED*. In addition, note, we only use GP in the adversarial context we introduced it in. We do not use GP with RED. Each technique is re-implemented based on the associated paper, with the same hyper-parameters, with the exception of RED: instead of using the per-environment scale for the prediction loss on which the RED reward is built, we keep a running estimate of the standard deviation of this prediction loss and rescale said prediction loss with its running standard deviation. This modification is consistent with the rescaling done in the paper RED is based on RND. By contrast, the per-environment scales in RED's official implementation span several orders of magnitude (four). We here opt for environment-agnostic methods.

The results in Fig. 6 show that the wrapping techniques introduced in Kostrikov et al. (2019) and described in Sect. 4 increases performance overall. Like we have shown before in Figs. 2, 3, and 4, not using GP causes a considerable drop in performance. PU prevents the agent to learn an expert-like policy, in every environment. Note, while the comparison is fair, PU was introduced in *visual* tasks. In particular, we see that, in

Hopper, PU’s empirical return hits a plateau at about 1000 reward units (*abbrv.* r.u.). We observe the exact same phenomenon with RED, for which it occurs in *every* environment. This is caused by the agent being stuck performing the same sub-optimal actions, accumulating sub-optimal outcomes until episode termination artificially triggered by timeout. The agent exploits the fact that it has a lifetime upper-bounded by said timeout and is therefore *biased* by its *survival* (survival bias, cf. Sect. 4). The RED agents are in effect staying alive until termination, and therefore avoid falling down (organic trigger) until the timeout (artificial trigger) is reached. While the reward used in RED is not negative, the agent quickly reaches a performance level at which all the rewards are almost identical—since the RED reward is trained *beforehand*, with no chance of adaptive tuning like training the reward *at the same time* allows in this work, and since RED’s score is based on how the agent and expert distribution match. Once the agent is similar enough to the expert, it always gets the same rewards and has therefore no incentive to resemble the expert with higher fidelity. Instead, it is content and just tries to live through the episode. This propensity to survival bias explains why such care was taken to hand-tune its scale. Finally, even though wrapping absorbing transitions generally improves performance, Fig. 6 shows that survival bias is avoided even *without* it (occurrence in Hopper has been overcome).

The results in Fig. 3 provide empirical evidence that enforcing Lipschitz-continuity on D_φ over the input space via the gradient regularization [cf. Eq. (14)] is *necessary* and *sufficient* for the agent to achieve expert performance in the considered off-policy setting. We therefore ask the question: is the positive impact that GP has on training imitation policies via bootstrapping explained (a) by its *direct* effect on the reward smoothness, or (b) by its *indirect* effect on the state-action value smoothness? We argue that *both* contribute to the stability and performance of the studied method. While point (a) is intuitive from the analyses laid out in Sects. 5.1–5.3, we believe that point (b) deserves further analysis and discussion. As such, we derive theoretical results to qualify, both qualitatively and quantitatively, the Lipschitz-continuity that is potentially *implicitly enforced* on the state-action value when assuming the Lipschitz-continuity of the reward. These results are reported in Sect. 6.1, and will hopefully help us answer the previous question. A discussion of the *indirect* effect and how it compares to the direct effect implemented by target smoothing is carried out in Sect. 6.2.4.

6 Pushing the analysis further: robustness guarantees and provably more robust extension

6.1 Robustness guarantees: state-action value Lipschitzness

In this section, we ultimately show that enforcing a Lipschitzness constraint on the reward r_φ has the effect of enforcing a Lipschitzness constraint on the associated state-action value Q_φ. Note, Q_φ is the *real* Q-value derived from r_φ, while Q_ω is a function approximation of it. We discuss this point in more detail in Sect. 6.2. We characterize and discuss the conditions under which such result is satisfied, as well as how the exhibited Lipschitz constant for Q_φ relates to the one enforced on r_φ. We work in the *episodic* setting, i.e. with a finite-horizon T, which is achieved by assuming that $\gamma = 0$ once an absorbing state is reached. Note, since we optimize over mini-batches in practice, nothing guarantees that the Lipschitz constraint is satisfied by the learned function approximation *globally* across the whole joint space $\mathcal{S} \times \mathcal{A}$, at every training iteration. In such setting, we are therefore reduced to *local* Lipschitzness, defined as Lipschitzness in neighborhoods around samples at which the constraint is applied. The provenance of these samples is not the focus of this theoretical section and assume they are agent-generated. We study the effect of enforcing Lipschitzness constraints on other data distributions in Sect. 6.3.

Notations Given a function $f : \mathbb{R}^n \times \mathbb{R}^m \rightarrow \mathbb{R}^d$, taking the pair of vectors (x, y) as inputs, we denote by $\nabla_{x,y} f$ the pair of Jacobians associated with x and y, $\nabla_x f$ and $\nabla_y f$ respectively, which are rectangular matrices in $\mathbb{R}^{d \times n}$ and $\mathbb{R}^{d \times n}$ respectively. Now that the stable concepts and notations have been laid out, we introduce the variables x_i and y_i, indexed by $i \in \mathcal{I} \subseteq \mathbb{N}$. Note, indices i's do not depict different occurrences of the x variable: the x_i's and y_i's are distinct variables. These families of variables will enable us to formalize the Jacobian of f with respect to (x_i, y_i) evaluated at $(x_{i'}, x_{i'})$, defined as $(\frac{\mathrm{d}f(x_{i'}, y_{i'})}{\mathrm{d}x_i}, \frac{\mathrm{d}f(x_{i'}, y_{i'})}{\mathrm{d}y_i})$, where $i' \in \mathcal{I}, i' \geq i$. To lighten the notations, we overload the symbol ∇ and introduce the shorthands $\nabla^i_x[f]_{i'} := \frac{\mathrm{d}f(x_{i'}, y_{i'})}{\mathrm{d}x_i}$ and $\nabla^i_y[f]_{i'} := \frac{\mathrm{d}f(x_{i'}, y_{i'})}{\mathrm{d}y_i}$. By analogy, the shorthand $\nabla^i_{x,y}[f]_{i'}$ denotes the pair $(\nabla^i_x[f]_{i'}, \nabla^i_y[f]_{i'})$. In this work, the difference between the index of derivation i and the index of evaluation i', $i - i' \leq 0$ will be referred to as *gap*. We use $\| \cdot \|_F$ to denote the Frobenius norm, which a) is naturally defined over rectangular matrices in $\mathbb{R}^{m \times n}$ and b) is *sub-multiplicative*: $\|UV\|_F \leq \|U\|_F \, \|V\|_F$, for U and V rectangular with compatible sizes (provable via Cauchy-Schwarz inequality). In proofs, we use "$\otimes$" for matrix multiplication, to avoid collisions with the scalar product.

Lemma 1 (Recursive inequality—induction step) *Let the MDP with which the agent interacts be deterministic, with the dynamics of the environment determined by the function $f : \mathcal{S} \times \mathcal{A} \rightarrow \mathcal{S}$. The agent follows a deterministic policy $\mu : \mathcal{S} \rightarrow \mathcal{A}$ to map states to actions, and receives rewards from $r_\varphi : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ upon interaction. The functions f, μ and r_φ need be C^0 and differentiable over their respective input spaces. This property is satisfied by the usual neural network function approximators. The "almost-everywhere" case can be derived from this lemma without major changes (relevant when at least one activation function is only differentiable almost-everywhere, ReLU).* (*a*) *Under the previous assumptions, for $k \in [0, T - t - 1] \cap \mathbb{N}$ the following recursive inequality is verified:*

$$\|\nabla^t_{s,a}[r_\varphi]_{t+k+1}\|^2_F \leq C_t \, \|\nabla^{t+1}_{s,a}[r_\varphi]_{t+k+1}\|^2_F \tag{15}$$

where $C_t := A^2_t \max(1, B^2_{t+1})$, A_t and B_t being defined as the supremum norms associated with the Jacobians of f and μ respectively, with values in $\mathbb{R} \cup \{+\infty\}$:

$$\forall t \in [0, T] \cap \mathbb{N}, \quad \begin{cases} A_t := \|\nabla^t_{s,a}[f]_t\|_\infty = \sup\left\{ \|\nabla^t_{s,a}[f]_t\|_F \ : \ (s_t, a_t) \in \mathcal{S} \times \mathcal{A} \right\} \\ B_t := \|\nabla^t_s[\mu]_t\|_\infty = \sup\left\{ \|\nabla^t_s[\mu]_t\|_F \ : \ s_t \in \mathcal{S} \right\} \end{cases} \tag{16}$$

(*b*) *Additionally, by introducing time-independent upper bounds $A, B \in \mathbb{R} \cup \{+\infty\}$ such that $\forall t \in [0, T] \cap \mathbb{N}$, $A_t \leq A$ and $B_t \leq B$, the recursive inequality becomes:*

$$\|\nabla^t_{s,a}[r_\varphi]_{t+k+1}\|^2_F \leq C \, \|\nabla^{t+1}_{s,a}[r_\varphi]_{t+k+1}\|^2_F \tag{17}$$

where $C := A^2 \max(1, B^2)$ is the time-independent counterpart of C_t.

Proof of Lemma 1(a) First, we take the derivative with respect to each variable separately:

$$\nabla_s^t[r_\varphi]_{t+k+1} = \frac{\mathrm{d}r_\varphi(s_{t+k+1}, a_{t+k+1})}{\mathrm{d}s_t} \tag{18}$$

$$= \frac{\mathrm{d}r_\varphi\big(f(s_{t+k}, a_{t+k}), \mu(f(s_{t+k}, a_{t+k}))\big)}{\mathrm{d}s_t} \tag{19}$$

$$= \frac{\mathrm{d}r_\varphi(s_{t+k+1}, a_{t+k+1})}{\mathrm{d}s_{t+1}} \otimes \frac{\mathrm{d}f(s_t, a_t)}{\mathrm{d}s_t} \tag{20}$$

$$\begin{aligned} &+ \frac{\mathrm{d}r_\varphi(s_{t+k+1}, a_{t+k+1})}{\mathrm{d}a_{t+1}} \otimes \frac{\mathrm{d}\mu(s_{t+1})}{\mathrm{d}s_{t+1}} \otimes \frac{\mathrm{d}f(s_t, a_t)}{\mathrm{d}s_t} \\ &= \nabla_s^{t+1}[r_\varphi]_{t+k+1} \otimes \nabla_s^t[f]_t + \nabla_a^{t+1}[r_\varphi]_{t+k+1} \otimes \nabla_s^{t+1}[\mu]_{t+1} \otimes \nabla_s^t[f]_t \end{aligned} \tag{21}$$

$$\nabla_a^t[r_\varphi]_{t+k+1} = \frac{\mathrm{d}r_\varphi(s_{t+k+1}, a_{t+k+1})}{\mathrm{d}a_t} \tag{22}$$

$$= \frac{\mathrm{d}r_\varphi\big(f(s_{t+k}, a_{t+k}), \mu(f(s_{t+k}, a_{t+k}))\big)}{\mathrm{d}a_t} \tag{23}$$

$$= \frac{\mathrm{d}r_\varphi(s_{t+k+1}, a_{t+k+1})}{\mathrm{d}s_{t+1}} \otimes \frac{\mathrm{d}f(s_t, a_t)}{\mathrm{d}a_t} \tag{24}$$

$$\begin{aligned} &+ \frac{\mathrm{d}r_\varphi(s_{t+k+1}, a_{t+k+1})}{\mathrm{d}a_{t+1}} \otimes \frac{\mathrm{d}\mu(s_{t+1})}{\mathrm{d}s_{t+1}} \otimes \frac{\mathrm{d}f(s_t, a_t)}{\mathrm{d}a_t} \\ &= \nabla_s^{t+1}[r_\varphi]_{t+k+1} \otimes \nabla_a^t[f]_t + \nabla_a^{t+1}[r_\varphi]_{t+k+1} \otimes \nabla_s^{t+1}[\mu]_{t+1} \otimes \nabla_a^t[f]_t \end{aligned} \tag{25}$$

By assembling the norm with respect to both input variables, we get:

$$\begin{aligned} &\|\nabla_{s,a}^t[r_\varphi]_{t+k+1}\|_F^2 \\ &\quad = \|\nabla_s^t[r_\varphi]_{t+k+1}\|_F^2 + \|\nabla_a^t[r_\varphi]_{t+k+1}\|_F^2 \end{aligned} \tag{26}$$

$$= \|\nabla_s^{t+1}[r_\varphi]_{t+k+1} \otimes \nabla_s^t[f]_t + \nabla_a^{t+1}[r_\varphi]_{t+k+1} \otimes \nabla_s^{t+1}[\mu]_{t+1} \otimes \nabla_s^t[f]_t\|_F^2 \tag{27}$$

$$\begin{aligned} &+ \|\nabla_s^{t+1}[r_\varphi]_{t+k+1} \otimes \nabla_a^t[f]_t + \nabla_a^{t+1}[r_\varphi]_{t+k+1} \otimes \nabla_s^{t+1}[\mu]_{t+1} \otimes \nabla_a^t[f]_t\|_F^2 \\ &\leq \|\nabla_s^{t+1}[r_\varphi]_{t+k+1} \otimes \nabla_s^t[f]_t\|_F^2 \qquad \blacktriangleright \textit{triangular inequality} \end{aligned} \tag{28}$$

$$\begin{aligned}
&+ \|\nabla_a^{t+1}[r_\varphi]_{t+k+1} \otimes \nabla_s^{t+1}[\mu]_{t+1} \otimes \nabla_s^t[f]_t\|_F^2 \\
&+ \|\nabla_s^{t+1}[r_\varphi]_{t+k+1} \otimes \nabla_a^t[f]_t\|_F^2 \\
&+ \|\nabla_a^{t+1}[r_\varphi]_{t+k+1} \otimes \nabla_s^{t+1}[\mu]_{t+1} \otimes \nabla_a^t[f]_t\|_F^2
\end{aligned} \tag{29}$$

$$\begin{aligned}
\leq \|\nabla_s^{t+1}[r_\varphi]_{t+k+1}\|_F^2 \, \|\nabla_s^t[f]_t\|_F^2 \qquad &\blacktriangleright \textit{sub-multiplicativity} \\
+ \|\nabla_a^{t+1}[r_\varphi]_{t+k+1}\|_F^2 \, \|\nabla_s^{t+1}[\mu]_{t+1}\|_F^2 \, \|\nabla_s^t[f]_t\|_F^2& \\
+ \|\nabla_s^{t+1}[r_\varphi]_{t+k+1}\|_F^2 \, \|\nabla_a^t[f]_t\|_F^2& \\
+ \|\nabla_a^{t+1}[r_\varphi]_{t+k+1}\|_F^2 \, \|\nabla_s^{t+1}[\mu]_{t+1}\|_F^2 \, \|\nabla_a^t[f]_t\|_F^2&
\end{aligned} \tag{30}$$

$$\begin{aligned}
= \|\nabla_s^{t+1}[r_\varphi]_{t+k+1}\|_F^2 \left(\|\nabla_s^t[f]_t\|_F^2 + \|\nabla_a^t[f]_t\|_F^2\right) \qquad &\blacktriangleright \textit{factorization} \\
+ \|\nabla_a^{t+1}[r_\varphi]_{t+k+1}\|_F^2 \, \|\nabla_s^{t+1}[\mu]_{t+1}\|_F^2 \left(\|\nabla_s^t[f]_t\|_F^2 + \|\nabla_a^t[f]_t\|_F^2\right)& \\
= \|\nabla_s^{t+1}[r_\varphi]_{t+k+1}\|_F^2 \, \|\nabla_{s,a}^t[f]_t\|_F^2 \qquad &\blacktriangleright \textit{total norm} \\
+ \|\nabla_a^{t+1}[r_\varphi]_{t+k+1}\|_F^2 \, \|\nabla_s^{t+1}[\mu]_{t+1}\|_F^2 \, \|\nabla_{s,a}^t[f]_t\|_F^2&
\end{aligned} \tag{31}$$

Let A_t, B_t and C_t be time-dependent quantities defined as:

$$\forall t \in [0, T] \cap \mathbb{N}, \quad \begin{cases} A_t := \|\nabla_{s,a}^t[f]_t\|_\infty = \sup\left\{\|\nabla_{s,a}^t[f]_t\|_F \; : \; (s_t, a_t) \in \mathcal{S} \times \mathcal{A}\right\} \\ B_t := \|\nabla_s^t[\mu]_t\|_\infty = \sup\left\{\|\nabla_s^t[\mu]_t\|_F \; : \; s_t \in \mathcal{S}\right\} \\ C_t := A_t^2 \max(1, B_{t+1}^2) \end{cases} \tag{32}$$

Finally, by substitution, we obtain:

$$\|\nabla_{s,a}^t[r_\varphi]_{t+k+1}\|_F^2 \leq A_t^2 \|\nabla_s^{t+1}[r_\varphi]_{t+k+1}\|_F^2 + A_t^2 B_{t+1}^2 \|\nabla_a^{t+1}[r_\varphi]_{t+k+1}\|_F^2 \tag{33}$$

$$\leq A_t^2 \max(1, B_{t+1}^2)\left(\|\nabla_s^{t+1}[r_\varphi]_{t+k+1}\|_F^2 + \|\nabla_a^{t+1}[r_\varphi]_{t+k+1}\|_F^2\right) \tag{34}$$

$$= A_t^2 \max(1, B_{t+1}^2) \, \|\nabla_{s,a}^{t+1}[r_\varphi]_{t+k+1}\|_F^2 \qquad \blacktriangleright \textit{total norm} \tag{35}$$

$$= C_t \, \|\nabla_{s,a}^{t+1}[r_\varphi]_{t+k+1}\|_F^2 \qquad \blacktriangleright C_t\textit{definition} \tag{36}$$

which concludes the proof of Lemma 1(a). □

Proof of Lemma 1(b) By introducing time-independent upper bounds A and B such that $A_t \leq A$ and $B_t \leq B \; \forall t \in [0, T] \cap \mathbb{N}$, as well as $C := A^2 \max(1, B^2)$, we obtain, by substitution in Eq. (35):

$$\|\nabla_{s,a}^t[r_\varphi]_{t+k+1}\|_F^2 \leq A^2 \max(1, B^2) \, \|\nabla_{s,a}^{t+1}[r_\varphi]_{t+k+1}\|_F^2 \tag{37}$$

$$= C \, \|\nabla_{s,a}^{t+1}[r_\varphi]_{t+k+1}\|_F^2 \tag{38}$$

which concludes the proof of Lemma 1(b). □

Lemma 1 tells us how the norm of the Jacobian associated with a gap between derivation and evaluation indices equal to $t+1$ relate to the norm of the Jacobian associated with a gap equal to t. We will use this recursive property to prove our first theorem, Theorem 1. Additionally, from this point forward, we will use the time-independent upper-bounds exclusively, i.e. Lemma 1(b).

Theorem 1 (Gap-dependent reward Lipschitzness) *In addition to the assumptions laid out in Lemma 1, we assume that the function r_φ is δ-Lipschitz over $\mathcal{S}\times\mathcal{A}$. Since r_φ is C^0 and differentiable over $\mathcal{S}\times\mathcal{A}$, this assumption can be written as $\|\nabla^u_{s,a}[r_\varphi]_u\|_F \leq \delta$, where $u \in [0,T]\cap\mathbb{N}$. (a) Then, under these assumptions, the following is verified:*

$$\|\nabla^t_{s,a}[r_\varphi]_{t+k}\|^2_F \leq \delta^2 \prod_{u=0}^{k-1} C_{t+u} \tag{39}$$

where $k \in [0,T]\cap\mathbb{N}$ and C_v is defined as in Lemma 1(a), $\forall v \in [0,T]\cap\mathbb{N}$. (b) Additionally, by involving the time-independent upper bounds introduced in Lemma 1(b), we have the following:

$$\|\nabla^t_{s,a}[r_\varphi]_{t+k}\|^2_F \leq C^k\,\delta^2 \tag{40}$$

where $k \in [0,T]\cap\mathbb{N}$ and C is defined as in Lemma 1(b).

Proof of Theorem 1(a) We will prove Theorem 1(a) by induction.

Let us introduce the dummy variable v, along with the induction hypothesis for v:

$$\|\nabla^t_{s,a}[r_\varphi]_{t+v}\|^2_F \leq \delta^2 \prod_{u=0}^{v-1} C_{t+u} \qquad \blacktriangleright \textit{induction hypothesis} \tag{41}$$

where v represents the gap between the derivation timestep and the evaluation timestep.

Step 1: initialization When the gap $v=0$, Eq. (41) becomes $\|\nabla^t_{s,a}[r_\varphi]_t\|^2_F \leq \delta^2$, $\forall t \in [0,T]\cap\mathbb{N}$, which is trivially verified since it exactly corresponds to Theorem 1's main assumption.

Step 2: induction Let us assume that Eq. (41) is verified for v fixed, and show that Eq. (41) is satisfied when the gap is equal to $v+1$.

$$\|\nabla^t_{s,a}[r_\varphi]_{t+v+1}\|^2_F \leq C_t\,\|\nabla^{t+1}_{s,a}[r_\varphi]_{t+v+1}\|^2_F \qquad \blacktriangleright \textit{Lemma } 1\text{(a)} \tag{42}$$

$$\leq C_t\,\delta^2 \prod_{u=0}^{v-1} C_{t+1+u} \qquad \blacktriangleright \text{Eq. (41) } \textit{since gap isv, at } t+1 \tag{43}$$

$$= C_t\,\delta^2 \prod_{u=1}^{v} C_{t+u} \qquad \blacktriangleright \textit{index shift} \tag{44}$$

$$= \delta^2 \prod_{u=0}^{v} C_{t+u} \qquad \blacktriangleright \textit{repack product} \tag{45}$$

Equation (41) is therefore satisfied for $v + 1$ when assumed at v, which proves the induction step.

Step 3: conclusion Since Eq. (41) has been verified for both the initialization and induction steps, the hypothesis is valid $\forall v \in [0, T] \cap \mathbb{N}$, which concludes the proof of Theorem 1(a). □

Proof of Theorem 1b We will prove Theorem 1(b) by induction.

Let us introduce the dummy variable v, along with the induction hypothesis for v:

$$\|\nabla^{t}_{s,a}[r_\varphi]_{t+v}\|^2_F \leq C^v\,\delta^2 \qquad \blacktriangleright \textit{induction hypothesis} \tag{46}$$

where v represents the gap between the derivation timestep and the evaluation timestep.

Step 1: initialization When the gap $v = 0$, Eq. (46) becomes $\|\nabla^{t}_{s,a}[r_\varphi]_t\|^2_F \leq \delta^2$, $\forall t \in [0, T] \cap \mathbb{N}$, which is trivially verified since it exactly corresponds to Theorem 1's main assumption.

Step 2: induction Let us assume that Eq. (46) is verified for v fixed, and show that Eq. (46) is satisfied when the gap is equal to $v + 1$.

$$\|\nabla^{t}_{s,a}[r_\varphi]_{t+v+1}\|^2_F \leq C\,\|\nabla^{t+1}_{s,a}[r_\varphi]_{t+v+1}\|^2_F \qquad \blacktriangleright \textit{Lemma } 1\text{(b)} \tag{47}$$

$$\leq C\,C^v\,\delta^2 \qquad \blacktriangleright \text{EQ. (46) } \textit{since gap isv} \tag{48}$$

$$= C^{v+1}\,\delta^2 \tag{49}$$

Equation (46) is therefore satisfied for $v + 1$ when assumed at v, which proves the induction step.

Step 3: conclusion Since Eq. (46) has been verified for both the initialization and induction steps, the hypothesis is valid $\forall v \in [0, T] \cap \mathbb{N}$, which concludes the proof of Theorem 1(b). □

This result shows that when there is a gap k between the derivation and evaluation indices, the norm of the Jacobian of r_φ is upper-bounded by a *gap-dependent* quantity equal to $\sqrt{C^k}\delta$, over the entire input space. Crucially, this property applies if and only if the gap between the timestep of the derivation variable and the timestep of the evaluation variable is equal to 0, hence the use of the same letter u in the assumption formulation.

Theorem 2 (State-action value Lipschitzness) *We work under the assumptions laid out in both Lemma 1 and Theorem 1, and repeat the main lines here for Theorem 2 to be self-contained: (a) the functions f, μ and r_φ are C^0 and differentiable over their respective input spaces, and (b) the function r_φ is δ-Lipschitz over $\mathcal{S} \times \mathcal{A}$, i.e. $\|\nabla^{u}_{s,a}[r_\varphi]_u\|_F \leq \delta$, where $u \in [0, T] \cap \mathbb{N}$. Then the quantity $\nabla^{u}_{s,a}[Q_\varphi]_u$ exists $\forall u \in [0, T] \cap \mathbb{N}$, and verifies:*

$$\|\nabla^{t}_{s,a}[Q_\varphi]_t\|_F \leq \begin{cases} \delta\sqrt{\dfrac{1-\left(\gamma^2 C\right)^{T-t}}{1-\gamma^2 C}}, & \text{if } \gamma^2 C \neq 1 \\ \delta\sqrt{T-t}, & \text{if } \gamma^2 C = 1 \end{cases} \tag{50}$$

$\forall t \in [0, T] \cap \mathbb{N}$, *where* $C := A^2 \max(1, B^2)$, *with A and B time-independent upper bounds of* $\|\nabla^t_{s,a}[f]_t\|_\infty$ *and* $\|\nabla^t_s[\mu]_t\|_\infty$ *respectively [see Eq.* (32) *for definitions of the supremum norms].*

Proof of Theorem 2 With finite horizon T, we have $Q_\varphi(s_t, a_t) := \sum_{k=0}^{T-t-1} \gamma^k \, r_\varphi(s_{t+k}, a_{t+k})$, $\forall t \in [0, T] \cap \mathbb{N}$, since f, μ, and r_φ are all deterministic (no expectation). Additionally, since r_φ is assumes to be C^0 and differentiable over $\mathcal{S} \times \mathcal{A}$, Q_φ is by construction also C^0 and differentiable over $\mathcal{S} \times \mathcal{A}$. Consequently, $\nabla^u_{s,a}[Q_\varphi]_u$ exists, $\forall u \in [0, T] \cap \mathbb{N}$. Since both r_φ and Q_φ are scalar-valued (their output space is $\mathbb{R}$), their Jacobians are the same as their gradients. We can therefore use the linearity of the gradient operator: $\nabla^t_{s,a}[Q_\varphi]_t = \sum_{k=0}^{T-t-1} \gamma^k \, \nabla^t_{s,a}[r_\varphi]_{t+k}$, $\forall t \in [0, T] \cap \mathbb{N}$.

$$\|\nabla^t_{s,a}[Q_\varphi]_t\|^2_F = \left\| \sum_{k=0}^{T-t-1} \gamma^k \, \nabla^t_{s,a}[r_\varphi]_{t+k} \right\|^2_F \qquad \blacktriangleright \textit{operator's linearity} \tag{51}$$

$$\leq \sum_{k=0}^{T-t-1} \gamma^{2k} \, \|\nabla^t_{s,a}[r_\varphi]_{t+k}\|^2_F \qquad \blacktriangleright \textit{triangular inequality} \tag{52}$$

$$\leq \sum_{k=0}^{T-t-1} \gamma^{2k} \, C^k \, \delta^2 \qquad \blacktriangleright \textit{Theorem } 7 \tag{53}$$

$$= \delta^2 \sum_{k=0}^{T-t-1} \left(\gamma^2 C\right)^k \tag{54}$$

When $\gamma^2 C = 1$, we obtain $\|\nabla^t_{s,a}[Q_\varphi]_t\|^2_F = \delta^2 (T - t)$. On the other hand, when $\gamma^2 C \neq 1$:

$$\|\nabla^t_{s,a}[Q_\varphi]_t\|^2_F \leq \delta^2 \, \frac{1 - \left(\gamma^2 C\right)^{T-t}}{1 - \gamma^2 C} \qquad \blacktriangleright \textit{finite sum of geometric series} \tag{55}$$

$$\Longrightarrow \quad \|\nabla^t_{s,a}[Q_\varphi]_t\|^2_F \leq \begin{cases} \delta^2 \, \dfrac{1 - \left(\gamma^2 C\right)^{T-t}}{1 - \gamma^2 C}, & \text{if } \gamma^2 C \neq 1 \\ \delta^2 (T - t), & \text{if } \gamma^2 C = 1 \end{cases} \tag{56}$$

By applying $\sqrt{\cdot}$ (monotonically increasing) to the inequality, we obtain the claimed result. □

Finally, we derive a corollary from Theorem 2 corresponding to the infinite-horizon regime.

Corollary 1 (Infinite-horizon regime) *Under the assumptions of Theorem* 2, *including that* r_φ *is* δ*-Lipschitz over* $\mathcal{S} \times \mathcal{A}$, *and assuming that* $\gamma^2 C < 1$, *we have, in the infinite-horizon regime:*

$$\|\nabla^t_{s,a}[Q_\varphi]_t\|_F \leq \frac{\delta}{\sqrt{1-\gamma^2 C}} \tag{57}$$

which translates into Q_φ being $\frac{\delta}{\sqrt{1-\gamma^2 C}}$-Lipschitz over $\mathcal{S} \times \mathcal{A}$.

Proof of Corollary 1 We now have $Q_\varphi(s_t, a_t) := \sum_{k=0}^{+\infty} \gamma^k \, r_\varphi(s_{t+k}, a_{t+k})$, $\forall t \in [0, T] \cap \mathbb{N}$, since f, μ, and r_φ are all deterministic and are now working working under the infinite-horizon regime. Considering the changes in Q_φ's definition, the first part of the proof can be done by analogy with the proof of Theorem 2, until Eq. (54), which is our starting point. In this regime, $\gamma^2 C \geq 1$ yields an infinite sum in Eq. (54), which results in an uninformative (because infinite) upper-bound on $\|\nabla^t_{s,a}[Q_\varphi]_t\|_F$. On the other hand, when $\gamma^2 C < 1$ (note, we always have $\gamma^2 C \geq 0$ by definition), the infinite sum in Eq. (54) is defined. Since we have shown that $\gamma^2 C < 1$ is the only setting in which the sum is defined, we continue from the infinite-horizon version of Eq. (54) with $\gamma^2 C < 1$ onwards. Hence,

$$\|\nabla^t_{s,a}[Q_\varphi]_t\|_F^2 \leq \delta^2 \sum_{k=0}^{+\infty} \left(\gamma^2 C\right)^k = \frac{\delta^2}{1-\gamma^2 C} \qquad \blacktriangleright \textit{infinite sum of geometric series} \tag{58}$$

Using $\sqrt{\cdot}$ (monotonically increasing) on both sides concludes the proof of Corollary 1. □

To conclude the section, we now give interpretations of the derived theoretical results, discuss the implications of our results, and also exhibit to what extent they transfer to the practical setting.

6.2 Discussion I: implications and limitations of the theoretical guarantees

6.2.1 Function approximation bias

Theorem 2 exhibits the Lipschitz constant of Q_φ when r_φ is δ-Lipschitz. In practice however, the state-action value (or value function) is usually modeled by a neural network, and learned via gradient descent either by using a Monte-Carlo estimate of the collected return as regression target, or by bootstrapping using a subsequent model estimate (Sutton 1988). We therefore have access to a learned estimate Q_ω, as opposed to the real state-action value Q_φ. As such, the results derived in Theorem 2 will transfer favorably into the function approximation setting as Q_ω becomes a better parametric estimate of Q_φ. Note, the reward is denoted by r_φ for the reader to easily distinguish it from the *black-box* reward traditionally returned by the environment. Albeit arbitrary, the notation r_φ allows for the reward to be modeled by a neural network parameterized by the weights φ, and learned via gradient descent, as is indeed the case in this work. Crucially, having control over r_φ in practice allows for the enforcement of constraints, making the δ-Lipschitzness assumption in Theorems 1, 2 and Corollary 1 practically satisfiable via gradient penalization 5.4. It is crucial to note that, while function approximation creates a gap between theory and practice for the Q-value (*worse* when bootstrapping), there is a meaningfully lesser gap for the reward as the δ-Lipschitzness constraint is directly enforced on the parametric reward r_φ.

6.2.2 Value lipschitzness

In Corollary 1 we showed that $\|\nabla^t_{s,a}[Q_\varphi]_t\|_F \leq \delta/\sqrt{1-\gamma^2 C}$, in the infinite-horizon regime, when r_φ is assumed δ-Lipschitz over $\mathcal{S}\times\mathcal{A}$, and assuming $\gamma^2 C < 1$. In other words, in this setting, enforcing r_φ to be δ-Lipschitz causes Q_φ to be Δ_∞-Lipschitz, where $\Delta_\infty := \delta/\sqrt{1-\gamma^2 C}$, $C := A^2 \max(1, B^2)$, and A, B are upper-bounds of $\|\nabla^t_{s,a}[f]_t\|_\infty$, $\|\nabla^t_s[\mu]_t\|_\infty$. Starting from the assumption that $\gamma^2 C < 1$, we arrive at $\sqrt{1-\gamma^2 C} < 1$, then $1/\sqrt{1-\gamma^2 C} > 1$, and since $\delta \geq 0$ by definition (cf. Sect. 5.4), we finally get $\Delta_\infty > \delta$. Without loss of generality, consider the case in which r_φ is *not* a contraction, i.e. r_φ is δ -Lipschitz C^0 over $\mathcal{S}\times\mathcal{A}$, with $\delta \geq 1$. As a result, $\Delta_\infty > \delta \geq 1$, i.e. $\Delta_\infty > 1$, which means that, under the considered conditions, Q_φ is *not* a contraction over $\mathcal{S}\times\mathcal{A}$ either. The latter naturally extends to any $u \in \mathbb{R}_+$ that lower-bounds δ: if $\delta > u$, then $\Delta_\infty > u$, $\forall u \in \mathbb{R}_+$. Lipschitz functions and especially contractions are at the core of many fundamental results in dynamics programming, hence also in reinforcement learning. Crucially, the Bellman operator being a contraction causes a fixed point iterative process, such as value iteration (Sutton and Barto 1998), to converge to a unique fixed point whatever the starting iterate of Q. Since we learn Q_φ with temporal-difference learning (Sutton 1988) via a bootstrapped objective, the convergence of our method is a direct consequence of the contractant nature of the Bellman operator. As such the Lipschitzness-centric analysis laid out in this section is complementary to the latter. It provides a characterization of Q_φ's Lipschitzness over the input space $\mathcal{S}\times\mathcal{A}$ as opposed to over iterates, i.e. time. As such, our analysis therefore does not give convergence guarantees of an iterative process, which are already carried over from temporal-difference learning at the core of our algorithm. Rather, we provide *variation upper-bounds* for Q_φ when r_φ has upper-bounded variations: if r_φ is δ-Lipschitz, then Q_φ is Δ_∞-Lipschitz. *In fine*, this result has an immediate corollary, derived previously in this block: if the variations of r_φ are lower-bounded by δ, then the variations of Q_φ are lower-bounded by $\Delta_\infty > \delta$.

6.2.3 Compounding variations

The relative position of $\gamma^2 C$ with respect to 1 is instrumental in the behavior of the exhibited variation bounds, in both the finite- and infinite-horizon settings. In the latter, we see that the upper-bound gets to infinity when $\gamma^2 C$ (non-negative by definition, and lower than 1 as necessary condition for the infinite sum to exist) gets closer to 1 from below. In the former, we focus on the $\gamma^2 C \neq 1$ case, as in the other case, the bound does not even depend on $\gamma^2 C$. As such, we study the value of $\|\nabla^t_{s,a}[Q_\varphi]_t\|_F$'s upper-bound in the finite-horizon setting when $\gamma^2 C \neq 1$, dubbed $\Delta_t := \delta\sqrt{1-(\gamma^2 C)^{T-t}/1-\gamma^2 C}$. Beforehand, we would remind the reader how the bounded quantity should behave throughout an episode. Since Q_φ is defined as the expected sum of *future* rewards r_φ, predicting such value should get increasingly tainted with uncertainty as it tries to predict across long time ranges. As such, predicting Q_φ at time $t=0$ is the most challenging, as it corresponds to the value of an entire trajectory, whereas predicting Q_φ at time $t=T$ is the easiest (equal to last reward r_φ). Higher horizons T consequently make the prediction task more difficult, as do discount factors γ closer to 1. We now discuss Δ_t. As long as $\gamma^2 C \neq 1$, Δ_t gets to 0 as t gets to T. This is consistent with the previous reminder: as t gets to T, the Q_φ estimation task becomes easier, hence the variation bound (Δ_t) due to prediction uncertainty should decrease to 0. As t gets to 0 however, the behavior of Δ_t depends on the value of $\gamma^2 C$: if $\gamma^2 C \gg 1$, Δ_t explodes to infinity, whereas for reasonable values of $\gamma^2 C$, Δ_t does not. Since $C := A^2 \max(1, B^2)$, $\gamma^2 C \gg 1$ translates to $((\exists u > 1) : A \gg u) \vee ((\exists v > 1) : B \gg v)$. Let us assume that A (B) not only upper-bounds every A_t (B_t) but is also the tightest time-independent bound: $A := A_{t'}$ ($B := B_{t''}$) where $t' = \underset{t}{\operatorname{argmax}} A_t$ ($t'' = \underset{t}{\operatorname{argmax}} B_t$). We then have $((\exists u > 1)(\exists t') : A_{t'} \gg u) \vee ((\exists v > 1)(\exists t'') : B_{t''} \gg v)$, i.e. $((\exists u > 1)(\exists t') : \|\nabla^{t'}_{s,a}[f]_{t'}\|_\infty \gg u) \vee$ $((\exists v > 1)(\exists t'') : \|\nabla^{t''}_{s}[\mu]_{t''}\|_\infty \gg v)$ over $\mathcal{S} \times \mathcal{A}$. Note, the "or" is inclusive. In other words, if the variations (in space) of the policy or the dynamics are large in the early stage of an episode ($0 \leq t \ll T$), then Δ_t (variation bound on Q_φ) explodes. The exhibited phenomenon is somewhat reminiscent of the compounding of errors isolated in Ross and Bagnell (2010).

6.2.4 Is value lipschitzness enough?

We showed that under mild conditions, and in finite- and infinite- horizon regimes, r_φ Lipschitzness implies Q_φ Lipschitzness, i.e. that if similar state-action are mapped to similar rewards by r_φ, then Q_φ also maps then to similar state-action values. This regularization desideratum is evocative of the *target policy smoothing* add-on introduced in (Fujimoto et al. 2018), already presented earlier in Sect. 4. In short, target policy smoothing perturbs the target action slightly. In effect, the temporal-difference optimization now fits the value estimate against an expectation of *similar* bootstrapped target value estimates. Forcing similar action to have similar values naturally smooths out the value estimate, which by definition emulates the enforcement of a Lipschitzness constraint on the value, and as such mitigates value overfitting which deterministic policies are prone to. While its smoothing effect on the value function is somewhat intuitive, we set out to investigate formally how target policy smoothing affects the optimization dynamics, and particularly to what extent it smooths out the state-action value landscape. Since the function approximator Q_ω is optimized as a supervised learning problem using the traditional squared loss criterion, we first study how perturbing the inputs with additive random noise, denoted by ξ, impacts the optimized criterion, and what kind of behavior it encourages in the predictive function. As such, to lighten the expressions, we consider the supervised criterion $C(x) := (y - f(x))^2$, where $f(x)$ is the predicted vector at the input vector x, and y is the supervised target vector. We also consider, in line with (Fujimoto et al. 2018), that the noise is sampled from a spherical zero-centered Gaussian distribution, omitting here that the noise is truncated for legibility, hence $\xi \sim \mathcal{N}(0, \sigma^2 I)$. The criterion injected with input noise is $C_\xi(x) := C(x + \xi) = (y - f(x + \xi))^2$. Assuming the noise has small amplitude (further supporting the original truncation), we can write the second-order Taylor series expansion of the perturbed criterion near $\xi = 0$, as a polynomial of ξ:

$$C_\xi(x) = C(x) + \sum_i \frac{\partial C}{\partial x_i}\bigg|_x \xi_i + \frac{1}{2} \sum_i \sum_j \frac{\partial^2 C}{\partial x_i \partial x_j}\bigg|_x \xi_i \xi_j + \mathcal{O}(\|\xi\|^3) \tag{59}$$

where $\|\cdot\|$ denotes the Euclidean norm in the appropriate vector space. From this point forward, we assume the noise has a small enough norm to allow the third term, $\mathcal{O}(\|\xi\|^3)$, to be neglected. By integrating over the noise distribution, we obtain:

$$\int C_\xi(x) p(\xi) d\xi = C(x) + \sum_i \frac{\partial C}{\partial x_i}\bigg|_x \int \xi_i p(\xi) d\xi + \frac{1}{2} \sum_i \sum_j \frac{\partial^2 C}{\partial x_i \partial x_j}\bigg|_x \int \xi_i \xi_j p(\xi) d\xi \tag{60}$$

Since the noise is sampled from the zero-centered and spherical distribution $\mathcal{N}(0, \sigma^2 I)$, we have respectively that $\int \xi_i p(\xi) d\xi = 0$ and

$$\int \xi_i \xi_j p(\xi) d\xi = \int \xi_i^2 \delta_{ij} p(\xi) d\xi = \delta_{ij} \int \xi_i^2 p(\xi) d\xi = \delta_{ij} \sigma^2$$

, where δ_{ij} is the Kronecker symbol. By injecting these expressions in Eq. (60), we get:

$$\int C_\xi(x) p(\xi) d\xi = C(x) + \frac{\sigma^2}{2} \sum_i \left. \frac{\partial^2 C}{\partial x_i^2} \right|_x = C(x) + \frac{\sigma^2}{2} \mathrm{Tr}(H_x\, C) \tag{61}$$

where $\mathrm{Tr}(H_x\, C)$ is the trace of the Hessian of the criterion C, *w.r.t.* the input variable x. We now want to express the exhibited regularizer $\mathrm{Tr}(H_x\, C))$ as a function of the derivatives of the prediction function f, and therefore calculate the consecutive derivative sums:

$$\sum_i \left. \frac{\partial C}{\partial x_i} \right|_x = -2 \sum_i \big(y - f(x)\big) \left. \frac{\partial f}{\partial x_i} \right|_x \tag{62}$$

$$\sum_i \left. \frac{\partial^2 C}{\partial x_i^2} \right|_x = 2 \sum_i \left[\left(\left. \frac{\partial f}{\partial x_i} \right|_x \right)^2 - \big(y - f(x)\big) \left. \frac{\partial^2 f}{\partial x_i^2} \right|_x \right] \tag{63}$$

hence,

$$\int C_\xi(x) p(\xi) d\xi = C(x) + \sigma^2 \sum_i \left[\left(\left. \frac{\partial f}{\partial x_i} \right|_x \right)^2 - \big(y - f(x)\big) \left. \frac{\partial^2 f}{\partial x_i^2} \right|_x \right] \tag{64}$$

In fine, we can write, in a more condensed form:

$$\mathbb{E}_\xi[C(x + \xi)] = C(x) + \sigma^2 \Big[\|\nabla_x f\|^2 - \mathrm{Tr}\big(C(x) H_x f\big) \Big] \tag{65}$$

The previous derivations—derived somewhat similarly in Webb (1994) and Bishop (1995)—show that minimizing the criterion with noise injected in the input is equivalent to minimizing the criterion without any noise *and* a regularizer containing norms of both the Jacobian and Hessian of the prediction function f. As raised in Bishop (1995), the second term of the regularizer is unsuitable for the design of a practically viable learning algorithm, since (a) it involves prohibitively costly second-order derivatives, and (b) it is not positive definite, and consequently not lower-bounded, which overall makes the regularizer a bad candidate for an optimization problem loss. Nevertheless, Bishop (1995) further shows that this regularization is equivalent to the use of a standard Tikhonov-like positive-definite regularization scheme involving *only* first-order derivatives, provided the noise has small amplitude—ensured here with a small σ and noise clipping. As such, the regularizer induced by the input noise ξ is equivalent to $\sigma^2 \big[\|\nabla_x f\|^2 \big]$, and by direct analogy, we can say that target policy smoothing induces an implicit regularizer on the TD objective, of the form $\sigma^2 \big[\|\nabla_a Q_{\omega'}\|^2 \big]$, Note, ω' are the target critic parameters, given that target policy smoothing adds noise to the target action, an input of target critic value $Q_{\omega'}$. By construction, the target parameters ω' slowly follow the online parameters ω (cf. Sect. 4). In addition, temporal-difference learning urges Q_ω to move closer to $Q_{\omega'}$ by design [cf. Eq. (3)]. Consequently, properties enforced on one set of parameters should *eventually* be transfered to the other, such that *in fine* both ω and ω' possess the given property only explicitly enforced on one (albeit delayed). Based on this line of reasoning, the temporal-difference learning dynamics and soft target updates should make the theoretically equivalent $\sigma^2\left[\|\nabla_a Q_{\omega'}\|^2\right]$ regularizer enforce smoothness on the online parameters ω too, even if it explicitly only constrains the target weights ω'. All in all, we have shown that target smoothing is equivalent to adding a regularizer to the temporal-difference error to minimize when learning Q_ω, where said regularizer is reminiscent of the gradient penalty regularizer, presented earlier in Eq. (14). As such, target smoothing *does* implement a gradient penalty regularization, but on Q_ω. Crucially, the gradient in the penalty is only taken *w.r.t.* the action dimension, but not *w.r.t.* the state dimension. In spite of the use of target policy smoothing in our method, it was not enough to yield stable learning behaviors, as shown in Sect. 5.5. Gradient penalization was an absolute necessity. Even though both methods encourage Q_ω to be smoother (directly in Fujimoto et al. (2018), and indirectly via reward Lipschitzness in this work), on its own, learning a smooth Q_ω estimate seems not to be *sufficient* for our method to work: learning a smooth r_φ estimate to serve as basis for Q_ω seems to be a *necessary* condition.

6.2.5 Indirect reward regularization

The theoretical guarantees we have derived (cf. Theorems 1, 2 and Corollary 1) all build on the premise that the reward r_φ is δ-Lipschitz over the joint input space $\mathcal{S} \times \mathcal{A}$, i.e. that $\|\nabla^t_{s,a}[r_\varphi]_t\|_F \leq \delta$. Crucially, we do *not* enforce this regularity property *directly* is practice, but instead urge the discriminator D_φ to be k-Lipschitz by restricting the norm of the Jacobian of the latter via regularization [cf. Eq. (2)]. We here set out to figure out to what extent the k-Lipschitzness enforced onto D_φ propagates and transfers to r_φ; in particular, whether it results in the *indicrectly*-urged δ-Lipschitzness of r_φ, with $\delta \neq k$ outside of edge cases. While k is fixed throughout the lifetime of the agent, δ need not be. As such, discussing the behavior of this evolving Lipschitz constant *w.r.t.* the learning dynamics is crucial to better understand *when* the guarantees we have just derived (whose main premise is $\|\nabla^t_{s,a}[r_\varphi]_t\|_F \leq \delta$) apply in practice. As laid out ealier in Sect. 4, in this work, we consider two forms of reward, crafted purely from the scores returned by D_φ: the minimax (saturating) one $r^{\text{MM}}_\varphi := -\log(1 - D_\varphi)$ and the non-saturating one $r^{\text{NS}}_\varphi := \log(D_\varphi)$ (names purposely chosen to echo their counterpart GAN generator loss). Although we opted for the minimax form (based on the ablation study we carried out on the matter, cf. Appendix 6), we here tackle and discuss both forms, as we suspect there could be more to it than just zero-order numerics. Analyzing first-order behavior is the crux of most GAN design breakthroughs, which is far from surprising, considering how intertwined the inner networks are (generator G, and discriminator D). Yet, in adversarial IL, the policy (playing the role of G) does not receive gradients flowing back from D like in GANs. Instead, it gets a reward signal crafted from D's returned scalar value, detached from the computational graph, and try to maximize it over time via *policy*-gradient optimization. The discussion in adversarial IL has thus always limited to the numerics of the reward signal and how to shape it in a way that faciliates the resolution of the task at hand (similarly to how we discuss the impact of its shape when reporting our last empirical findings of Sect. 5.5).

By constrast, we here are interested in the gradients of these rewards ($r_{\varphi,\text{MM}}$ and $r_{\varphi,\text{NS}}$) in this studied adversarial IL context, with the end-goal of characterizing their Lipschitz-continuity (or absence thereof). Their respective Jacobians' norms, under the setting laid out earlier in Sect. 6.1, are $\|\nabla^t_{s,a}[r^{\text{MM}}_\varphi]_t\|_F = \|\nabla^t_{s,a}[D_\varphi]_t\|_F \big/ (1 - D_\varphi(s_t, a_t))$ and

$\|\nabla^t_{s,a}[r^{\text{NS}}_\varphi]_t\|_F = \|\nabla^t_{s,a}[D_\varphi]_t\|_F \big/ D_\varphi(s_t,a_t)$, with $D_\varphi(s_t,a_t) \in (0,1)$ (D_φ's score is wrapped with a sigmoid). As laid out above, we here posit that D_φ is k-Lipschitz-continuous as founding assumption—$\|\nabla^t_{s,a}[D_\varphi]_t\|_F \leq k$. We can now upper-bound the Jacobians' norms unpacked above with the Lipschitz constant of D_φ: $\|\nabla^t_{s,a}[r^{\text{MM}}_\varphi]_t\|_F \leq k \big/ (1 - D_\varphi(s_t,a_t))$ and $\|\nabla^t_{s,a}[r^{\text{NS}}_\varphi]_t\|_F \leq k \big/ D_\varphi(s_t,a_t)$. Since $D_\varphi(s_t,a_t) \in (0,1)$, both denominators (for either reward form) are in (0, 1), which makes the Jacobian's norm of either reward form unbounded over its domain (due to $D_\varphi \to 0$ from above for r^{NS}_φ; due to $D_\varphi \to 1$ from below for r^{MM}_φ), despite the D_φ's k-Lipschitzness. Since treating the entire range of values that *can* be taken by $D_\varphi(s_t,a_t)$, (0, 1), lead us to a dead end, and leaving us unable to upper-bound neither $\|\nabla^t_{s,a}[r^{\text{MM}}_\varphi]_t\|_F$ nor $\|\nabla^t_{s,a}[r^{\text{NS}}_\varphi]_t\|_F$, we now adopt a more granular approach and procede by dichotomy. As such, $\exists\, \ell \in (0,1)$ verifying $0 < \ell \ll 1$ such that $1 \big/ D_\varphi(s_t,a_t) \big($ and as a result also $\|\nabla^t_{s,a}[r^{\text{NS}}_\varphi]_t\|_F \leq k \big/ D_\varphi(s_t,a_t) \big)$ is unbounded when $D_\varphi(s_t,a_t) \in (0,\ell]$ and bounded when $D_\varphi(s_t,a_t) \in (\ell,1)$. Similarly, $\exists\, L \in (0,1)$ verifying $0 \ll L < 1$ such that $1 \big/ (1 - D_\varphi(s_t,a_t)) \big($and as a result also $\|\nabla^t_{s,a}[r^{\text{MM}}_\varphi]_t\|_F \leq k \big/ (1 - D_\varphi(s_t,a_t)) \big)$ is bounded when $D_\varphi(s_t,a_t) \in (0,L]$ and unbounded when $D_\varphi(s_t,a_t) \in (L,1)$. If we were to figure out the *effective* range covered by D_φ's values throughout the learning process, we would maybe be able to exploit the dichotomy.

In practice, the untrained agent initially performs poorly at the imitation task, and is therefore assigned low scores by D_φ (near 0, as "0" is the label assigned to samples from the agent in the classification update D_φ goes through every iteration). As learning progresses, the agent's scores gradually shift towards 1—the label used for expert samples in D_φ's update, and *optimally* converge to the central value of 0.5 in the (0, 1) range that D_φ can describe. Indeed, the *perfect* discriminator consistently predicts scores equal to 0.5 for the agent's actions (Goodfellow 2017): the agent has managed to perfectly confuse D_φ as to where the data it is fed comes from (both sources, expert and agent, are perceived as equiprobable). What matters for $\|\nabla^t_{s,a}[r_\varphi]_t\|_F$ (either form) to be bounded *in practice* is for it to be bounded for values of D_φ in (0, M], where $0.5 \leq M < 1$ (the values *realistically* taken by D_φ throughout the learning process). Since $M < L$ in effect (for L, cf. dichotomy above), we can conclude that $\|\nabla^t_{s,a}[r^{\text{MM}}_\varphi]_t\|_F$ is effectively bounded: $\exists\, \delta$, $0 \leq \delta < +\infty$, such that $\|\nabla^t_{s,a}[r^{\text{MM}}_\varphi]_t\|_F \leq \delta$. We however can not conclude as such for $\|\nabla^t_{s,a}[r^{\text{NS}}_\varphi]_t\|_F$, however close to zero ℓ might be (for ℓ, cf. dichotomy above). It is not rare for D_φ to take 0 as value early in training, which makes $\|\nabla^t_{s,a}[r^{\text{NS}}_\varphi]_t\|_F$ unbounded in the interval described by the values taken by D in practice: (0, M]. Interestingly, when D_φ is near 0 early in training, $\|\nabla^t_{s,a}[r^{\text{MM}}_\varphi]_t\|_F \leq k \big/ (1 - D_\varphi(s_t,a_t)) \approx k$. The lowest upper-bound for $\|\nabla^t_{s,a}[r^{\text{MM}}_\varphi]_t\|_F$ is $\delta \approx k$, and can only happen early in the training process, when D_φ correctly classifies the agent's actions as coming from the agent. In other words, the Lipschitz constant of r^{MM}_φ is at its lowest early in training. Besides, as the agent becomes more proficient at mimicking the expert and therefore collects higher scores from D_φ, δ increases monotonically and grows aways from its initial value k. Compared to the alternative (highest Lipschitz constant early in training and then monotonically decreasing as the scores increase when the agent gets better at the task, nearing the lowest value of k when $D_\varphi \to 1$), which as it turns out is exactly the behavior adopted by r^{NS}_φ, the behavior of r^{MM}_φ is far more desirable.

Crucially, to sum up, r^{NS}_φ is not Lipschitz early in training when the agent would benefit most from regularity in the reward landscape. r^{MM}_φ however *is* Lipschitz-continuous early in training, with the lowest Lipschitz constant of its lifetime, which aligns with the Lipschitz constant enforced on D_φ ($\delta \approx k$). As such, r^{MM}_φ is at its most regular when the agent needs it most (early, when it knows nothing), and then becomes less and less restrictive (the Lipschitz constant δ increases) as the agent collects higher similarity scores with the expert from D_φ. One could therefore see r^{MM}_φ as having built-in *"training wheels"*, which gradually phase out as the agent becomes better, providing less safety as the agent becomes more proficient at the imitation task. To conclude this discussion point, with the minimax reward form $r_\varphi := r_\varphi^{\text{MM}}$, we have $\|\nabla_{s,a}^t[D_\varphi]_t\|_F \leq k \implies \|\nabla_{s,a}^t[r_\varphi]_t\|_F \leq \delta$ in practice. This means that the premise of our theoretical guarantees consisting in positing that the reward is δ-Lipschitz-continuous *can* be satisfied in practice by enforcing k-Lipschitz-continuity on D_φ via gradient penalty regularization [cf. Eq. (14)]. This is *not* the case when $r_\varphi := r_\varphi^{\text{NS}}$. We propose this analytical observation as an explanation as to why using r_φ^{NS} yields such poor results in our reported ablation, cf. Appendix 6. Our discussion detaches itself from the one adopting a zero-order numerics scope, laid out in Sect. 5.5, by discussing first-order numerics instead, which blends into our Lipschitzness narrative.

6.2.6 Local smoothness

The local Lipschitzness assumption is reminiscent of many theoretical results in the study of robustness to adversarial examples. Notably, Yang et al. (2020) shows that local Lipschitzness is correlated with empirical robustness and accuracy in various benchmark datasets. As mentioned when we justified the *local* nature of the Lipschitz-continuity notion tackled in this work (cf. Definition 1), we optimize the different modules over mini-batches of samples. While forcing the constraint to be satisfied globally might be feasible in some low-dimensional supervised or unsupervised learning problems, the notion of fixed dataset does not exist *a priori* in reinforcement learning. Section 6.3 describes, compares and discusses the effect of *where* the local Lipschitzness constraint is enforced (e.g. expert demonstration manifold, fictitious replay experiences). Wherever the regularizer is applied, the constraint is local nonetheless. One can therefore not guarantee that the δ-Lipschitz-continuity of r_φ, formalized as $\|\nabla_{s,a}^t[r_\varphi]_t\|_F \leq \delta$, and urged by enforcing $\|\nabla_{s,a}^t[D_\varphi]_t\|_F \leq k$ via gradient penalization (cf. our previous discussion on indirect reward regularization in Sect. 6.2.5), will be satisfied *everywhere* in $\mathcal{S} \times \mathcal{A}$. Plus, considering that Theorem 2 and Corollary 1 rely on the satisfaction of the constraint on r_φ along every trajectory, which is likely not to be verified in practice, we can say with high confidence that the constraint on Q_φ, $\|\nabla_{s,a}^t[Q_\varphi]_t\|_F \leq \Delta_\infty$, will not be satisfied over the whole joint input space either. Still, we can hope to enhance the coverage of the subspace on which the constraint $\|\nabla_{s,a}^t[r_\varphi]_t\|_F \leq \delta$ is satisfied, dubbed $\mathfrak{C}$, by doing more r_φ learning updates with the regularizer—technically, D_φ learning updates encouraging D_φ to satisfy $\|\nabla_{s,a}^t[D_\varphi]_t\|_F \leq k$ via gradient penalization, cf. Eq. (14). From this point onward, we will qualify a state-action pair (s_t, a_t)—equivalently, an action a_t in a given state s_t—as "$\mathfrak{C}$-*valid*" if it belongs to $\mathfrak{C} \ni (s_t, a_t)$, i.e. if r_φ is δ-Lipschitz, verifying $\|\nabla_{s,a}^t[r_\varphi]_t\|_F \leq \delta$. Note, the notion of $\mathfrak{C}$-validity is inherently local, since we have defined the notion for a single given input pair (s_t, a_t). As such, future statements about $\mathfrak{C}$-validity will all be local ones by essence. In addition, despite having $\|\nabla_{s,a}^t[D_\varphi]_t\|_F \leq k \implies \|\nabla_{s,a}^t[r_\varphi]_t\|_F \leq \delta$ *in practice* for the minimax reward form (cf. our previous discussion on indirect reward regularization in Sect. 6.2.5), there is not an exact equivalence between r_φ being δ-Lipschitz and D_φ being k-Lipschitz in theory. Therefore, we will qualify a state-action pair (s_t, a_t)—equivalently, an action a_t in a given state s_t—as *"approximately* $\mathfrak{C}$-*valid"* if D_φ is k-Lipschitz, verifying $\|\nabla_{s,a}^t[D_\varphi]_t\|_F \leq k$. As it has been made clear by now, D_φ's k-Lipschitzness is encouraged by plugging a gradient penalty regularizer $\mathfrak{R}_\varphi^\zeta(k)$ into D_φ's loss [cf. Eq. (14)]. Despite being encouraged, $\|\nabla_{s,a}^t[D_\varphi]_t\|_F \leq k$ can nonetheless not be guaranteed solely from the application of the regularizer at (s_t, a_t). As such, to cover all bases, we will qualify a state-action pair (s_t, a_t) —equivalently, an action a_t in a given state s_t—as *"probably approximately* $\mathfrak{C}$-*valid"* if (s_t, a_t) is in the support of the distribution ζ that determines where the gradient penalty regularizer $\mathfrak{R}_\varphi^\zeta(k)$ of ℓ_φ^{GP} is applied in $\mathcal{S} \times \mathcal{A}$, i.e. if $(\operatorname{supp} \zeta) \ni (s_t, a_t)$. A probably approximately $\mathfrak{C}$-valid point is supported by the distribution that describes where $\|\nabla^t_{s,a}[D_\varphi]_t\|_F \leq k$ is enforced, and as such, $\mathfrak{R}^\zeta_\varphi(k)$ may be applied at this point.

Importantly, the policy might, due to its exploratory motivations, pick an action a_t in state s_t that is not $\mathfrak{C}$-valid. Depending on where the constraint will then be enforced, the sample might then be $\mathfrak{C}$-valid after r_φ's update (technically, indirectly via D_φ's update; cf. Sect. 6.3). This observation motivates the investigation we carry out in Sect. 6.4, in which we define a soft $\mathfrak{C}$ -validity pseudo-indicator of $\mathfrak{C}$ [cf. Eq. (67)] that enables us to assess whether the agent consistently performs approximately $\mathfrak{C}$-valid actions when it interacts with the MDP $\mathbb{M}^*$ following μ_θ.

6.3 A new reinforcement learning perspective on gradient penalty

We begin by considering a few variants of the original gradient penalty regularizer (Gulrajani et al. 2017) introduced in Sect. 5.4. Each variant corresponds to a particular case of the *generalized* version of the regularizer, described in Eq. (14). Subsuming all versions, we remind Eq. (14) here for didactic purposes:

$$\ell^{\text{GP}}_\varphi := \ell_\varphi + \lambda\,\mathfrak{R}^\zeta_\varphi(k) := \ell_\varphi + \lambda\,\mathbb{E}_{s_t\sim\rho^\zeta, a_t\sim\zeta}[(\|\nabla_{s_t,a_t} D_\varphi(s_t,a_t)\| - k)^2] \tag{66}$$

where ζ is the distribution that describes *where* the regularizer is applied—where the Lipschitz-continuity constraint is enforced in the input space $\mathcal{S}\times\mathcal{A}$. In Gulrajani et al. (2017), ζ corresponds to sampling point uniformly along segments joining samples generated by the agent following its policy and samples generated by the expert policy, i.e. samples from the expert demonstrations $\mathcal{D}$. Formally, focusing on the action only for legibility—the counterpart formalism for the state is derived easily by using the visitation distribution instead of the policy—$a\sim\zeta$ means $a = u\,a' + (1-u)\,a''$, where $a'\sim\pi_\theta$, $a''\sim\pi_e$, and $u\sim\text{unif}(0,1)$. The distribution ζ we have just described corresponds to the transposition of the GAN formulation to the GAIL setting, which is an *on*-policy setting. Therefore, in this work, we amend the ζ previously described, and replace it with its *off*-policy counterpart, where $a'\sim\beta$ (cf. Sect. 4). As for the penalty target, Gulrajani et al. (2017) use $k=1$, in line with the theoretical result derived by the authors. By contrast, DRAGAN (Kodali et al. 2017) use a ζ such that $a\sim\zeta$ means $a = a'' + \epsilon$, where $a''\sim\pi_e$, and $\epsilon\sim\mathcal{N}(0,10)$. Like WGAN-GP (Gulrajani et al. 2017), DRAGAN uses the penalty target $k=1$. Finally, for the sake of symmetry, we introduce a reversed version of DRAGAN, dubbed NAGARD (name reversed). To the best of our knowledge, the method has not been explored in the literature. NAGARD also uses $k=1$ as penalty target, but perturbs the policy-generated samples as opposed to the expert ones: $a\sim\zeta$ means $a = a' + \epsilon$, where $a'\sim\beta$ (*off*-policy setting), and $\epsilon\sim\mathcal{N}(0,10)$. We use $\lambda = 10$ in all the variants, in line with the original hyper-parameter settings in Gulrajani et al. (2017) and Kodali et al. (2017).

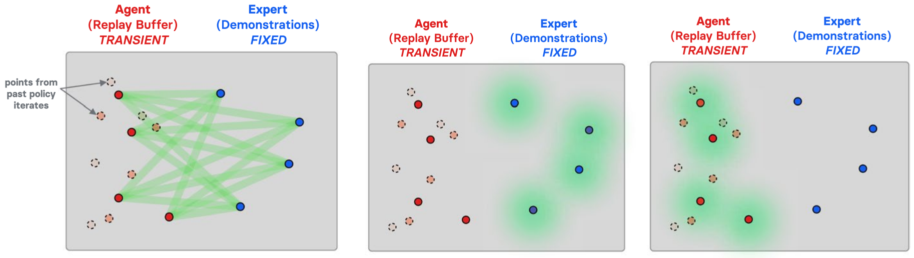
Figure 7 depicts in green the subspace of the input space $\mathcal{S}\times\mathcal{A}$ where the k-Lipschitz-continuity constraint, formalized as $\|\nabla^t_{s,a}[D_\varphi]_t\|_F \leq k$, and enouraged in ℓ^{GP}_φ by $\mathfrak{R}^\zeta_\varphi(k)$, is applied. In other words, Fig. 7 highlights the support of the distribution ζ for each variant, which have just been described above. As such, the green areas in Fig. 7b, c, and a are schematic depictions of where the state-actions pairs are *probably approximately* $\mathfrak{C}$*-valid*.

One conceptual difference between the DRAGAN penalty and the two others is that the support of the distribution ζ does not change throughout the entire training process for the former, while is does for the latter. Borrowing the intuitive terminology used in Kodali et al. (2017), WGAN-GP proposes a *coupled penalty*, while DRAGAN (like NAGARD) propose a *local* penalty. In Kodali et al. (2017), the authors perform a comprehensive empirical study of mode collapse, and diagnose that the generator collapsing to single modes is often coupled with the discriminator displaying sharp gradients around the samples from the real distribution. In model-free generative adversarial imitation learning, the generator does not have access to the gradient of the discriminator with respect to its actions in the backward pass, although it could be somewhat accessed using a model-based approach Baram et al. (2017). In spite of not being accessible *per se*, the sharpness of the discriminator's gradients near real samples observed in Kodali et al. (2017) translates, in the setting considered in this work, to sharp rewards, which we referred to as reward overfitting and was discussed thoroughly in Sect. 5.3. As such, mode collapse mitigation in the GAN setting translates to a problem of credit assignment in our setting, caused by the peaked reward landscape (cf. Appendix 7 to witness the sensitivity *w.r.t.* the discount factor γ, controlling how far ahead in the episode the agent looks). The stability issues the methods incur in either settings are on par. Both gradient penalty regularizers aim to address these stability weaknesses, and do so by enforcing a Lipschitz-continuity constraint, albeit on a different support $\operatorname{supp} \zeta$ (cf. Fig. 7).

As mentioned earlier in Sect. 5.4, the distribution ζ used in WGAN-GP (Gulrajani et al. 2017) is motivated by the fact that—as they show in their work—the *optimal* discriminator is 1-Lipschitz along lines joining real and fake samples. The authors of Kodali et al. (2017) deem the assumptions underlying this result to be unrealistic, which naturally weakens the ensuing method derived from this line of reasoning. They instead propose DRAGAN, whose justification is straightforward and unarguable: since they witness sharp discriminator gradients around real samples, they introduce a *local* penalty that aims to smooth out the gradients of the discriminator *around* the real data points. Formally, as described above when defining the distribution ζ associated with the approach, it tries to ensure Lipschitz-continuity of the discriminator in the neighborhoods (additive Gaussian noise perturbations) of the real samples. The generator or policy is more likely to escape the narrow peaks of the optimization landscape—corresponding to the real data points—with this extra stochasticity. *In fine*, in our setting, DRAGAN can dial down the sharpness of the reward landscape at expert samples the discriminator overfits on. This technique should therefore fully address the shortcomings raised and discussed in Sect. 5.4. While the method seem to yield better results than WGAN-GP in generative modeling with generative adversarial nets, the empirical results we report in Fig. 8 show otherwise. All the considered penalties help close the significant performance gap reported in Fig. 3, in almost every environment, but the penalty from WGAN-GP generally pulls ahead. Additionally, not only does is display higher empirical return, it also crucially exhibits more stable and less jittery behavior.

Despite the apparent disadvantage of *local* penalties (DRAGAN (Kodali et al. 2017) and NAGARD) compared to WGAN-GP in terms of their schematically-depicted $\operatorname{supp} \zeta$ sizes (cf. Fig. 7), it is important to remember that the additive Gaussian perturbation is distributed as $\mathcal{N}(0, 10)$. For these local methods, ζ is therefore covering a *large*[3] area around the central sample, including with high probability samples that are, according to the discriminator, from both categories—fake samples (predicted as from β), and real samples (predicted as from π_e). As such, the perceived diameter of the green disks in the schematic representations in Fig. 7b and c maybe smaller than it would be in reality. It is crucial to consider the coverage of the different ζ distributions as they determine how strongly the Lipschitz-continuity property is potentially enforced at a given state-action pair, for a fixed number of discriminator updates. Consequently, for a given optimization step, while the *local* penalties are—somewhat ironically—applying the Lipschitz-continuity constraint on data points *scattered* around the agent—(NAGARD) or

[3] Considering the observations are clipped to be in $[-5.0, 5.0]$, as is customary in the MuJoCo (Todorov et al. 2012) benchmark (Brockman et al. 2016), an additive Gaussian perturbation with $\sigma^2 = 10$ can, in all fairness, be qualified as *large*.

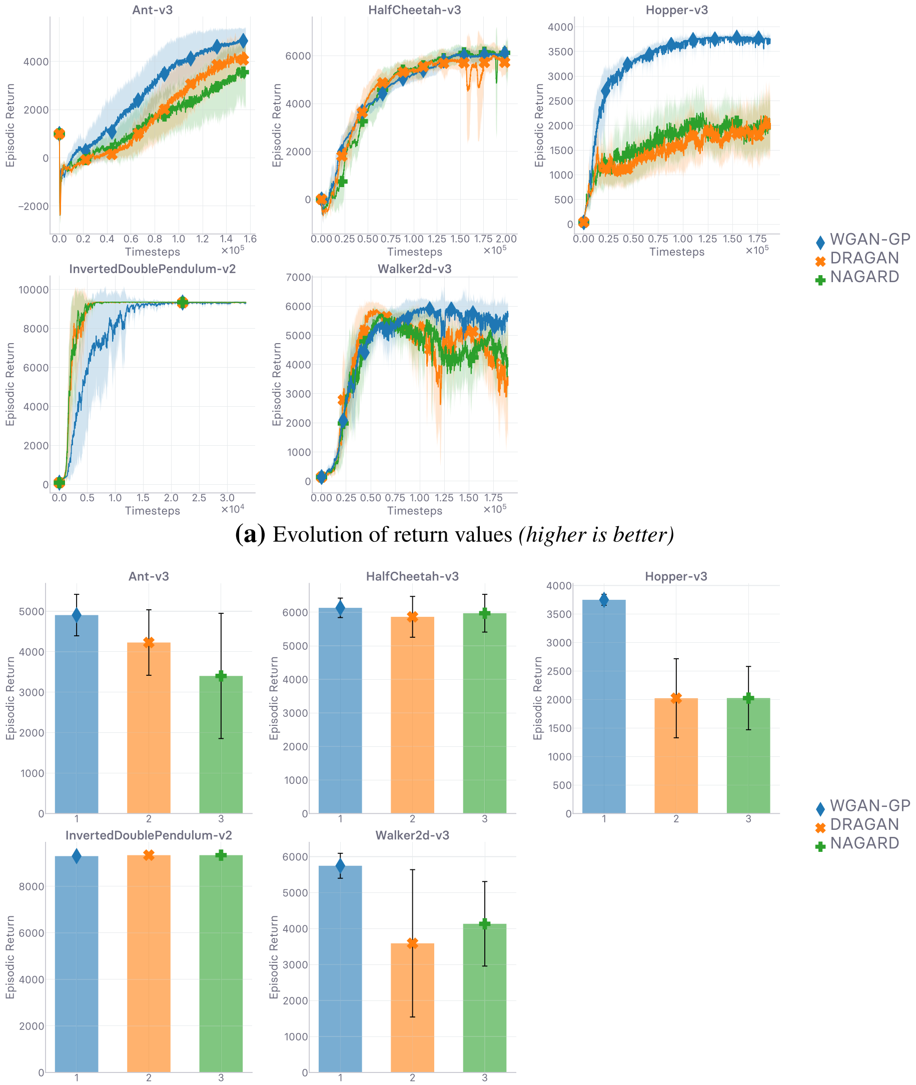


(a) WGAN-GP [49] **(b)** DRAGAN [74] **(c)** NAGARD (illustrative *only*)

Fig. 7 Schematic representation (in green) of the support of the ζ distribution, depicting *where* the gradient penalty regularizer is enforced, at a given iteration, and for all iterations throughout the lifetime of the learning agent. It corresponds to the subspace of $\mathcal{S} \times \mathcal{A}$ on which the Lipschitz-continuity constraint is applied: where the state-action pairs are *likely* $\mathfrak{C}$-valid. The intensity of the green color indicates the probability assigned by the distribution ζ on the state-action pair. The more opaque the coloration, the higher the probability. Best seen in color (Color figure online)

expert-generated (DRAGAN) samples, the supp ζ for WGAN-GP is less diffuse. Local penalties ensure the Lipschitzness is somewhat satisfied all around the selected samples, which for DRAGAN is motivated by the fact that there are narrow peaks on the reward landscape located at the expert samples, where it us prone to overfit (cf. Sect. 5.3). The distribution ζ used in WGAN-GP also supports data points near expert samples, but these are not scattered all around for the sole purpose of making the whole area smooth and escape bad basins of attraction like in DRAGAN. In other terms, the Lipschitz-continuity constraint is applied isotropically, from the original expert sample outwards. By contrast, WGAN-GP's ζ only supports a few discrete directions from a given expert sample, the lines joining said sample to all the agent-generated samples (of the mini-batch). Intuitively, while DRAGAN smooths out the reward landscape starting from expert data points and going in every direction from there, WGAN-GP smooths out the reward landscape starting from expert data points and going only in the directions that point toward agent-generated data points. As such, one could qualify DRAGAN as *isotropic* regularizer, and WGAN-GP as *directed* regularizer.

We believe that WGAN-GP outperforms DRAGAN in the setting and environments considered in this work (cf. Fig. 8) due to the fact that the agent benefits from having smooth reward *pathways* in the reward landscape in-between agent samples and expert samples. Along these pathways, going from the agent sample end to the expert sample end, the reward *progressively* increases. For the agent trying to maximize its return, these series of gradually increasing rewards joining agent to the expert data points are akin to an *automatic curriculum* (Karpathy and Van De Panne 2012; OpenAI 2019) assisting the reward-driven agent and leading it towards the expert. Figure 8 shows that WGAN-GP indeed achieves consistently better results across every environment but the least challenging, as seen in the `IDP` environment (cf. Table 1). In the four considerably more challenging environments, the *directed* method allows the agent to attain overall significantly higher empirical return than its competitors. Besides, it displays greater stability when approaching the asymptotic regime, whereas the *local* regularizers clearly suffer from instabilities, especially DRAGAN in the results obtained in environments `Walker2d` and `HalfCheetah`, depicted in Fig. 8. While the proposed interpretation laid out previously corroborates the results obtained and reported in Fig. 8, it does not explain the instability issues hindering the local penalties. We believe the jittery behavior observed in the results obtained in environments `Walker2d` and `HalfCheetah`



(a) Evolution of return values *(higher is better)*

(b) Final return values at timeout *(higher is better)*

Fig. 8 Evaluation of gradient penalty variants. Explanation in text. Runtime is 48 h

(cf. Fig. 8)—once the peak performance is attained—is caused by supp ζ (green areas in Fig. 7) not changing *is size* as the agent learns to imitate and gets closer to the expert in $\mathcal{S} \times \mathcal{A}$.

Indeed, in DRAGAN, ζ is a stationary distribution: it applies the regularizer on perturbations of the expert samples, where the additive noise's underlying sufficient statistics are constant throughout the learning process, and where the expert data points are distributed according to the stationary policy π_e and its associated state visitation distribution. For NAGARD, the perturbations follow the same distribution, and remain constant across the updates. However, unlike DRAGAN, ζ is defined by adding the stationary noise to samples from the *current* agent, every update, distributed as β in our *off*-policy setting. Since β is by construction non-stationary across the updates, as a mixture of past π_θ updates, ζ is non-stationary in NAGARD. Despite ζ's having these different support and stationary traits, the results of either local penalties are surprisingly similar. This is due to the variance of the additive noise used in both methods being large relative to the distance between the expert and agent samples, at all times, in the considered environments. As such, their supp ζ are virtually overlapping, which makes the two local penalties virtually equivalent, and explains the observed similarities in-between them.

Coming back to the main point—"*why do local penalties suffer from instabilities at the end of training?*"—even though the agent samples are close to the expert ones, the local methods both apply the same large perturbation before applying the Lipschitz-continuity penalty. The probability mass assigned by ζ is therefore still spread similarly over the input space, and is therefore severely decreased in-between agent and expert samples since these are getting closer in the space. The local methods are therefore often applying the constraint on data points that the policy will never visit again (since it wants to move towards the expert) and equivalently, rarely enforces the constraint between the agent and the expert, which is where the agent should be encouraged to go. With this depiction, it is clearer why WGAN-GP pulls ahead. Compared to the fixed size of supp ζ in the local penalties, ζ *adapts* to the current needs of the agent (hence qualifying as non-stationary). As the agent gets closer to the expert, Lipschitz-continuity is always enforced on data points between them, which is where it potentially benefits the agent most. The support of ζ is therefore decreasing in size as the iterations go by, focusing the probability mass of ζ where enforcing a smooth reward landscape matters most: where the agent should go, i.e. in the direction of the expert data points.

Besides, considering the inherent sample selection bias (Heckman 1979) the control agent is subjected to, where the latter end up in $\mathcal{S} \times \mathcal{A}$ depends on its actions, in every interaction with the dynamical system represented by its environment. This aspect dramatically differs from the traditional *non-Markovian* GAN setting—in which these penalties were introduced—where the generator's input noise is *i.i.d.*-sampled. Indeed, suffering from said sample selection bias, an imitation agent straying from the expert demonstrations is likely to keep on doing so until the episode is reset (cf. discussion in Sect. 5.4). Distributions ζ whose definition involve samples generated by the learning agent and adapt to the agent's current relative position *w.r.t.* the expert data points therefore provide valuable extra guidance in Markovian settings. Additionally, assuming the input also contained the *phase*—"*how far the agent/expert is in the current episode*", $0 \leq t \leq T$—[like in Peng et al. (2018)] not only would the imitation task be easier, but the benefits of the WGAN-GP penalty would be further enhanced, as it would allow the models to exploit the temporal structure of to the considered Markovian setting.

Finally, in reaction to the recent interest towards "*zero-centered*" gradient penalties (Roth et al. 2017; Mescheder et al. 2018), due to the theoretical convergence guarantees they allow for, we have conducted a grid search on the values of the Lipschitz constant k and the regularizer importance coefficient λ, as described in Sect. 6.3. The results are reported in Appendix 5.3. In short, the method performs poorly when $k = 0$, unless a very small value is used for λ. Enforcing 0-Lipschitzness is far too restraining for the agent to learning anything, unless this constraint is only loosely imposed. Conversely, a smaller λ value yields worse results when $k = 1$, revealing the interaction between the gradient penalty hyper-parameters k and λ. In particular, we will momentarily provide comprehensive evidence along with a greater characterization of how the choice of scaling factor λ not only impacts the agent's performance (which is already depicted in Appendix 5.3), but how it correlates quantitatively with the approximate $\mathfrak{C}$-validity displayed by the agent (cf. Sect. 6.4). Unless explicitly stated otherwise, we use the WGAN-GP penalty variant, with Lipschitz constant target $k = 1$, and scaling coefficient $\lambda = 10$ throughout the empirical results exhibited in both the body and appendix.

6.4 Diagnosing $\mathfrak{C}$-validity: Is the Lipschitzness premise of the theoretical guarantees satisfied in practice?

To put things in perspective, we first give a side-by-side rundown of how what we set out to tackle here compares to what we have just tackled in Sect. 6.3, thereby giving a glimpse of what we set out to investigate in what follows. In the previous section, we showed how (a) the choice of ζ (*where* do we want to encourage approximately $\mathfrak{C}$-valid behavior), and (b) the choice of λ (*to what degree* do we want to encourage approximately $\mathfrak{C}$-valid behavior) both independently impact the agent's performance in terms of empirical episodic return. In this section on the other hand, we will show how (a) the choice of ζ, and (b) the choice of λ both independently impact the agent's consistency at *effectively* selecting approximately $\mathfrak{C}$-valid actions with its learned policy μ_θ. If we were to find a strong positive correlation between the agent's asymptotic return and its effectively measured approximate $\mathfrak{C}$-validity rate—high when high, low when low, for all tested ζ's and for all tested λ 's—then we would have further quantitative evidence to support our work's main claim: reward Lipschitzness is necessary to achieve high return, and higher Lipschitzness uptime correlates strongly with higher return. Perhaps most crucially, we would be able to correlate high empirical episodic return with high chance of satisfying the premise of our theoretical guarantees (r_φ's Lipschitzness). As such, these would consequently apply in in practice too. This would attest to the practical relevance of Sect. 6.1.

We have shown that enforcing a Lipschitz-continuity constraint on the learned reward r_φ (albeit indirectly via D_φ) is instrumental in achieving expert-level performance in off-policy generative adversarial imitation learning (cf. Sect. 5.5). We have also shown that directed regularization techniques yield better results, seemingly due to the better guidance they provide to the mimicking agent, in the form of an automatic curriculum of rewards towards the expert data points (cf. Sect. 6.3). Such curriculum only exists where the Lipschitz-continuity constraint is satisfied. Said differently, it could not exist if the constraint were not satisfied along μ_θ's pathways which would then involve non-smooth hurdles. It is therefore crucially important for said constraint to be satisfied *in effect* for the state-actions pairs in the the support of the policy the agent uses in its learning update, μ_θ, i.e. supp $\mu_\theta \ni (s_t, a_t)$. Still, the deterministic policy μ_θ likely performs only approximately $\mathfrak{C}$-valid actions as it is trained with the sole objective to maximize cumulative rewards that represent its similarity *w.r.t.* the expert π_e. The imitation rewards corresponding to a greater degree of similarity are, by design of the generative adversarial imitation learning framework, situated between the agent's current position and the expert's position on the current reward landscape. Since this is where we apply the Lipschitzness constraint (with WGAN-GP, our baseline, as said above)—equivalently, since these regions are approximately $\mathfrak{C}$ -valid—μ_θ is likely to never select $\mathfrak{C}$-invalid actions as it optimizes for its utility function (cf. Sect. 3). Conversely, in the considered setting, picking $\mathfrak{C}$-invalid actions could in theory hinder the optimization process the policy is subject to, as μ_θ would *a priori* venture in regions of the state-action space that do not increase its similarity with the expert policy π_e —or, at the very least, for which the *non*-satisfaction of the reward's Lipschitz-continuity premise $\|\nabla^t_{s,a}[r_\varphi]_t\|_F \leq \delta$ might lead to instabilities due to $\|\nabla^t_{s,a}[Q_\varphi]_t\|_F > \Delta_\infty$ as a direct consequence of our theoretical guarantees (cf. Sect. 6.2). Since we do not have such a tight control over where and to what degree the Lipschitzness constraint over the reward r_φ is *satisfied* (hence our introduction of the notions of approximately $\mathfrak{C}$-valid samples and probably approximately $\mathfrak{C}$-valid samples), we instead turn to the closest surrogate over which we do have a tighter control: where and to what degree D_φ's constraint is *enforced*. The *"where"* is controlled by the choice of ζ (determined by the gradient penalty regularization method in use), and the *'to what degree'* by the choice of λ scale.

Still, even in the occurrence where D_φ's constraint is enforced by adding $\mathfrak{R}^\zeta_\varphi(k)$ as in ℓ^{GP}_φ [cf. Eq. (14)] at the point (s_t, a_t), the most we could say is that (s_t, a_t) is probably approximately $\mathfrak{C}$-valid, since $(s_t, a_t) \in \operatorname{supp} \zeta$—otherwise, the gradient penalty regularizer $\mathfrak{R}^\zeta_\varphi(k)$ could never have been applied at that point in the landscape $\mathcal{S} \times \mathcal{A}$. In effect, enforcing the constraint at the point was enough to guarantee that $\|\nabla^t_{s,a}[D_\varphi]_t\|_F \leq k$, and we therefore do not know whether (s_t, a_t) is approximately $\mathfrak{C}$-valid, or not. As a direct consequence, we can *a fortiori* not guarantee that $\|\nabla^t_{s,a}[r_\varphi]_t\|_F \leq \delta$; we do not know whether (s_t, a_t) is $\mathfrak{C}$-valid, or not—cf. Sect. 6.2.5 for our discussion on indirect reward regularization, in which we establish that D_φ's k-Lipschitzness causes r_φ to be δ-Lipschitz in practice. On the flip side, based on the latter result about indirect Lipschitz-continuity inducement, we can state that ensuring empirically that $\|\nabla^t_{s,a}[D_\varphi]_t\|_F \leq k$ is *enough* to ensure that $\|\nabla^t_{s,a}[r_\varphi]_t\|_F \leq \delta$ is verified in practice. In other words, showing that (s_t, a_t) is approximately $\mathfrak{C}$-valid can be used as a proxy for showing that (s_t, a_t) is $\mathfrak{C}$-valid, empirically. As such, in order to assess whether the premise of the theoretical guarantees we derived in Sect. 6.1 is satisfied in practice (r_φ's -δ-Lipschitz-continuity), it is sufficient to assess whether the agent's actions $a_t = \mu_\theta(s_t)$ are approximately $\mathfrak{C}$-valid. In particular, we want to know the relative impacts the choices of ζ and the λ in ℓ^{GP}_φ have on the propensity for an action from μ_θ to be approximately $\mathfrak{C}$-valid. So as to estimate how often the actions selected by the agent via μ_θ are approximately $\mathfrak{C}$-valid, we build an estimator that *softy* approximates $\mathbb{1}_\mathfrak{C} : \mathcal{S} \times \mathcal{A} \to \{0, 1\}$, the indicator of the $\mathfrak{C}$-validity subspace over $\mathcal{S} \times \mathcal{A}$, where $\mathbb{1}_\mathfrak{C}(s_t, a_t) = 1$ when $(s_t, a_t) \in \mathfrak{C}$, and $\mathbb{1}_\mathfrak{C}(s_t, a_t) = 0$ when $(s_t, a_t) \notin \mathfrak{C}$. Accordingly, we call our estimator *soft approximate* $\mathfrak{C}$ *-validity pseudo-indicator*, implementing a soft, C^0 mapping $\widehat{\mathbb{1}}_\mathfrak{C} : \mathcal{S} \times \mathcal{A} \to (0, 1]$, and formally defined as, $\forall t \in [0, T] \cap \mathbb{N}, \forall (s_t, a_t) \in \mathcal{S} \times \mathcal{A}$:

$$\begin{gathered}\widehat{\mathbb{1}}_\mathfrak{C}(s_t, a_t) := \exp\Big(-\max\big(0, \|\nabla_{s_t,a_t} D_\varphi(s_t, a_t)\| - k\big)^2\Big)\\ \blacktriangleright \textit{soft approximate } \mathfrak{C}\textit{-validity pseudo-indicator}\end{gathered} \tag{67}$$

Thus, for a given pair (s_t, a_t), $\widehat{\mathbb{1}}_\mathfrak{C}(s_t, a_t) = 1$ when $\|\nabla^t_{s,a}[D_\varphi]_t\|_F \leq k$ and $\widehat{\mathbb{1}}_\mathfrak{C}(s_t, a_t) \to 0$ when $\|\nabla^t_{s,a}[D_\varphi]_t\|_F \gg k$.

Figures 9 and 10 depict respectively the evolution of the values taken by the soft approximate $\mathfrak{C}$-validity pseudo-indicator $\widehat{\mathbb{1}}_\mathfrak{C}$ [cf. Eq. (67)] for different choices of ζ (different gradient penalty variants) and λ (sweep over $\mathfrak{R}^\zeta_\varphi(k)$'s scaling factor). In Figs. 9 and 10, we also share the return accumulated by the agents throughout their respective training periods, (cf. Figs. 9a and 10a, respectively). In particular, what we report in Figs. 9a and 10a echoes what we have already reported in Figs. 8 and 16, but the settings in which the agents were trained differ (ever so) slightly. We indicate the specificities of the setting tackled in this section below, in this very paragraph. Still, since their settings do not match perfectly, we report their return along their soft approximate $\mathfrak{C}$-validity pseudo-indicator $\widehat{\mathbb{1}}_\mathfrak{C}$ values. We monitor and record these values during the evaluation trials the agent periodically goes through, in which the agent uses μ_θ to decide what to do in a given state. To best align with the definition of Lipschitz-continuity (cf. Definition 1), which is also how we

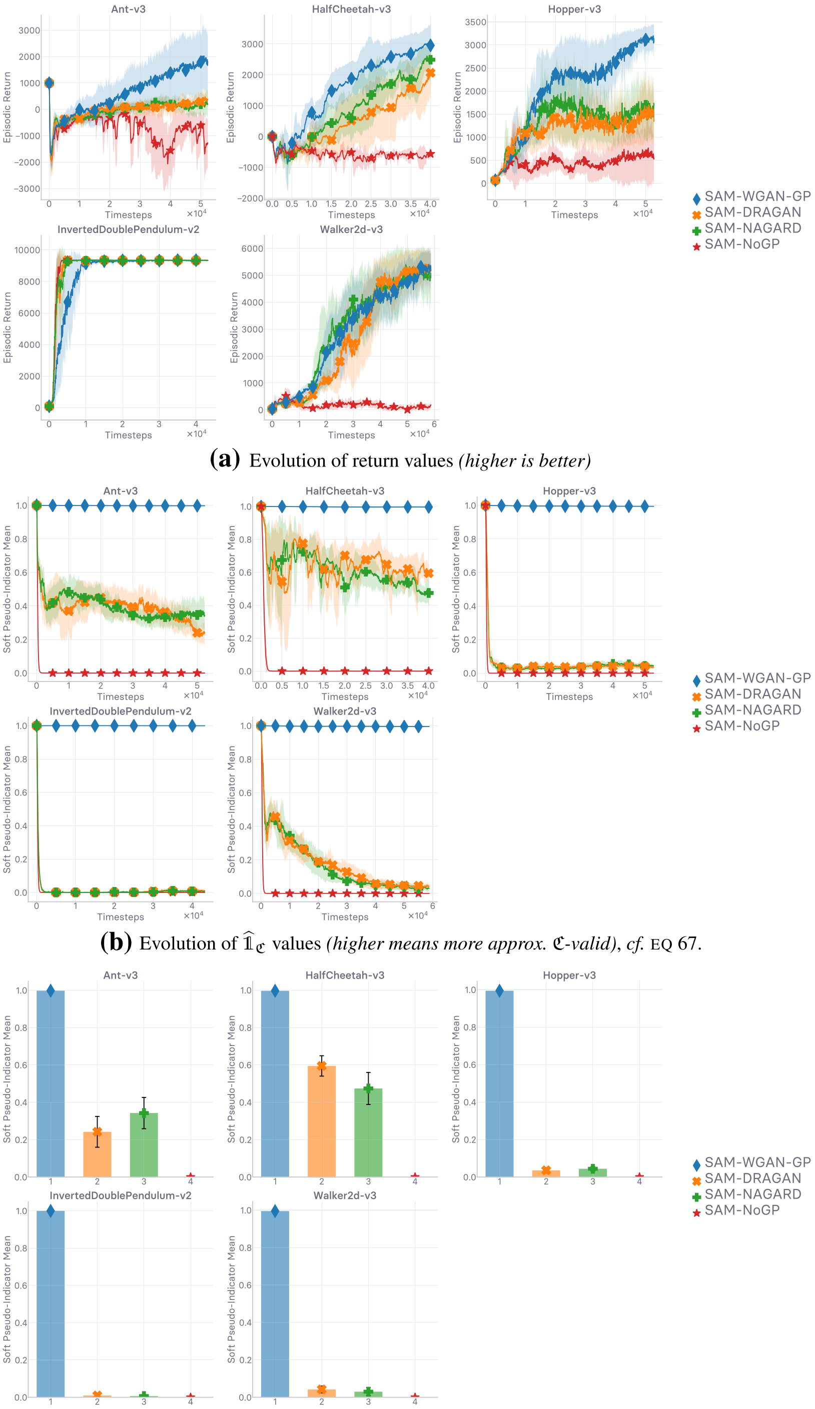
Fig. 9 Evaluation of several GP methods differing by their ζ distribution In line with how we defined it in Eq. (14), ζ controls *"where"* the GP constraint is enforced. Also, we report what happens without any GP regularization (`NoGP`). Explanation in text. Runtime is 48h ▶

designed our soft approximate $\mathfrak{C}$-validity pseudo-indicator $\widehat{\mathbb{1}}_{\mathfrak{C}}$, we use one-sided gradient penalties $\mathfrak{R}^{\zeta}_{\varphi}(k)$ in the λ sweep—$\max(0, \|\nabla_{s_t,a_t} D_\varphi(s_t, a_t)\| - k)^2$, which *purely* encourages $\|\nabla^t_{s,a}[D_\varphi]_t\|_F \leq k$ to be satisfied (nothing more, nothing less)—although we have shown the variant presents very little empirical difference with the base two-sided one (cf. ablation in Appendix 5.1). It is worth noting that the experiments whose results are reported in Figs. 9 and 10 carry out less iterations during the fixed allowed runtime, due to the substantial cost entailed by computing soft approximate $\mathfrak{C}$-validity pseudo-indicator $\widehat{\mathbb{1}}_{\mathfrak{C}}$ at every single evaluation step, in every evaluation trial. One could cut down that cost simply by evaluating $\widehat{\mathbb{1}}_{\mathfrak{C}}$ less frequently, but we decided otherwise, as we gave priority to having a finer tracking of $\widehat{\mathbb{1}}_{\mathfrak{C}}$. Besides, despite this slight apparent hindrance, the values of the proposed pseudo-indicator reported in either figure seem to have reached maturity, nearing their asymptotic regime, in the allowed runtime. We now go over and interpret the results reported in both figures.

In Fig. 9, we observe that the monitored soft approximate $\mathfrak{C}$-validity pseudo-indicator $\widehat{\mathbb{1}}_{\mathfrak{C}}$ [cf. Eq. (67)] consistently takes values close to 1 when using the distribution ζ advocated in WGAN-GP to assemble the regularizer $\mathfrak{R}^{\zeta}_{\varphi}(k)$. Conversely, *not* using any gradient penalty regularizer causes the approximate $\mathfrak{C}$-validity rate to be in the vicinity of 0. Albeit *a priori* not surprising, it is still substantially valuable to notice that D_φ's k-Lipschitz-continuity (and therefore r_φ's δ-Lipschitz-continuity; cf. Sect. 6.2) *never* happens by accident (or rather, by chance). As for DRAGAN and NAGARD (both being non-directed gradient penalty schemes, unlike WGAN-GP; cf. Sect. 6.3), both perform similarly across the board in terms of collected $\widehat{\mathbb{1}}_{\mathfrak{C}}$ values. Their recorded soft pseudo-indicator values stay around a fixed value per environment, different for every one of them. These are within the [0.1, 0.7] range, and as such, are definitely encouraging $\|\nabla^t_{s,a}[D_\varphi]_t\|_F \leq k$ in practice, yet are falling short of achieving the same *(a)* effective approximate $\mathfrak{C}$-validity value, and *(b)* effective approximate $\mathfrak{C}$-validity consistency as WGAN-GP. These phenomenona occur consistently across the spectrum of tackled environments.

In Fig. 10, we observe the unsurprising fact that the higher λ's value is—equivalently, the more we encourage the regularity property $\|\nabla^t_{s,a}[D_\varphi]_t\|_F \leq k$ to be satisfied—the more $\|\nabla^t_{s,a}[D_\varphi]_t\|_F \leq k$ is satisfied in effect. Besides confirming that gradient penalization indeed urges Lipschitzness (which we were not doubting), the figure helps us gauge to what degree the value of $\mathfrak{R}^{\zeta}_{\varphi}(k)$'s scaling coefficient in $\ell^{\mathrm{GP}}_{\varphi}$ [cf. Eq. (14)] affects quantitatively the satisfaction of $\|\nabla^t_{s,a}[D_\varphi]_t\|_F \leq k$ monitored via the soft proxy $\widehat{\mathbb{1}}_{\mathfrak{C}}$. We considered powers of 10 for λ's sweep, tackling the values $\lambda_i := 10^i$, for $i \in \{-3, -2, -1, 0, 1\}$. The gap inbetween the $\widehat{\mathbb{1}}_{\mathfrak{C}}$ values associated with each of these λ_i differ per environment, but their ranking remain the same (higher $\widehat{\mathbb{1}}_{\mathfrak{C}}$'s for higher i's). At its lowest (i.e. for minimum i: $i = -3$) the soft pseudo-indicator values lie more often that not near 0. For $i = 1$, $\widehat{\mathbb{1}}_{\mathfrak{C}}$ perfectly aligns on the 1 value, meaning that the value we used so far ($\lambda = 10$, which corresponds to λ_i with $i = 1$) is enough for μ_θ to achieve a 100% satisfaction rate of $\|\nabla^t_{s,a}[D_\varphi]_t\|_F \leq k$. The case $i = 0$ is right on the edge: in some environments, the approximate $\mathfrak{C}$-validity exactly equals 1, while for other environments, it nears it, yet does not quite reach it.

Since we use WGAN-GP's ζ in the experiments reported in Fig. 10, we can first conclude that picking WGAN-GP's ζ variant and $\lambda = 10$ not only yields the best



(a) Evolution of return values *(higher is better)*

(b) Evolution of $\widehat{\mathbb{1}}_{\mathfrak{C}}$ values *(higher means more approx. $\mathfrak{C}$-valid)*, *cf.* EQ 67.

(c) Final $\widehat{\mathbb{1}}_{\mathfrak{C}}$ values at timeout *(higher means more approx. $\mathfrak{C}$-valid)*, *cf.* EQ 67.

Fig. 10 Evaluation of several GP methods differing by their λ scaling factor In line with how we defined it ▶ in Eq. (14), ζ controls *"to what degree"* the GP constraint is enforced. Also, we report what happens without any GP regularization (NoGP). Explanation in text. Runtime is 48 h

empirical return (as reported and discussed in Sect. 6.3), but also guarantees that the constraint $\|\nabla^t_{s,a}[D_\varphi]_t\|_F \leq k$ (and therefore $\|\nabla^t_{s,a}[r_\varphi]_t\|_F \leq \delta$; cf. Sect. 6.2) is satisfied for 100% of the actions performed by the agent's μ_θ in practice. As such, we can conclude that, in practice, the main premise of the theoretical guarantees we have derived in Sect. 6.1—the reward δ-Lipschitz-continuity, $\|\nabla^t_{s,a}[r_\varphi]_t\|_F \leq \delta$—is satisfied, hence making our theoretical guarantees *practically* relevant and insightful. In addition, since we showed that the learning agent's policy μ_θ (or rather, it's companion Q-value) is trained on a reward surrogate r_φ that verifies $\|\nabla^t_{s,a}[r_\varphi]_t\|_F \leq \delta$ almost 100% of the time, we have empirically proved that the agent effectively sees virtually uninterrupted sequences of smooth rewards. This new observation somewhat corroborates our RL-grounded interpretation of directed gradient penalization as as the automated and adaptive creation of reward curricula (cf. Sect. 6.3, and particularly our schematic depiction of WGAN-GP's supp ζ in Fig. 7a).

Despite having answered the question we asked in the title of the section (in the block right above), interpreting the findings laid out both in this section and in the previous one side-by-side allows us to draw another critical conclusion, substantially more meaningful than if we were to interpret either in a vacuum. In Sect. 6.3, we studied the impact ζ and λ both have on the agent's performance, in terms of the empirical return in the MDP $\mathbb{M}$. We refer here to the latter via the shorthand Return. In *this* section, on the other hand, we have studied the impact ζ and λ both have on the effective approximate $\mathfrak{C}$-validity rate of the agent. We refer here to the latter via the shorthand Validity. What emerges from comparing these two sets of results is that, for every given pair (ζ, λ) (*where* to apply the gradient penalty, and *to what degree*, respectively) in ℓ^{GP}_φ [cf. Eq. (14)]: low Return co-occurs with low Validity; intermediate Return co-occurs with intermediate Validity; high Return co-occurs with high Validity. Said differently, Return and Validity behave similarly under the various pairings (ζ, λ) that we have considered. Through these observations, we therefore witness a strong correlation between Return and Validity. Ultimately, by combining our two previous empirical analyses, we have shown that Validity is a good predictor or Return, and *vice versa*.

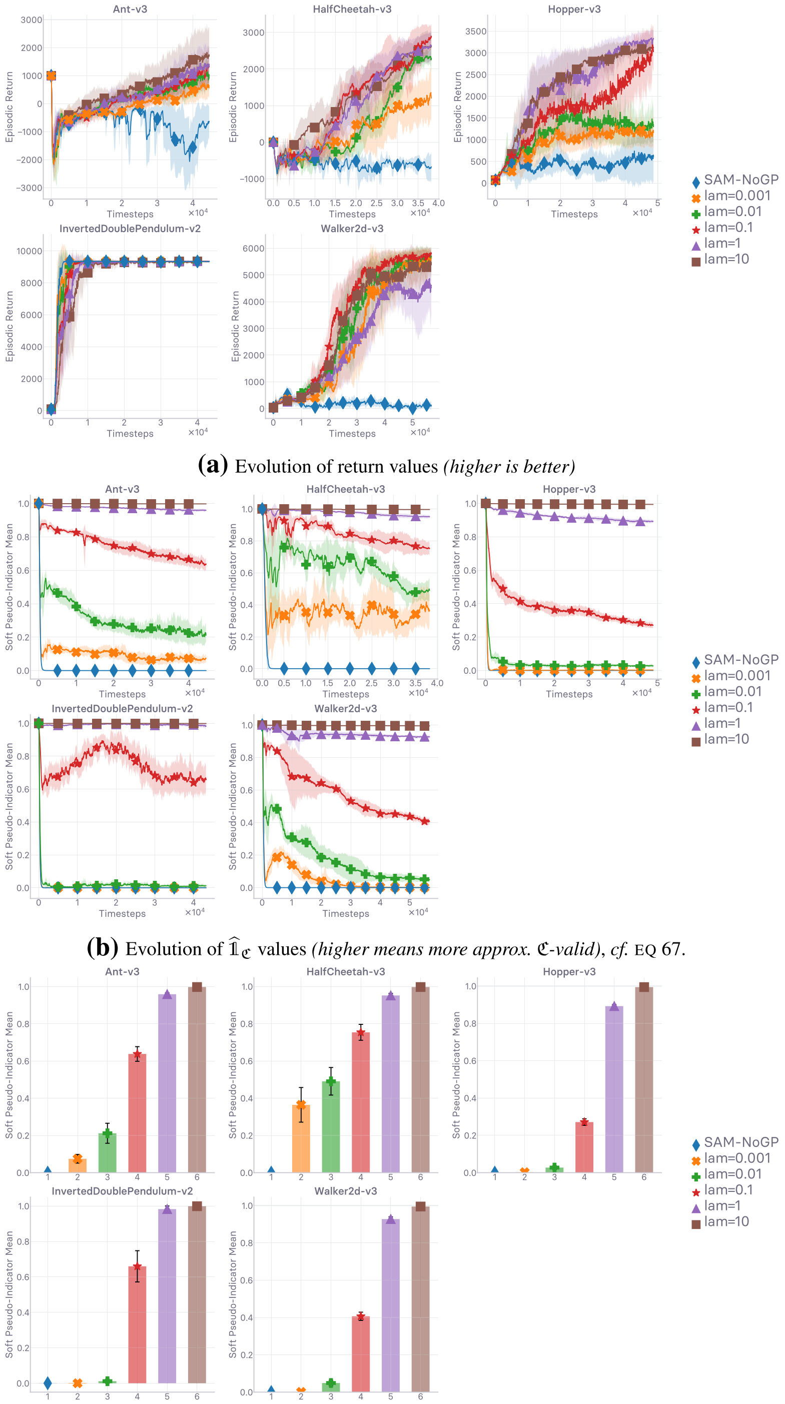
In fine, compared to Sects. 5.5, 6.4 (this section) gives a far more fine-grained diagnostic of how reward Lipschitzness relates to empirical return, along with insights related to the practicality of our theorerical guarantees.

6.5 Towards fulfilling the premise: a provably more robust way to further encourage Lipschitzness

We introduce two new entities, κ_t and $\tilde{r}_\varphi : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$, formally defined as:

$$\tilde{r}_\varphi(s_t, a_t) \vcentcolon= \kappa_t \, r_\varphi(s_t, a_t) \qquad \blacktriangleright \kappa_t - \textit{preconditioned reward } \tilde{r}_\varphi \tag{68}$$

$\forall t \in [0, T] \cap \mathbb{N}, \forall (s_t, a_t) \in \mathcal{S} \times \mathcal{A}$, where $0 < \kappa_t \leq 1, \forall t \in [0, T] \cap \mathbb{N}$ (in any episode).



(a) Evolution of return values *(higher is better)*

(b) Evolution of $\widehat{\mathbb{1}}_{\mathfrak{C}}$ values *(higher means more approx. $\mathfrak{C}$-valid)*, *cf.* EQ 67.

(c) Final $\widehat{\mathbb{1}}_{\mathfrak{C}}$ values at timeout *(higher means more approx. $\mathfrak{C}$-valid)*, *cf.* EQ 67.

We call κ_t a *reward preconditioner* since it functionally echoes the numerical transformation that conditions the tackled problem into a form that is more amenable to be solved via first-order optimization methods. Since our preconditioner is a scalar, we use the shorthand κ_t to constrast with the usual preconditioning matricies, denoted with capitalization. We have the following ranking of values, depending on the sign of the original learned synthetic reward r_φ: $\forall t \in [0, T] \cap \mathbb{N}$ and $\forall (s_t, a_t) \in \mathcal{S} \times \mathcal{A}$, we have $\tilde{r}_\varphi(s_t, a_t) \leq r_\varphi(s_t, a_t)$ whenever $r_\varphi(s_t, a_t) > 0$, and conversely, we have $\tilde{r}_\varphi(s_t, a_t) > r_\varphi(s_t, a_t)$ whenever $r_\varphi(s_t, a_t) < 0$.

We posit that κ_t does not depend on (i.e., is constant *w.r.t.*) the current state s_t and action a_t:

$$\frac{\mathrm{d}\kappa_t}{\mathrm{d}s_t} = 0 \qquad \text{and} \qquad \frac{\mathrm{d}\kappa_t}{\mathrm{d}a_t} = 0 \qquad \blacktriangleright\textit{property 1} \tag{69}$$

$\forall t \in [0, T] \cap \mathbb{N}, \forall (s_t, a_t) \in \mathcal{S} \times \mathcal{A}$. Thus, we can write $\frac{\mathrm{d}\tilde{r}_\varphi(s_t,a_t)}{\mathrm{d}s_t} = \kappa_t \frac{\mathrm{d}r_\varphi(s_t,a_t)}{\mathrm{d}s_t} + \frac{\mathrm{d}\kappa_t(s_t,a_t)}{\mathrm{d}s_t} r_\varphi = \kappa_t \frac{\mathrm{d}r_\varphi(s_t,a_t)}{\mathrm{d}s_t}$, and similarly $\frac{\mathrm{d}\tilde{r}_\varphi(s_t,a_t)}{\mathrm{d}a_t} = \kappa_t \frac{\mathrm{d}r_\varphi(s_t,a_t)}{\mathrm{d}a_t}$. As such, we have $\|\nabla^t_{s,a}[\tilde{r}_\varphi]_t\|_F = \kappa_t \|\nabla^t_{s,a}[r_\varphi]_t\|_F$, hence $\|\nabla^t_{s,a}[\tilde{r}_\varphi]_t\|_F \leq \|\nabla^t_{s,a}[r_\varphi]_t\|_F$ since $0 < \kappa_t \leq 1$, $\forall t \in [0, T] \cap \mathbb{N}$. Applying such a preconditioner to r_φ therefore squashes the absolute value of r_φ and in effect *shrinks* r_φ's Lipschitz constant (assuming here that r_φ is δ-Lipschitz, with $\|\nabla^t_{s,a}[r_\varphi]_t\|_F \leq \delta < +\infty$) without regard to the sign of the signal. Formally, since κ_t is posited constant in s_t and a_t, we have, $\forall t \in [0, T] \cap \mathbb{N}$ and $\forall (s_t, a_t) \in \mathcal{S} \times \mathcal{A}$:

$$\|\nabla^t_{s,a}[r_\varphi]_t\|_F \leq \delta \qquad \Longrightarrow \qquad \|\nabla^t_{s,a}[\tilde{r}_\varphi]_t\|_F = \kappa_t \|\nabla^t_{s,a}[r_\varphi]_t\|_F \leq \kappa_t \, \delta \quad (\leq \delta) \tag{70}$$

That is, if r_φ is δ-Lipschitz-continuous at t, then $\tilde{r}_\varphi$ is $\kappa_t\delta$-Lipschitz-continuous at t. Importantly, Eq. (70) will be instrumental in proving the first stages of our next theoretical guarantees, in which we deal with the counterpart action-value of $\tilde{r}_\varphi$, denoted by $\widetilde{Q}_\varphi$.

Because of its *"reward-squashing"* effect, we name the method corresponding to the subtitution of r_φ with the preconditioned reward $\tilde{r}_\varphi$ *"Pessimistic" Reward Preconditioning Enforcing Lipschitzness*. We dub the plug-in technique *"PURPLE"* (it is an acronym, with minor vowel filling and letter shuffle for legibility and easy of pronunciation). From this point onward, we study the effect of plugging PURPLE into SAM. The pseudo-code of the resulting algorithm can be obtained by replacing the learned reward r_φ in SAM's pseudo-code laid out in Algorithm 1 with the preconditioned reward $\tilde{r}_\varphi$.

We now study how the injection of PURPLE in SAM impacts the theoretical guarantees we have previously derived in Sect. 6.1. Concretely, we derive the PURPLE counterparts of Lemma 1, Theorems 1, 2, and Corollary 1. In order for us to characterize the Lipschitzness of $\widetilde{Q}_\varphi$, we also posit that the introduced preconditioner does not depend on (i.e., is constant *w.r.t.*) the *previously visited* (past) states and actions. Formally:

$$\frac{\mathrm{d}\kappa_{t+k+1}}{\mathrm{d}s_t} = 0 \quad \text{and} \quad \frac{\mathrm{d}\kappa_{t+k+1}}{\mathrm{d}a_t} = 0 \qquad \blacktriangleright\textit{property 2} \tag{71}$$

$\forall t \in [0, T] \cap \mathbb{N}, \forall k \in [0, T - t - 1] \cap \mathbb{N}, \forall (s_t, a_t) \in \mathcal{S} \times \mathcal{A}$. All in all, to develop the counterpart guarantees that will follow, the preconditioner κ_t must possess the following properties:

$$\frac{\mathrm{d}\kappa_t}{\mathrm{d}s_t} = 0 \quad \text{and} \quad \frac{\mathrm{d}\kappa_t}{\mathrm{d}a_t} = 0 \qquad \blacktriangleright \textit{property 1}, \text{Eq.}$$

$$\blacktriangleright \textit{gave us}\,\text{Eq. (70)}, \textit{ itself used in the proof (step 1) of}\text{Theorem } 3\textit{(a)+(b)}$$

$$\frac{\mathrm{d}\kappa_{t+k+1}}{\mathrm{d}s_t} = 0 \quad \text{and} \quad \frac{\mathrm{d}\kappa_{t+k+1}}{\mathrm{d}a_t} = 0 \qquad \blacktriangleright \textit{property } 2,\text{Eq. (71)}$$

$$\blacktriangleright \textit{used in the proof of Lemma } 2, \textit{ itself then used to prove (step 2)Theorem } 3\text{(a)} + \text{(b)}$$

$\forall t \in [0, T] \cap \mathbb{N}, \forall k \in [0, T-t-1] \cap \mathbb{N}, \forall (s_t, a_t) \in \mathcal{S} \times \mathcal{A}$. Note, the last two properties, Eqs. (69) and (71), can be condensed into, $\forall t \in [0, T] \cap \mathbb{N}, \forall k \in [0, T-t] \cap \mathbb{N}, \forall (s_t, a_t) \in \mathcal{S} \times \mathcal{A}$:

$$\frac{\mathrm{d}\kappa_{t+k}}{\mathrm{d}s_t} = 0 \qquad \text{and} \qquad \frac{\mathrm{d}\kappa_{t+k}}{\mathrm{d}a_t} = 0 \qquad \blacktriangleright \textit{property 1+2 condensed into one} \tag{72}$$

Property that κ_t must have In plain English, to get our guarantees, we need the preconditioner to not depend on neither current nor past states visited and actions taken by the agent. Note, the property $\kappa_t \leq 1$ is only ever used in Sect. 6.6.1, and will not be leveraged anywhere else. The developed theory will still hold if $\exists t \in [0, T] \cap \mathbb{N}$ such that $\kappa_t > 1$.

PURPLE in the broader algorithmic landscape Setting aside the fact that κ_t depends on a schedule indexed by the timestep t, PURPLE has the effect of reducing the (policy) gradients received by the GAIL or SAM policy, since it squashed the reward received by the agent. This scales down the gradients traditionally designed for the policy. The most direct adaptation of PURPLE to the GAN world would consist in scaling down the output of the discriminator (from which the reward is directly crafted in GAIL and SAM). The generator in a GAN is updated with gradients of the output of the discriminator *w.r.t.* its own parameters, similarly to how the actor is updated with gradients of the critic in an actor-critic. Consequently, squashing the output of the discriminator squashes the gradients used by the generator, which is equivalent to reducing the learning rate for the optimization of the generator (assuming no exotic optimizer or regularizer are in use).

Lemma 2 *Let the MDP with which the agent interacts be deterministic, with the dynamics of the environment determined by the function $f : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$. The agent follows a deterministic policy $\mu : \mathcal{S} \to \mathcal{A}$ to map states to actions, and receives rewards from $r_\varphi : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ upon interaction. The functions f, μ and r_φ need be C^0 and differentiable over their respective input spaces. This property is satisfied by the usual neural network function approximators. The "almost-everywhere" case can be derived from this lemma without major changes (relevant when at least one activation function is only differentiable almost-everywhere, ReLU).* (*a*) *Under the previous assumptions, for $k \in [0, T-t-1] \cap \mathbb{N}$ the following* (*non-recursive*) *inequality is verified:*

$$\| \nabla^t_{s,a} [\tilde{r}_\varphi]_{t+k+1} \|^2_F \leq \kappa^2_{t+k+1} C_t \, \| \nabla^{t+1}_{s,a} [r_\varphi]_{t+k+1} \|^2_F \tag{73}$$

where $0 < \kappa_u \leq 1 \; \forall u \in [0, T] \cap \mathbb{N}$, and $C_t := A_t^2 \max(1, B^2_{t+1})$, A_t and B_t being defined as the supremum norms associated with the Jacobians of f and μ respectively, with values in $\mathbb{R} \cup \{+\infty\}$:

$$\forall t \in [0, T] \cap \mathbb{N}, \quad \begin{cases} A_t := \| \nabla^t_{s,a} [f]_t \|_\infty = \sup \left\{ \| \nabla^t_{s,a} [f]_t \|_F \ : \ (s_t, a_t) \in \mathcal{S} \times \mathcal{A} \right\} \\ B_t := \| \nabla^t_s [\mu]_t \|_\infty = \sup \left\{ \| \nabla^t_s [\mu]_t \|_F \ : \ s_t \in \mathcal{S} \right\} \end{cases} \tag{74}$$

(*b*) *Additionally, by introducing time-independent upper bounds* $A, B \in \mathbb{R} \cup \{+\infty\}$ *such that* $\forall t \in [0, T] \cap \mathbb{N}$, $A_t \leq A$ *and* $B_t \leq B$, *and* κ *such that* $\kappa_u \leq \kappa \leq 1$ $\forall u \in [0, T] \cap \mathbb{N}$, *the non-recursive inequality becomes:*

$$\|\nabla^t_{s,a}[\tilde{r}_\varphi]_{t+k+1}\|_F^2 \leq \kappa^2 C \, \|\nabla^{t+1}_{s,a}[r_\varphi]_{t+k+1}\|_F^2 \tag{75}$$

where $C := A^2 \max(1, B^2)$ *is the time-independent counterpart of* C_t.

Proof of Lemma 2(a) (a) First, we take the derivative with respect to each variable separately:

$$\nabla^t_s[\tilde{r}_\varphi]_{t+k+1} = \frac{\mathrm{d}\tilde{r}_\varphi(s_{t+k+1}, a_{t+k+1})}{\mathrm{d}s_t} \tag{76}$$

$$= \kappa_{t+k+1} \frac{\mathrm{d}r_\varphi(s_{t+k+1}, a_{t+k+1})}{\mathrm{d}s_t} \qquad \blacktriangleright \textit{Eq. } (71)\textit{(property 2), left} \tag{77}$$

$$= \kappa_{t+k+1} \, \nabla^t_s[r_\varphi]_{t+k+1} \qquad \blacktriangleright \textit{repack} \tag{78}$$

$$\nabla^t_a[\tilde{r}_\varphi]_{t+k+1} = \frac{\mathrm{d}\tilde{r}_\varphi(s_{t+k+1}, a_{t+k+1})}{\mathrm{d}a_t} \tag{79}$$

$$= \kappa_{t+k+1} \frac{\mathrm{d}r_\varphi(s_{t+k+1}, a_{t+k+1})}{\mathrm{d}a_t} \qquad \blacktriangleright \textit{Eq. } (71)\textit{(property 2), right} \tag{80}$$

$$= \kappa_{t+k+1} \, \nabla^t_a[r_\varphi]_{t+k+1} \qquad \blacktriangleright \textit{repack} \tag{81}$$

By assembling the norm with respect to both input variables, we get:

$$\begin{aligned} &\|\nabla^t_{s,a}[\tilde{r}_\varphi]_{t+k+1}\|_F^2 \\ &\quad = \|\nabla^t_s[\tilde{r}_\varphi]_{t+k+1}\|_F^2 + \|\nabla^t_a[\tilde{r}_\varphi]_{t+k+1}\|_F^2 \end{aligned} \tag{82}$$

$$= \kappa^2_{t+k+1} \, \|\nabla^t_s[r_\varphi]_{t+k+1}\|_F^2 + \kappa^2_{t+k+1} \, \|\nabla^t_a[r_\varphi]_{t+k+1}\|_F^2 \tag{83}$$

$$= \kappa^2_{t+k+1} \left(\|\nabla^t_s[r_\varphi]_{t+k+1}\|_F^2 + \|\nabla^t_a[r_\varphi]_{t+k+1}\|_F^2 \right) \tag{84}$$

$$= \kappa^2_{t+k+1} \, \|\nabla^t_{s,a}[r_\varphi]_{t+k+1}\|_F^2 \qquad \blacktriangleright \textit{total norm} \tag{85}$$

As in Lemma 1, let A_t, B_t and C_t be time-dependent quantities defined as:

$$\forall t \in [0, T] \cap \mathbb{N}, \quad \begin{cases} A_t := \|\nabla^t_{s,a}[f]_t\|_\infty = \sup \left\{ \|\nabla^t_{s,a}[f]_t\|_F \; : \; (s_t, a_t) \in \mathcal{S} \times \mathcal{A} \right\} \\ B_t := \|\nabla^t_s[\mu]_t\|_\infty = \sup \left\{ \|\nabla^t_s[\mu]_t\|_F \; : \; s_t \in \mathcal{S} \right\} \\ C_t := A_t^2 \max(1, B_{t+1}^2) \end{cases} \tag{86}$$

Finally, by injecting Eq. (35), we directly obtain:

$$\|\nabla^t_{s,a}[\tilde{r}_\varphi]_{t+k+1}\|_F^2 = \kappa^2_{t+k+1}\,\|\nabla^t_{s,a}[r_\varphi]_{t+k+1}\|_F^2 \tag{87}$$

$$\leq \kappa^2_{t+k+1}\,A_t^2\max(1,B_{t+1}^2)\,\|\nabla^{t+1}_{s,a}[r_\varphi]_{t+k+1}\|_F^2 \qquad \blacktriangleright \mathit{Eq}\,(35) \tag{88}$$

$$= \kappa^2_{t+k+1}\,C_t\,\|\nabla^{t+1}_{s,a}[r_\varphi]_{t+k+1}\|_F^2 \qquad \blacktriangleright C_t \textit{ definition} \tag{89}$$

which concludes the proof of Lemma 2(a). □

Proof of Lemma 2(b) By introducing time-independent upper bounds A and B such that $A_t \leq A$ and $B_t \leq B$ $\forall t \in [0,T] \cap \mathbb{N}$, $C := A^2\max(1,B^2)$, and κ such that $\kappa_u \leq \kappa \leq 1$ $\forall u \in [0,T] \cap \mathbb{N}$, we obtain, through Eq. (88):

$$\|\nabla^t_{s,a}[\tilde{r}_\varphi]_{t+k+1}\|_F^2 \leq \kappa^2\,A^2\max(1,B^2)\,\|\nabla^{t+1}_{s,a}[r_\varphi]_{t+k+1}\|_F^2 \tag{90}$$

$$= \kappa^2\,C\,\|\nabla^{t+1}_{s,a}[r_\varphi]_{t+k+1}\|_F^2 \tag{91}$$

which concludes the proof of Lemma 2(b). □

Theorem 3 (Gap-dependent reward Lipschitzness) *In addition to the assumptions laid out in Lemma 2, we assume that the function r_φ is δ-Lipschitz over $\mathcal{S}\times\mathcal{A}$. Since r_φ is C^0 and differentiable over $\mathcal{S}\times\mathcal{A}$, this assumption can be written as $\|\nabla^u_{s,a}[r_\varphi]_u\|_F \leq \delta$, where $u \in [0,T] \cap \mathbb{N}$. (a) Then, under these assumptions, the following is verified:*

$$\|\nabla^t_{s,a}[\tilde{r}_\varphi]_{t+k}\|_F^2 \leq \kappa^2_{t+k}\,\delta^2\prod_{u=0}^{k-1} C_{t+u} \tag{92}$$

where $k \in [0,T] \cap \mathbb{N}$ and C_v is defined as in Lemma 2(a), $\forall v \in [0,T] \cap \mathbb{N}$. (b) Additionally, by involving the time-independent upper bounds introduced in Lemma 2(b), we have the following:

$$\|\nabla^t_{s,a}[\tilde{r}_\varphi]_{t+k}\|_F^2 \leq \kappa^2\,C^k\,\delta^2 \tag{93}$$

where $k \in [0,T] \cap \mathbb{N}$; C and κ are defined as in Lemma 2(b).

Proof of Theorem 3(a) We will prove Theorem 3(a) directly, not by induction (Lemma 2 proposes non-recursive inequalities, one side containing r_φ, the other $\tilde{r}_\varphi$). We want to prove the following Eq. (94), $\forall v \in [0,T] \cap \mathbb{N}$:

$$\|\nabla^t_{s,a}[\tilde{r}_\varphi]_{t+v}\|_F^2 \leq \kappa^2_{t+v}\,\delta^2\prod_{u=0}^{v-1} C_{t+u} \tag{94}$$

To do so, we will procede in two steps: (1) prove it for $v = 0$, and (2) prove it $\forall v \in [1,T] \cap \mathbb{N}$.

Step 1: case $v = 0$. When the gap $v = 0$, Eq. (94) becomes $\|\nabla^t_{s,a}[\tilde{r}_\varphi]_t\|_F^2 \leq \kappa_t^2\,\delta^2$, $\forall t \in [0,T] \cap \mathbb{N}$, which is verified by coupling Theorem 3's main assumption about the δ -Lipschitzness of r_φ and the observation laid out in Eq. (70).

Step 2: case $v \in [1, T] \cap \mathbb{N}$. We start from the result we derived in Lemma 2 (a), valid $\forall w \in [0, T-1] \cap \mathbb{N}$:

$$\|\nabla^t_{s,a}[\tilde{r}_\varphi]_{t+w+1}\|_F^2 \leq \kappa^2_{t+w+1}\, C_t\, \|\nabla^{t+1}_{s,a}[r_\varphi]_{t+w+1}\|_F^2 \qquad \blacktriangleright \textit{Lemma } 2\text{(a)} \tag{95}$$

$$\leq \kappa^2_{t+w+1}\, C_t\, \delta^2 \prod_{u=0}^{w-1} C_{t+1+u} \qquad \blacktriangleright \textit{Theorem } 1\text{(a)}, \textit{att} + 1 \tag{96}$$

$$= \kappa^2_{t+w+1}\, C_t\, \delta^2 \prod_{u=1}^{w} C_{t+u} \qquad \blacktriangleright \textit{index shift} \tag{97}$$

$$= \kappa^2_{t+w+1}\, \delta^2 \prod_{u=0}^{w} C_{t+u} \qquad \blacktriangleright \textit{repack product} \tag{98}$$

This shows that Eq. (94) is verified when $v = w + 1$, $\forall w \in [0, T-1] \cap \mathbb{N}$. Equation (94) is therefore valid $\forall v \in [1, T] \cap \mathbb{N}$.

Conclusion We have shown that Eq. (94) is valid $\forall v \in [0, T] \cap \mathbb{N}$, which concludes the proof of Theorem 3(a). □

Proof of Theorem 3(b) We will prove Theorem 3(b) directly, not by induction (Lemma 2 proposes non-recursive inequalities, one side containing r_φ, the other $\tilde{r}_\varphi$). We want to prove the following Eq. (99), $\forall v \in [0, T] \cap \mathbb{N}$:

$$\|\nabla^t_{s,a}[\tilde{r}_\varphi]_{t+v}\|_F^2 \leq \kappa^2\, C^v\, \delta^2 \tag{99}$$

where κ satisfies $\kappa_u \leq \kappa \leq 1\, \forall u \in [0, T] \cap \mathbb{N}$.

To do so, we will procede in two steps: (1) prove it for $v = 0$, and (2) prove it $\forall v \in [1, T] \cap \mathbb{N}$.

Step 1: case $v = 0$. When the gap $v = 0$, Eq. (99) becomes $\|\nabla^t_{s,a}[\tilde{r}_\varphi]_t\|_F^2 \leq \kappa_t^2\, \delta^2 \leq \kappa^2\, \delta^2$, $\forall t \in [0, T] \cap \mathbb{N}$, which is verified by coupling Theorem 3's main assumption about the δ -Lipschitzness of r_φ, the observation laid out in Eq. (70), and finally the definition of κ (upper bound for all the κ_u's).

Step 2: case $v \in [1, T] \cap \mathbb{N}$. We start from the result we derived in Lemma 2(b), valid $\forall w \in [0, T-1] \cap \mathbb{N}$:

$$\|\nabla^t_{s,a}[\tilde{r}_\varphi]_{t+w+1}\|_F^2 \leq \kappa^2\, C\, \|\nabla^{t+1}_{s,a}[r_\varphi]_{t+w+1}\|_F^2 \qquad \blacktriangleright \textit{Lemma } 2\text{(b)} \tag{100}$$

$$\leq \kappa^2\, C\, C^w\, \delta^2 \qquad \blacktriangleright \textit{Theorem } 1\text{(b)}, \textit{att} + 1 \tag{101}$$

$$= \kappa^2\, C^{w+1}\, \delta^2 \qquad \blacktriangleright \textit{repack product} \tag{102}$$

This shows that Eq. (99) is verified when $v = w + 1$, $\forall w \in [0, T-1] \cap \mathbb{N}$. Equation (99) is therefore valid $\forall v \in [1, T] \cap \mathbb{N}$.

Conclusion. We have shown that Eq. (99) is valid $\forall v \in [0, T] \cap \mathbb{N}$, which concludes the proof of Theorem 3(b). □

Theorem 4 (State-action value Lipschitzness) *We work under the assumptions laid out in both Lemma 2 and Theorem 3, and repeat the main lines here for Theorem 4 to be self-contained: (a) the functions f, μ and r_φ are C^0 and differentiable over their respective input spaces, and (b) the function r_φ is δ-Lipschitz over $\mathcal{S} \times \mathcal{A}$, i.e. $\|\nabla^u_{s,a}[r_\varphi]_u\|_F \leq \delta$, where $u \in [0, T] \cap \mathbb{N}$. Then the quantity $\nabla^u_{s,a}[\widetilde{Q}_\varphi]_u$ exists $\forall u \in [0, T] \cap \mathbb{N}$, and verifies:*

$$\|\nabla^t_{s,a}[\widetilde{Q}_\varphi]_t\|_F \leq \begin{cases} \kappa\delta \sqrt{\dfrac{1-\left(\gamma^2 C\right)^{T-t}}{1-\gamma^2 C}}, & \text{if } \gamma^2 C \neq 1 \\ \kappa\delta\sqrt{T-t}, & \text{if } \gamma^2 C = 1 \end{cases} \tag{103}$$

$\forall t \in [0, T] \cap \mathbb{N}$, where $C := A^2 \max(1, B^2)$, with A and B time-independent upper bounds of $\|\nabla^t_{s,a}[f]_t\|_\infty$ and $\|\nabla^t_s[\mu]_t\|_\infty$ respectively (see Eq. (86) for definitions of the supremum norms), and where κ satisfies $\kappa_u \leq \kappa \leq 1\, \forall u \in [0, T] \cap \mathbb{N}$.

Proof of Theorem 4 With finite horizon T, we have $\widetilde{Q}_\varphi(s_t, a_t) := \sum_{k=0}^{T-t-1} \gamma^k \tilde{r}_\varphi(s_{t+k}, a_{t+k})$, $\forall t \in [0, T] \cap \mathbb{N}$, since f, μ, r_φ, and $\tilde{r}_\varphi$ [cf. Eq. (68)] are all deterministic (no expectation). Additionally, since r_φ is assumes to be C^0 and differentiable over $\mathcal{S} \times \mathcal{A}$, $\widetilde{Q}_\varphi$ is by construction also C^0 and differentiable over $\mathcal{S} \times \mathcal{A}$. Consequently, $\nabla^u_{s,a}[\widetilde{Q}_\varphi]_u$ exists, $\forall u \in [0, T] \cap \mathbb{N}$. Since both r_φ and $\widetilde{Q}_\varphi$ are scalar-valued (their output space is $\mathbb{R}$), their Jacobians are the same as their gradients. We can therefore use the linearity of the gradient operator: $\nabla^t_{s,a}[\widetilde{Q}_\varphi]_t = \sum_{k=0}^{T-t-1} \gamma^k \nabla^t_{s,a}[\tilde{r}_\varphi]_{t+k}$, $\forall t \in [0, T] \cap \mathbb{N}$.

$$\|\nabla^t_{s,a}[\widetilde{Q}_\varphi]_t\|_F^2 = \left\| \sum_{k=0}^{T-t-1} \gamma^k \nabla^t_{s,a}[\tilde{r}_\varphi]_{t+k} \right\|_F^2 \qquad \blacktriangleright \textit{operator's linearity} \tag{104}$$

$$\leq \sum_{k=0}^{T-t-1} \gamma^{2k} \|\nabla^t_{s,a}[\tilde{r}_\varphi]_{t+k}\|_F^2 \qquad \blacktriangleright \textit{triangular inequality} \tag{105}$$

$$\leq \sum_{k=0}^{T-t-1} \gamma^{2k} \kappa^2 C^k \delta^2 \qquad \blacktriangleright \textit{Theorem 3 (b)} \tag{106}$$

$$= (\kappa\delta)^2 \sum_{k=0}^{T-t-1} \left(\gamma^2 C\right)^k \tag{107}$$

When $\gamma^2 C = 1$, we obtain $\|\nabla^t_{s,a}[\widetilde{Q}_\varphi]_t\|_F^2 = \delta^2(T-t)$. On the other hand, when $\gamma^2 C \neq 1$:

$$\|\nabla^t_{s,a}[\widetilde{Q}_\varphi]_t\|_F^2 \leq (\kappa\delta)^2 \frac{1-\left(\gamma^2 C\right)^{T-t}}{1-\gamma^2 C} \qquad \blacktriangleright \textit{finite sum of geometric series} \tag{108}$$

$$\Longrightarrow \quad \|\nabla^t_{s,a}[\widetilde{Q}_\varphi]_t\|_F^2 \leq \begin{cases} (\kappa\delta)^2 \dfrac{1-\left(\gamma^2 C\right)^{T-t}}{1-\gamma^2 C}, & \text{if } \gamma^2 C \neq 1 \\ (\kappa\delta)^2 (T-t), & \text{if } \gamma^2 C = 1 \end{cases} \tag{109}$$

By applying $\sqrt{\cdot}$ (monotonically increasing) to the inequality, we obtain the claimed result. □

Finally, we derive a corollary from Theorem 4 corresponding to the infinite-horizon regime.

Corollary 2 (Infinite-horizon regime) *Under the assumptions of Theorem 4, including that r_φ is δ-Lipschitz and that $\tilde{r}_\varphi$ is defined as in Eq. (68) over $\mathcal{S} \times \mathcal{A}$, and assuming that $\gamma^2 C < 1$, we have, in the infinite-horizon regime:*

$$\|\nabla^t_{s,a}[\widetilde{Q}_\varphi]_t\|_F \leq \frac{\kappa\delta}{\sqrt{1-\gamma^2 C}} \tag{110}$$

which translates into $\widetilde{Q}_\varphi$ being $\frac{\kappa\delta}{\sqrt{1-\gamma^2 C}}$-Lipschitz over $\mathcal{S} \times \mathcal{A}$.

Proof of Corollary 2 By following the proof of Corollary 1, using Theorem 3 instead of Theorem 1, we arrive directly at the claimed result. □

Remark 1 Say we were to write a proof analogous to the one laid out right above for Theorem 4, but using the time-*dependent* version of Theorem 3 instead of the time-*independent* version that we used in Eq. (106) (version 3(a) instead of 3(b)). Despite not being identifiable as a finite or infinite sum of geometric series, the expression we would get instead of Eq. (106) not only is a tighter bound by construction, but it also has an interesting form:

$$\|\nabla^t_{s,a}[\widetilde{Q}_\varphi]_t\|_F^2 \leq \sum_{k=0}^{T-t-1} \left[\gamma^{2k}\, \kappa_{t+k}^2\, \delta^2 \prod_{u=0}^{k-1} C_{t+u}\right] \qquad \blacktriangleright\text{Theorem } 3(a) \tag{111}$$

Going through the first operands of the sum, and looking solely at the "κ" and "C" factors, we have the following:

$$\kappa_t^2 \to \kappa_{t+1}^2\, C_t \to \kappa_{t+2}^2\, C_t C_{t+1} \to \kappa_{t+3}^2\, C_t C_{t+1} C_{t+2} \to \dots \to \kappa_T^2\, C_t C_{t+1} C_{t+2} \dots C_{T-1} \tag{112}$$

This observation tells us that, in the derived Lipschitz constant of $\widetilde{Q}_\varphi$, the reward preconditioner κ_t at time t can compensate for *all the past* values $\{C_v \mid v < t\}$. Intuitively, the more we wait to reduce κ_t, the more the *next* κ_t's will need to compensate for the "negligence" of their predecessors. Note, the product of $\{C_v \mid v < t\}$ compounds quickly.

6.6 Discussion II: implications and limitations of the theoretical guarantees

6.6.1 Provably more robust

Given that, in this work, we aligned the notion of robustness of a function approximator with the value of its Lipschitz constant (*more* robust means *lower* Lipschitz constant, cf. Sect. 4), and given that κ_t's upper bound κ verifies $\kappa \leq 1$ (cf. Lemma 2), we can write, from the result of Corollary 2:

$$\|\nabla^t_{s,a}[\widetilde{Q}_\varphi]_t\|_F \leq \frac{\kappa\delta}{\sqrt{1-\gamma^2 C}} = \kappa\,\Delta_\infty := \widetilde{\Delta}_\infty \leq \Delta_\infty \tag{113}$$

where $\Delta_\infty := \delta/\sqrt{1-\gamma^2 C}$ is the upper bound of Q_φ's Lipschitz constant that we derived in Corollary 1. Note, all of what is written in this remark concerns the infinite-horizon regime, but one can derive the finite-horizon counterpart trivially—using Theorem 2 instead of Corollary 1, and Theorem 4 instead of Corollary 2—to arrive at the same conclusion: $\widetilde{Q}_\varphi$ has a lower derived Lipschitz constant upper bound than Q_φ by a factor of $\kappa \leq 1$ and is therefore *provably more robust* than Q_φ. In other words, employing the simple PURPLE reward preconditioning to SAM has the effect of making the learned Q-value provably more robust.

6.6.2 Detached guide

Consider the following particular form for κ_t, $\forall t \in [0,T] \cap \mathbb{N}, \forall (s_t, a_t) \in \mathcal{S} \times \mathcal{A}$:

$$\kappa_t := \exp(-\alpha\,\epsilon_t) \qquad \Longrightarrow \qquad \tilde{r}_\varphi(s_t,a_t) := \kappa_t\, r_\varphi(s_t,a_t) := \exp(-\alpha\,\epsilon_t)\, r_\varphi(s_t,a_t) \tag{114}$$

where α is an inverse temperature hyper-parameter involved in the definition of the kernel of the Boltzmann or Gibbs probability distribution $\kappa_t := \exp(-\alpha\,\epsilon_t)$, (hence $0 < \kappa_t \leq 1$), and where $\epsilon_t \geq 0$ for now depicts an arbitrary non-negative energy function. κ_t is non-normalized, and as such, it is *not* a probability *per se*. Nonetheless, it still echoes the propensity or tendency of the state-action pair (s_t, a_t) to possess the property described by the non-negative energy ϵ_t, which we define momentarily. Low values of $\epsilon_t \geq 0$ will push the preconditioner towards the upper limit $\kappa_t \to 1$, while high energy values will make it tend towards the lower limit $\kappa_t \to 0$ with $\kappa_t > 0$. Equivalently, the preconditioned reward $\tilde{r}_\varphi$ will verify the approximate identity $\tilde{r}_\varphi(s_t,a_t) \approx r_\varphi(s_t,a_t)$ whenever ϵ_t approaches zero (from above), and $\tilde{r}_\varphi(s_t,a_t) \approx 0$ whenever the energy ϵ_t grows towards higher levels. Under this orchestration, we need $\frac{\mathrm{d}\epsilon_{t+k}}{\mathrm{d}s_t} = 0$ and $\frac{\mathrm{d}\epsilon_{t+k}}{\mathrm{d}a_t} = 0$ to be satisfied $\forall t \in [0,T] \cap \mathbb{N}, \forall k \in [0, T-t] \cap \mathbb{N}, \forall (s_t,a_t) \in \mathcal{S} \times \mathcal{A}$ for the derived robustness guarantees to be readily applicable (we laid out the properties κ_t must possess in Sect. 6.5, right before exposing Lemma 2).

In particular, the soft approximate $\mathfrak{C}$-validity pseudo-indicator [cf. Eq. (67)] is an instantiation of the κ_t form laid out in Eq. (114), where $\alpha = 1$ for the inverse temperature, and $\epsilon_t = \max(0, \|\nabla_{s_t,a_t} D_\varphi(s_t,a_t)\| - k)^2$ for the energy. In such an instance, $\tilde{r}_\varphi(s_t,a_t) \approx r_\varphi(s_t,a_t)$ whenever the pair (s_t,a_t) is approximately $\mathfrak{C}$-valid, formally, $\|\nabla^t_{s,a}[D_\varphi]_t\|_F \leq k$. Conversely, in the extreme scenario where $\|\nabla^t_{s,a}[D_\varphi]_t\|_F \gg k$, ϵ_t grows large, κ_t is approximately equal to 0, and $\tilde{r}_\varphi(s_t,a_t) \approx 0$. As such, in effect, the agent's policy μ_θ is punished for selecting actions that do not satisfy the approximate $\mathfrak{C}$-validity condition above. Besides, it is punished in accordance to how far outside the allowed range, $[0, k]$, the norm of the Jacobian of D_φ gets. Nonetheless, in this particular instance, the empirical observations we have made in Sect. 6.4 attest to the fact that, provided the right choice of λ scaling factor and ζ distribution (both characterizing the gradient penalization), the approximate $\mathfrak{C}$-validity constraint $\|\nabla^t_{s,a}[D_\varphi]_t\|_F \leq k$ can easily be satisfied 100% of the time by *only* regularizing D_φ. For D_φ's k-Lipschitzness to be ensured, there is therefore no need to further alter the rewards provided to the agent's policy μ_θ through PURPLE's pessimistic reward preconditioning. Note, however, that under such a ϵ_t formulation, we see that we clearly have

$\frac{\mathrm{d}\epsilon_{t+k}}{\mathrm{d}s_t} \neq 0$ and $\frac{\mathrm{d}\epsilon_{t+k}}{\mathrm{d}a_t} \neq 0$, $\forall t \in [0, T] \cap \mathbb{N}, \forall k \in [0, T-t] \cap \mathbb{N}, \forall (s_t, a_t) \in \mathcal{S} \times \mathcal{A}$. While this does not mean that the studied entities are not robust, it prevents us from applying our derived results to guarantee such robustness.

Generally speaking, we will probably make the same observation whenever ϵ_t is defined from a constraint we want to enforce on a learned function approximation, for regularization purposes. Indeed, verifying said desideratum on the function approximator directly via the application of a regularizer seems to always be the easiest (since most direct) solution to encourage the satisfaction of a constraint on a *differentiable* function (e.g. D_φ, μ_θ). Constraints involving the Jacobian of a (*a fortioni* differentiable) function of the learned system (e.g. $\|\nabla^t_{s,a}[D_\varphi]_t\|_F \leq k$) is a particular case of the general class of constraints for which *direct* regularization is *a priori* prefereable to an analogous reward shaping as dictated by Eq. (114). On the flip side, due to the fact that the reward—albeit learned as a parametric function—is treated as an input in our computational graph, it is not differentiated through and *can* consequently be augmented with non-differentiable nodes through the design of ϵ_t. In other words, even if it is preferable to apply regularization directly the objective of the regularized function approximator for it to satisfy some constraint, it might not always be possible to do so directly. In that case, guiding the policy towards areas of the state-action landscape that satisfy said constraint could be a surrogate solution, albeit far less preferable than acting on the targeted approximator directly.

As such, by aligning ϵ_t with said constraint, Eq. (114) offers a way for the policy to act in view of the satisfaction of said constraint *while* enjoying the considerable advantage of being able to treat ϵ_t as a *black box*. We will leverage this *universality* in the next discussion point.

6.6.3 Partial compensation of compounding variations

In reaction to the theoretical robustness guarantees derived in Theorem 2 and Corollary 1, we have discussed earlier in Sect. 6.2.3 that, if the variations *in space* of the policy or the dynamics are large in the early stage of an episode (i.e. when $0 \leq t \ll T$), then Δ_t (the variation bound on Q_φ) might explode. As results, $\|\nabla^t_{s,a}[Q_\varphi]_t\|_F$ would then be unbounded, leaving us unable to guarantee the robustness of the learned Q-value Q_φ. The earlier large variations in either or both the policy and dynamics manifest, the more likely these variations are to compound to unreasonably high levels. Concretely, the degree of such compounding variations in space is entirely determined by the operand $\gamma^2 C$ that appears in the variation bounds derived in both Theorem 2 and Corollary 1. The exact same line of reasoning holds for the variation bounds laid out later in Sect. 6.5, in both Theorem 4 and Corollary 2 respectively. These guarantees unanimously agree on the critical role that C plays in the robustness bounds, which we here called variation bounds indifferently. Loosely, *high* values of C prevent Q_φ from enjoying the Lipschitzness guarantees laid out in Sects. 6.1 and 6.5. As such, it is paramount to devise a way to keep C in check by somewhat controling its magnitude, thereby preventing it from voiding our theoretical guarantees and from adopting a brittle behavior. We defined C in Lemma 1(b) as $C := A^2 \max(1, B^2)$, where A and B are time-independent upper bounds of the supremum Frobenius norms of the Jacobians of the dynamics f and the policy μ, $\|\nabla^t_{s,a}[f]_t\|_\infty$ and $\|\nabla^t_s[\mu]_t\|_\infty$, respectively [cf. Eq. (32) for definitions of the supremum norms $\|\cdot\|_\infty$]. Simply, $\forall t \in [0, T] \cap \mathbb{N}, \forall (s_t, a_t) \in \mathcal{S} \times \mathcal{A}$, $\|\nabla^t_{s,a}[f]_t\|_\infty \leq A$ and $\|\nabla^t_s[\mu]_t\|_\infty \leq B$. As such, to devise a way to limit the magnitude of C, we seek ways to limit the respective magnitudes of the A and B majorants. Similarly to the learned surrogate reward core D_φ, the policy μ_θ followed by the agent (of which μ is a placeholder) is learned as a parametric function approximator, enabling us to tame B by applying a gradient penalty regularizer *directly* on the policy (exactly like we already do to ensure that D_φ remains k-Lipschitz-continuous).

By contrast, we can not tame A the same way (via direct regularization applied onto f), due to the transition function f of the world (whether real or simulated) being a black box that we can not even query at will. Not only is f non-differentiable (the real world never is; non-trivial simulated worlds virtually never are), but we also can *not* evaluate it at *any state-action pair whenever we want*. Our desideratum then ultimately boils down to finding a way to keep A in check, since the usual candidate to enforce Lipschitzness (applying a regularizer on the Jacobian directly)—which is the preferable option by far for D_φ and μ_θ—is out of the question for f, as we have established. Despite the fact that, by nature, we can not change f in the MDP $\mathbb{M}$, we *can* change the transition function f' that effectively takes the place of f in practice and underlies the effectively observed MDP $\mathbb{M}'$ by urging the agent's policy μ_θ to avoid areas of the state-action landscape $\mathcal{S} \times \mathcal{A}$ that display high $\|\nabla^t_{s,a}[f']_t\|_\infty$ values. In fact, f' changes continually (f' is non-stationary) throughout the learning process as the preferences of the agent evolve across learning episodes. It is therefore fair to posit that we can devise a way to skew the policy towards areas of $\mathcal{S} \times \mathcal{A}$ where $\|\nabla^t_{s,a}[f']_t\|_\infty$ is tightly upper-bounded. As such, we can keep A in check by keeping $\|\nabla^t_{s,a}[f']_t\|_\infty$ in check in practice, which can be approximately achieved by keeping $\|\nabla^t_{s,a}[f_\psi]_t\|_\infty$ in check, where $f_\psi : \mathcal{S} \times \mathcal{A} \rightarrow \mathcal{S}$ is a learned functional approximation of the effective dynamics f'.

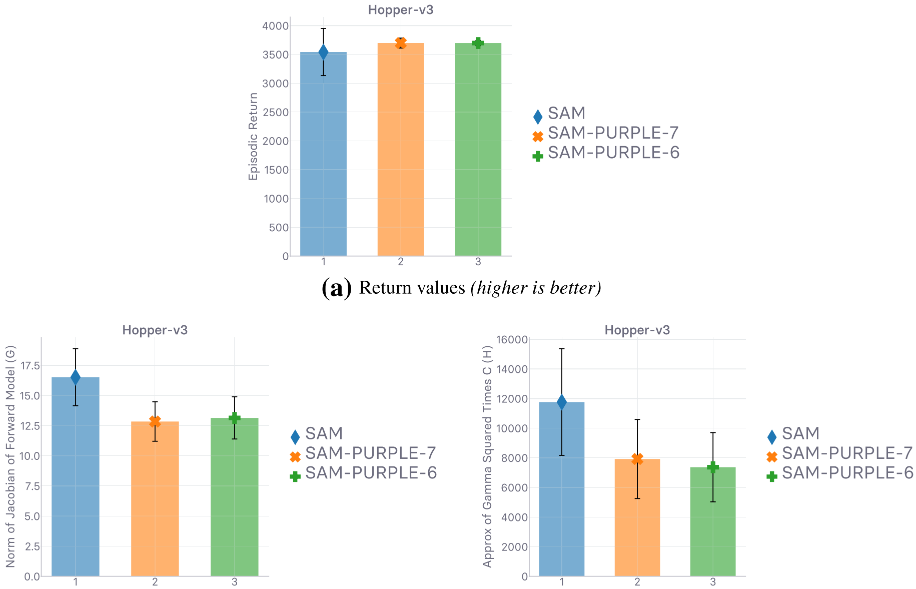
In fine, we urge the constraint $\|\nabla^t_{s,a}[f]_t\|_\infty \leq A$ to be satisfied by encouraging μ_θ to avoid areas where $\|\nabla^t_{s,a}[f_\psi]_t\|_\infty$ is high, which itself can be relaxed into $\|\nabla^t_{s,a}[f_\psi]_t\|_F$. Note, even if f_ψ is differentiable, regularizing it via gradient penalization does *not* have any effect on the value of $\|\nabla^t_{s,a}[f']_t\|_\infty$, since the agent does *not* interact with f_ψ, but with f'. For our line of reasoning to hold, we want $\|\nabla^t_{s,a}[f_\psi]_t\|_F$ to be a high-fidelity depiction of $\|\nabla^t_{s,a}[f']_t\|_\infty$.

We maintain the parametric model f_ψ because it allows us to approximate the norm of the Jacobian of the dynamics wherever we want, whenever we want. In order for μ_θ to avoid areas where $\|\nabla^t_{s,a}[f_\psi]_t\|_F$ is high, we leverage the universal preconditioner form exhibited in Eq. (114). Concretely, we reward the agent *less* for *not* navigating areas of $\mathcal{S} \times \mathcal{A}$ that satisfy the constraint $\|\nabla^t_{s,a}[f_\psi]_t\|_F \leq \tau$. The Lipschitz constant τ we want to enforce onto f_ψ is a hyper-parameter that must be tuned, like k for D_φ. We push μ_θ towards areas where $\|\nabla^t_{s,a}[f_\psi]_t\|_F \leq \tau$ (where f_ψ is τ-Lipschitz-continuous, thereby also satisfying the premise of the guarantees) by defining the energy function ϵ^ψ_t in the *model-based* preconditioner κ^ψ_t as a one-sided gradient penalty, as follows:

$$\tilde{r}^{\psi}_{\varphi}(s_t, a_t) := \kappa^{\psi}_t \, r_\varphi(s_t, a_t) \quad \text{where} \quad \begin{cases} \kappa^{\psi}_t & := \max\big(\kappa_{\min}, \exp\big(-\alpha\,\epsilon^{\psi}_t\big)\big) \quad \text{with} \\ \epsilon^{\psi}_t & := \max\big(0, \|\nabla^t_{s,a}[f_\psi]_t\|_F - \tau\big)^2 \Big/ \sigma^{\psi}_{\text{ON}} \end{cases} \tag{115}$$

$$\iff \quad \tilde{r}^{\psi}_{\varphi}(s_t, a_t) := \max\left(\kappa_{\min}, \exp\left(-\frac{\alpha}{\sigma^{\psi}_{\text{ON}}} \max\big(0, \|\nabla^t_{s,a}[f_\psi]_t\|_F - \tau\big)^2\right)\right) r_\varphi(s_t, a_t) \tag{116}$$

$\forall t \in [0, T] \cap \mathbb{N}, \forall (s_t, a_t) \in \mathcal{S} \times \mathcal{A}$, where $\sigma^{\psi}_{\text{ON}}$ denotes an online, running estimate of the standard deviation of $\max(0, \|\nabla_{s_t,a_t} f_\psi(s_t, a_t)\|_F - \tau)^2$. For completeness, we remind here that we used the same online normalization technique in our RED experiments (cf. Sect. 5.5), inspired by the discussion laid out in in Burda et al. (2018) on the importance of



(a) Return values *(higher is better)*

(b) G values *(lower is better)*

(c) H values *(lower is better)*

Fig. 11 Empirical evaluation of **a** the empirical return, **b** the norm of the Jacobian of the forward model f_ψ defined by $G:=\|\nabla^t_{s,a}[f_\psi]_t\|_F$, and **c** the approximation of $\gamma^2 C$ defined by $H:=\gamma^2\|\nabla^t_{s,a}[f_\psi]_t\|_F^2 \max(1, \|\nabla^t_s[\mu_\theta]_t\|_F^2)$. *SAM-PURPLE-7* and *SAM-PURPLE-6* are two instantiations of SAM (cf. Algorithm 1), augmented with the *model-based* instantiation of PURPLE whose template is laid out in Eqs. (115) and (116), with $\tau = 7$ and $\tau = 6$ respectively. We indicate how to read the plots (whether *lower* or *higher* is better) in the caption of each column. Despite displaying overlapping return curves, note how *tighter* the standard deviation envelope is for PURPLE runs. Runtime is 96 h

such normalization technique when the reward is grounded on a prediction loss. Considering the edge cases, and omitting here the clipping to κ_{min}, when ϵ_t^Ψ is close to zero, κ_t^Ψ is approximately equal to 1, i.e. $\tilde{r}_\varphi(s_t, a_t) \approx r_\varphi(s_t, a_t)$ [cf. Eqs. (115), (116)]. Conversely, in the extreme scenario where ϵ_t^Ψ is very large (i.e. $\|\nabla^t_{s,a}[f_\psi]_t\|_F \gg \tau$), κ_t^Ψ is approximately equal to 0, and $\tilde{r}_\varphi(s_t, a_t) \approx 0$.

Looking at the model-based instantiation of PURPLE laid out in Eq. (115), and specifically of the form exhibited in Eq. (113), we see that the energy ϵ_t^Ψ depends on the current state s_t and action a_t. Indeed, from the definitions of ϵ_t^Ψ and κ_t^Ψ, we immediately see that $\frac{\mathrm{d}\epsilon_{t+k}^\Psi}{\mathrm{d}s_t} \neq 0$ and $\frac{\mathrm{d}\epsilon_{t+k}^\Psi}{\mathrm{d}a_t} \neq 0$, which directly leads to $\frac{\mathrm{d}\kappa_{t+k}^\Psi}{\mathrm{d}s_t} \neq 0$ and $\frac{\mathrm{d}\kappa_{t+k}^\Psi}{\mathrm{d}a_t} \neq 0$, $\forall t \in [0, T] \cap \mathbb{N}, \forall k \in [0, T - t] \cap \mathbb{N}, \forall (s_t, a_t) \in \mathcal{S} \times \mathcal{A}$. As such, the crafted preconditioner does not satisfy the eligibily conditions for the derived theoretical guarantees to be applicable, which were represented in condensed form in Sect. 6.5, right before exposing Lemma 2. If we had used the *supremum* Frobenius norm $\|\nabla^t_{s,a}[f_\psi]_t\|_\infty$ to formulate ϵ_t^Ψ instead of relaxing it to $\|\nabla^t_{s,a}[f_\psi]_t\|_F$, its non-supremum counterpart, ϵ_t^Ψ would *not* depend on s_t and a_t (or any visited state or picked action), and our robustness guarantees would be readily applicable. Still, such a supremum Frobenius norm is intractable in practice. In order for us to be able to evaluate the developed prototype empirically, we resorted to the obvious tractable relaxation consisting in simply dropping the supremum altogether for this diagnostics-oriented case.

Now that we have laid out how the pessimistic model-based preconditioner κ^ψ_t impacts the reward received by the agent artificially upon interaction, we consider how this preconditioning affects the Lipschitz constant of $\widetilde{Q}_\varphi$ in the infinite-horizon setting, denoted by $\widetilde{\Delta}_\infty$ [cf. Eq. (113)]. As $\|\nabla^t_{s,a}[f]_t\|_\infty$ grows larger, its upper-bound A grows larger. Assuming B (upper-bounding $\|\nabla^t_s[\mu]_t\|_\infty$) remains unaffected and remains constant, larger values of A cause larger values of $C{:=}A^2\max(1,B^2)$, which in turn push the denominator of the Lipschitz constant $\widetilde{\Delta}^\psi_\infty{:=}\kappa^\psi_t\delta/\sqrt{1-\gamma^2C}$ towards 0 from above, exposing $\widetilde{\Delta}^\psi_\infty$ to diverge to $+\infty$. Without preconditioning ($\kappa^\psi_t=1$), the task of compensating for such a low-valued denominator would be left to δ alone, and picking $\delta\approx 0$ would be the only way to maintain the robustness bound from diverging. With preconditioning however, we can also try to prevent it from diverging with the preconditioner κ^ψ_t, whose value can be set far more finely (*per* timestep). Specifically, with the κ^ψ_t formulation laid out in Eqs. (115) and (116), and assuming $\|\nabla^t_{s,a}[f_\psi]_t\|_F$ approximates $\|\nabla^t_{s,a}[f]_t\|_\infty$ well—i.e. $\|\nabla^t_{s,a}[f_\psi]_t\|_F$ mirrors the behavior of $\|\nabla^t_{s,a}[f]_t\|_\infty$, we hold an analogous line of reasoning for the *numerator* of $\widetilde{\Delta}^\psi_\infty$. As $\|\nabla^t_{s,a}[f]_t\|_\infty$ grows larger, $\|\nabla^t_{s,a}[f_\psi]_t\|_F$ grows larger (with we can translate into $\|\nabla^t_{s,a}[f_\psi]_t\|_F\gg\tau$), which consequently pushes the preconditioner κ^ψ_t towards 0 from above. As such, the premise "$\|\nabla^t_{s,a}[f]_t\|_\infty$ *grows larger*" pushes both the numerator and denominator of $\widetilde{\Delta}^\psi_\infty$ towards 0 from above, taming the quotient in effect. Nonetheless, note, we can not *eliminate* the influence of $\|\nabla^t_{s,a}[f]_t\|_\infty$ on the bound. Still, the *partial compensation* of the detrimental impact of $\|\nabla^t_{s,a}[f]_t\|_\infty$ on $\widetilde{\Delta}^\psi_\infty$—that we were able to secure by proposing the model-based pessimistic reward preconditioning κ^ψ_t [cf. Eqs. (115), (116)]—can be tuned extensively in practice to achieve the desired level of compensation. We used $\kappa_{\min}=0.7$, $\alpha=1$, and $\tau\in\{6,7\}$ in the experiments we conducted to showcase how the proposed model-based reward preconditioning laid out above can help us achieve our robustness desideratum.

Since we aim to *showcase* its potential benefits, as opposed to convince the reader to plug this preconditioning method in every future architecture, we conducted *illustrative* experiments only in the `Hopper` environment (neither the easiest, nor the hardest among the ones considered, cf. Table 1). Note, when it comes to D_φ's gradient penalty regularization, we use the default ζ and λ (cf. Sect. 6.3): the directed ζ distribution of WGAN-GP, with $\lambda=10$ as scaling factor. Since the evaluated policy is penalized for navigating areas of $\mathcal{S}\times\mathcal{A}$ where $\|\nabla^t_{s,a}[f_\psi]_t\|_F>\tau$, we monitor $G{:=}\|\nabla^t_{s,a}[f_\psi]_t\|_F$. We expect to observe *lower* values of G when using the studied preconditioning. In order to grasp the extent to which variations can compound in the system, and therefore highlight the need for mechanims allowing the main method to contain such compounding of variations (like the proposed one), we also monitor an approximation of γ^2C, relaxed as $H{:=}\gamma^2\|\nabla^t_{s,a}[f_\psi]_t\|^2_F\max(1,\|\nabla^t_s[\mu_\theta]_t\|^2_F)$. We expect to see the same ranking of methods in the plots depicting G and H respectively. These are all reported in Fig. 11.

Note, the steep surge in overall computational cost caused by the evaluation of the monitored metrics (G and H) and expecially κ^ψ_t lowered the number of iterations our agent could do in the allowed runtime. As such, we increased said runtime from the usual 0.5-day or 2-day duration to a 4-day duration (or 96 h) Such runs are more costly to orchestrate, hence the sparser array of experiments to offset the steeper cost in compute. In Fig. 11, we observe that, at evaluation time, the model-based PURPLE instantiation in Eqs. (115) and (116) indeed enables the agent to achieve *lower* values of G and H, with the *same* episodic return. Said differently, it seems that the agent—with preconditioning, compared to the one without—achieves the same proficiency, with the same convergence speed, while making decisions that are *safer* in terms of incurred variations of the approximate dynamics f_ψ. *So, even if the preconditioner is not needed to reach a higher return (or reach it faster) per se, we have showcased that the studied model-based reward preconditioning can increase the robustness of the main method by augmenting it with the means to tame a priori untamable entities in the system (here, the dynamics).* Still, the studied model-based instantiation of PURPLE is set back by several drawbacks. (a) We need to maintain a forward model f_ψ that approximates the effective transition function f'. (b) To be estimated, κ_t^ψ requires explicit calls to an automatic differentiation library, making its frequent computation (every time a mini-batch is sampled from the replay buffer) extremely expensive overall. (c) The threshold τ (to be enforced as Lipschitz constant for f_ψ) must be set such that *not every* decision made by the agent is penalized, while making sure it is still strict enough in that respect. Besides, we observed in practice that the range of values taken by $\|\nabla^t_{s,a}[f_\psi]_t\|_F$ varies greatly across environments. As such, τ must be tuned carefully per environment, making the overall process tedious and computationally expensive. In effect, this brings us back to the original issues of reward shaping (Ng et al. 1999), that adversarial IL (Ho and Ermon 2016) circumvented.

6.6.4 Total compensation of compounding variations

Inspired by the insight laid out in Remark 1, we derive theoretical guarantees that characterize the robustness of $\widetilde{Q}_\varphi$ when using a preconditioner defined as follows:

$$\begin{aligned}\kappa_{t+k} &:= \frac{1}{\sqrt{\prod_{u=0}^{k-1} C_{t+u}}} \qquad \text{where cf. Eq. ((86))} \; \forall v \in [0, T-1], \\ C_v &:= \|\nabla^v_{s,a}[f]_v\|^2_\infty \max\left(1, \|\nabla^{v+1}_s[\mu]_{v+1}\|^2_\infty\right)\end{aligned} \tag{117}$$

$\forall t \in [0, T] \cap \mathbb{N}$, and $\forall k \in [0, T-t] \cap \mathbb{N}$. Since the norms involved in C_v are *supremum* ones, the preconditioner κ_t verifies $\frac{\mathrm{d}\kappa_{t+k}}{\mathrm{d}s_t} = 0$ and $\frac{\mathrm{d}\kappa_{t+k}}{\mathrm{d}a_t} = 0$, $\forall t \in [0, T] \cap \mathbb{N}, \forall k \in [0, T-t] \cap \mathbb{N}, \forall (s_t, a_t) \in \mathcal{S} \times \mathcal{A}$. The reward preconditioner therefore verifies the properties one must satisfy for the derived robustness guarantees to be applicable (cf. Sect. 6.5).

Again, note, the property $\kappa_t \leq 1$ is only ever used in Sect. 6.6.1, and has not been leveraged anywhere else. Given that the developed theory still holds if $\exists t \in [0, T] \cap \mathbb{N}$ such that $\kappa_t > 1$, the fact that the preconditioner defined in Eq. (117) does not necessarily lie in the (0, 1] interval is not an issue *a priori*. Still, in practice, it will virtually always be below 1.

We now derive the associated counterparts of Theorem 4 and Corollary 2.

Theorem 5 (State-action value Lipschitzness) *We work under the assumptions laid out in both Lemma 2 and Theorem 3, and repeat the main lines here for Theorem 5 to be self-contained:* (*a*) *the functions* f, μ *and* r_φ *are* C^0 *and differentiable over their respective input spaces, and* (*b*) *the function* r_φ *is* δ*-Lipschitz over* $\mathcal{S} \times \mathcal{A}$, *i.e.* $\|\nabla^u_{s,a}[r_\varphi]_u\|_F \leq \delta$, *where* $u \in [0, T] \cap \mathbb{N}$. *Then the quantity* $\nabla^u_{s,a}[\widetilde{Q}_\varphi]_u$ *exists* $\forall u \in [0, T] \cap \mathbb{N}$. *Assuming in addition that the reward preconditioner used on* r_φ *to obtain* $\tilde{r}_\varphi$ *is defined according to Eq.* (117), *the action-value* $\widetilde{Q}_\varphi$ *verifies:*

$$\|\nabla^t_{s,a}[\widetilde{Q}_\varphi]_t\|_F \leq \delta \sqrt{\frac{1 - \gamma^{2(T-t)}}{1 - \gamma^2}} \tag{118}$$

$\forall t \in [0, T] \cap \mathbb{N}$. *Note, the bound now only depends on* δ, γ, *and* $T - t$, *the* "*remaining time in the episode*".

Proof of Theorem 5 The reward preconditioner used to assemble $\tilde{r}_\varphi$ from r_φ is defined according to Eq. (117). As carried out in Remark 1, we start the proof of Theorem 5 analogously to the one laid out for Theorem 4, but using the time-*dependent* version of Theorem 3 instead of the time-*independent* version that we used in Eq. (106) (version 3 *(a)* instead of Theorem 3 *(b)*). Our starting point then aligns with the crux of Remark 1. As such:

$$\|\nabla^t_{s,a}[\widetilde{Q}_\varphi]_t\|_F^2 \leq \sum_{k=0}^{T-t-1} \left[\gamma^{2k}\, \kappa_{t+k}^2\, \delta^2 \prod_{u=0}^{k-1} C_{t+u}\right] \qquad \blacktriangleright \text{Theorem 3(a)} \tag{119}$$

$$= \sum_{k=0}^{T-t-1} \left[\gamma^{2k} \frac{1}{\prod_{u=0}^{k-1} C_{t+u}}\, \delta^2 \prod_{u=0}^{k-1} C_{t+u}\right] \qquad \blacktriangleright \textit{Eq.}\ (117) \tag{120}$$

$$= \delta^2 \sum_{k=0}^{T-t-1} (\gamma^2)^k \tag{121}$$

Since we defined γ to be within the interval [0, 1) in Sect. 3, we trivially have $\gamma^2 < 1$, hence $\gamma^2 \neq 1$ and:

$$\|\nabla^t_{s,a}[\widetilde{Q}_\varphi]_t\|_F^2 \leq \delta^2 \frac{1 - \gamma^{2(T-t)}}{1 - \gamma^2} \qquad \blacktriangleright \textit{finite sum of geometric series} \tag{122}$$

By applying $\sqrt{\cdot}$ (monotonically increasing) to the inequality, we obtain the claimed result. □

Finally, we derive a corollary from Theorem 5 corresponding to the infinite-horizon regime.

Corollary 3 (Infinite-horizon regime) *Under the assumptions of Theorem 5, including that* r_φ *is* δ*-Lipschitz and that* $\tilde{r}_\varphi$ *is defined as in Eq.* (68) *over* $\mathcal{S} \times \mathcal{A}$, *we have, in the infinite-horizon regime:*

$$\|\nabla^t_{s,a}[\widetilde{Q}_\varphi]_t\|_F \leq \frac{\delta}{\sqrt{1 - \gamma^2}} \tag{123}$$

which translates into $\widetilde{Q}_\varphi$ *being* $\frac{\delta}{\sqrt{1-\gamma^2}}$*-Lipschitz over* $\mathcal{S} \times \mathcal{A}$.

Proof of Corollary 3 As we adapt the proof of Theorem 5 to the infinite-horizon regime, Eq. (121) becomes

$$\|\nabla^t_{s,a}[\widetilde{Q}_\varphi]_t\|_F^2 \leq \delta^2 \sum_{k=0}^{+\infty} (\gamma^2)^k = \frac{\delta^2}{1 - \gamma^2} \qquad \blacktriangleright \textit{infinite sum of geometric series} \tag{124}$$

since we defined γ to be within the interval [0, 1) in Sect. 3, i.e. $\gamma^2 < 1$. We then apply $\sqrt{\cdot}$ to the inequality. □

In these theoretical guarantees, we have shown that by carefully crafting PURPLE's reward preconditioner according to Eq. (117), we obtain upper-bounds $\widehat{\Delta}_\infty$ on the Lipschitz constant of the resulting action-value $\widetilde{Q}_\varphi$ that are *independent* of C_v, $\forall v \in [0, T-1]$ —where $C_v := \|\nabla^v_{s,a}[f]_v\|^2_\infty \max\left(1, \|\nabla^{v+1}_s[\mu]_{v+1}\|^2_\infty\right)$ [cf. Eq. (117)]. In other words, we have shown that such preconditioner design allows us to *totally* compensate for the compounding variations (a) first tackled in the discussion led in Sect. 6.2.3, and (b) then addressed *only partially* by the model-based reward preconditioning discussed profusely in Sect. 6.6.3 (of which we showcase the applicability in practice). Echoing what motivated the emergence of Remark 1 in the first place, the form adopted by the reward preconditioning [cf. Eq. (117)] that allowed us to derive the robustness guarantees of Theorem 5 and Corollary 3 enjoys an insightful and intuitive *interpretation*. Going through the elements of the series described by the preconditioner of Eq. (117), $(\kappa_{t+k})_k$, $\forall t \in [0, T] \cap \mathbb{N}$, and $\forall k \in [0, T-t] \cap \mathbb{N}$, we have the following sequence of consecutive preconditioning values:

$$\kappa_{t+k}\big|_{k=0} = \kappa_t := 1 \;\rightarrow\; \kappa_{t+k}\big|_{k=1} = \kappa_{t+1} := \frac{1}{\sqrt{C_t}} \;\rightarrow\; \kappa_{t+k}\big|_{k=2} = \kappa_{t+2} := \frac{1}{\sqrt{C_t C_{t+1}}}$$
$$\rightarrow\; \kappa_{t+k}\big|_{k=3} = \kappa_{t+3} := \frac{1}{\sqrt{C_t C_{t+1} C_{t+2}}} \;\rightarrow\; \dots \;\rightarrow\; \kappa_{t+k}\big|_{k=T-t} = \kappa_T := \frac{1}{\sqrt{C_t C_{t+1} C_{t+2} \dots C_{T-1}}} \tag{125}$$

We observe that, when purposely defined as such, the reward preconditioner κ_{t+k} at a given stage $t+k$ compensates for the C_v's of all the *previous* timesteps—backwards from $t+k-1$ to t, where $\widetilde{Q}_\varphi$'s Lipschitz constant is characterized. In order to prevent the upper-bound on $\|\nabla^t_{s,a}[\widetilde{Q}_\varphi]_t\|_F$ to be burdened by incipient, potentially prone to compound, variation of $C_v := \|\nabla^v_{s,a}[f]_v\|^2_\infty \max\left(1, \|\nabla^{v+1}_s[\mu]_{v+1}\|^2_\infty\right)$, the preconditioner can *actively anticipate* said incipient compounding variations to compound further within the time remaining in the episode by preemptively squashing the *current* surrogate reward at $t+k$ based on how much C_v's variations have accumulated since t until $t+k-1$. The proposed interpretation of the studied preconditioner aligns with our intuitive desideratum: *"if you want to fend off from compounding of variations that threaten the stability of your action-value, make the latter more robust as soon as you see, from past metrics—here, monitored C_v values—that said variations might actually compound soon"*.

Despite appealing in principle thanks to its salient interpretation, and justified by theoretical guarantees, we did not experiment with the proposed preconditioner in practice. Indeed, considering how we have shown in Sect. 6.6.3 that the values in effect taken by $C_v := \|\nabla^v_{s,a}[f]_v\|^2_\infty \max\left(1, \|\nabla^{v+1}_s[\mu]_{v+1}\|^2_\infty\right)$ do not seem to affect the agent's return in practice, we do not expect the interpretable preconditioner tackled in this discussion to bring anything *practically* in the considered environments. Using a gradient penalty constraint to induce local Lipschitz-continuity of the function at the core of the reward function is, in a sense, *all you need* to achieve peak expert performance in the considered off-policy generative adversarial imitation learning setting. Still, we believe the design and study of methods able to actively tune their level of robustness—aligned in this work with the concept of spatial, local Lipschitz-continuity—depending on the choices (or more pessimistically, on the *mistakes*) made by the agent to be an interesting avenue of future work. Besides, by augmenting the reward-less MDP $\mathbb{M}$ (from which we first stripped the environmental reward) with our adversarially learned reward, preconditioned in line with Eq. (117), the resulting MDP has a *memory*, since the reward $\tilde{r}_\varphi$ depends on entities (C_v's) from previous timesteps in the episode. In effect, due to such a reward preconditioning formulation, the Markov property is not satisfied anymore as, given the present, the future now *does* depend on the past. We believe the observations made and results derived in this work could pave the way to further investigations aiming to decipher known methods and ultimately pinpoint the *most minimal* setup for it to still do well.

7 Conclusion

In this work, we conducted an in-depth study of the stability problems incurred by off-policy generative adversarial imitation learning. Our contributions closely follow the line of reasoning, and are as follows. (1) We characterized the various inherent hindrances the approach suffers from, in particular how learned parametric rewards affect the learned parametric state-action value. (2) We showed that enforcing a local Lipschitz-continuity constraint on the discriminator network used to formulate the imitation surrogate reward is a *sine qua non* condition for the approach to empirically achieve expert performance in challenging continuous control problems, within a number of timesteps that still enable us to call the method sample-efficient. (3) In line with the first and second steps, we derived theoretical guarantees that characterize the Lipschitzness of the Q-function when the reward is assumed δ-Lipschitz-continuous. Note, the reported theoretical results are valid for any reward satisfying the condition, nothing is specific to imitation. (4) We propose a new RL-grounded interpretation of the usual GAN gradient penalty regularizers—differing by *where* they induce Lipschitzness—along with an explanation as to (a) why they all have such a positive impact on stability, but also (b) how to make sense of the empirical gap between them. (5) We show that, in effect, the consistent satisfaction of the Lipschitzness constraint on the reward is a strong predictor of how well the mimicking agent performs empirically. (6) Finally, we introduce a pessimistic reward preconditioning technique which (a) makes the base method it is plugged into provably more robust, and (b) is accordingly backed by several theoretical guarantees. As in (3), these guarantees are not not specific to imitation and have a wide range of applicability. We give an illustrative example of how the technique can help further increasing the robustness of the method it is plugged into empirically.

Appendix 1: Hyper-parameters

The function approximators used in every learned module are two-layer multi-layer perceptrons, but the widths of their respective layers differ. We use layers of sizes 100–100 for the discriminator (from which the reward is formulated), 300–200 for the actor, and 400–300 for the critic, as they achieved the best overall result across the environments of the suite in our early experiments. Unless specifically stated otherwise, the discriminator network uses spectral normalization (Miyato et al. 2018) at every layer, while the actor and critic networks both use layer normalization (Ba et al. 2016) at every layer. Every neural network is initialized via orthogonal initialization (Saxe et al. 2013). Each network has its own optimizer (cf. Sect. 4 for a complete description of the optimization problems the networks of parameter φ, ω, and θ are involved in, along with the loss they optimize). We use Adam (Kingma and Ba 2014) for each of them, with respective learning rates reported in Table 2, while the other parameters of the optimizer are left to the default PyTorch (Paszke et al. 2019) values. In practice, we replace the squared error loss involved in the loss optimized by the critic [cf. Eq. (3)] by the Huber loss, as is commonly done in temporal-difference learning with function approximation and target networks (Mnih et al.

Table 2 Hyper-parameters used in this work. Unless explicitly stated otherwise, every method uses these

Hyper-parameter	Selected Value
Training steps per iteration	2
Evaluation steps per iteration	10
Evaluation frequency	10
Actor learning rate	2.5×10^{-4}
Critic learning rate	2.5×10^{-4}
Actor clip norm	40
Critic weight decay scale	0
Rollout length	2
Effective batch size	1024
Discount factor γ	0.99
Replay buffer $\mathcal{R}$ size	100000
Exploration (cf. Sect. 4)	$\sigma_a = 0.2, \sigma_b = 0.2$
Param. noise update frequency	50
Target update Polyak scale τ	0.005
Multi-step lookahead n	10
Target smoothing - noise σ (Fujimoto et al. 2018)	0.2
Target smoothing - noise clip (Fujimoto et al. 2018)	0.5
Actor update delay (Fujimoto et al. 2018)	2
Reward training steps per iteration	1
Agent training steps per iteration	1
Discriminator learning rate	5.0×10^{-4}
Entropy regularization scale	0.001
Positive label-smoothing	Real labels $\sim$ unif(0.7, 1.2)
Positive-Unlabeled (Xu and Denil 2019)—coeff. η	0.25

The *"effective"* batch size corresponds to the size of the mini-batch *aggregated* across parallel workers of the distributed architecture. In our case, every worker—of the grand total of $n = 16$ workers—samples a mini-batch of size 64 from its (individual) replay buffer, resulting in an effective batch size of $64 \times 16 = 1024$

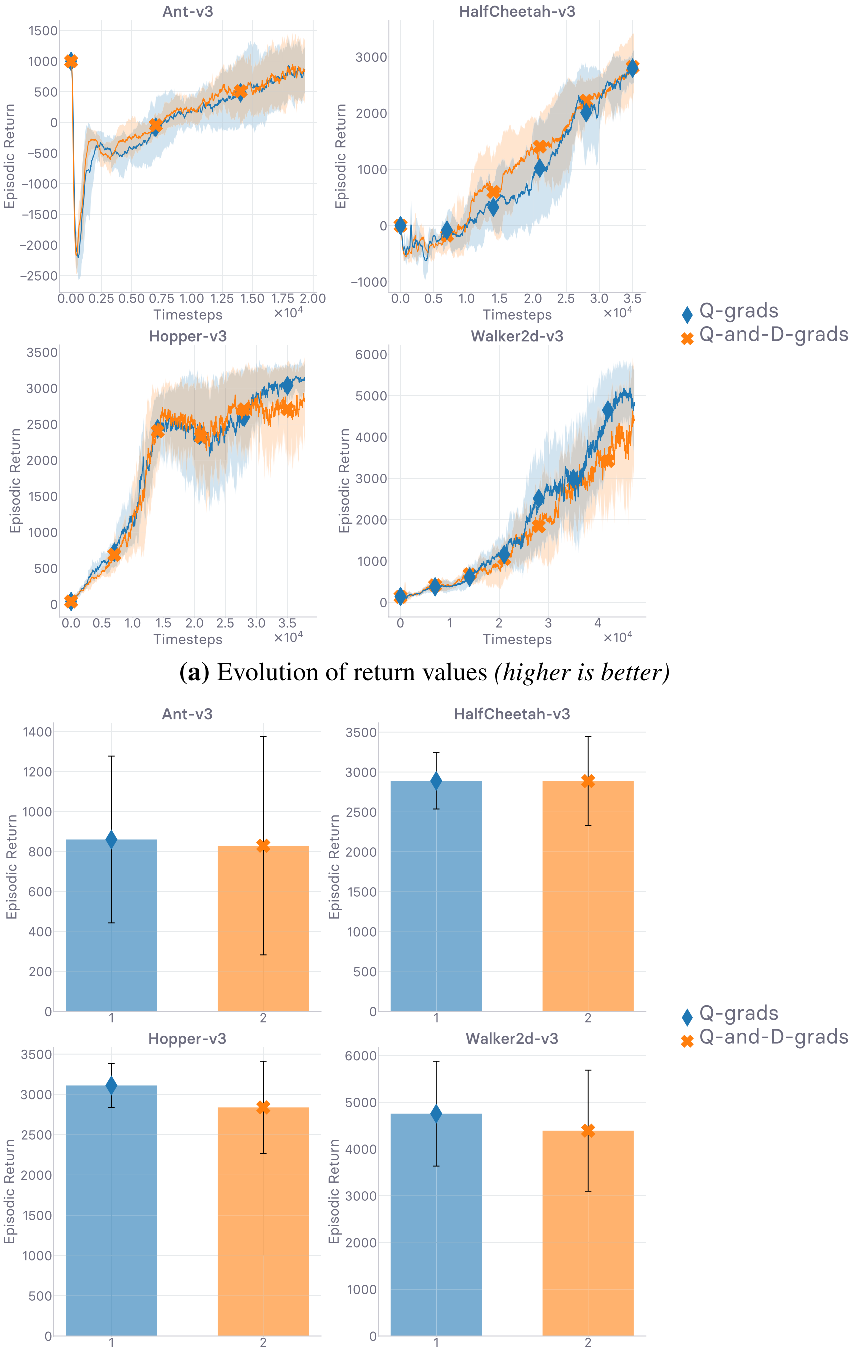
2013, 2015). As for the activations functions used in the neural networks, we used ReLU non-linearities in both the actor and critic, and used Leaky-ReLU (Maas et al. 2013) non-linearities with a leak of 0.1 in the discriminator. We used an *online* version of batch normalization (described earlier in Sect. 5.5) to standardize the actor and critic observations before they are fed to them. We do not use any learning rate scheduler, for any module.

Appendix 2: Sequential decision making under uncertainty in non-stationary Markov decision processes

In Sect. 3, we have defined $\mathbb{M}$ as a stationary MDP, in line with a vast majority of works in RL. Note, a stochastic process or a distribution is commonly said *stationary* if it remains unchanged when shifted in time. While the stationarity assumption allows for the derivation of various theoretical guarantees and is overall easier to deal with analytically, it fails to explain the inner workings of complex realistic simulations, and *a fortiori* the real world. One critical challenge incurred when modeling the world as a non-stationarity MDP is the unavailability of convergence guarantees for standard practical RL methods. Crucially, assuming stationarity in the dynamics p is necessary for the Markov property to hold, which is required for the convergence of Q-learning (Watkins 1989) algorithms (Abdallah and Kaisers 2016) like DQN (Mnih et al. 2013, 2015). As such, designing methods yielding agents that are robust against the non-stationarities naturally occurring in their realistic environments is a challenging yet timely milestone. Methods equipping models against unforeseen changes in the data distribution, a phenomenon qualified as *concept drift* (Schlimmer and Granger 1986), are surveyed in (Gama et al. 2014) who dedicate the study to the supervised case. In RL, a analysis of non-stationarity issues inherent to the Q-learning loss optimization under function approximation (Sutton et al. 1999) proposes qualitative and quantitative diagnostics along with a new replay sampling method to alleviate the isolated weaknesses (Fu et al. 2019). Non-stationarities are characterized by how they manifest in time. A distribution is *switching* if abrupt changes, called *change points*, occur while remaining stationarity in-between, making it in effect piece-wise stationary (Da Silva et al. 2006; Jaksch et al. 2010; Garivier and Moulines 2011; Abdallah and Kaisers 2016; Gajane et al. 2018; Padakandla et al. 2019; Auer et al. 2019). The change points are either given by an oracle or discovered via change point detection techniques. Once exhibited, one can employ stationary methods individually on each segment. A distribution is *drifting* if it gradually changes at an unknown rate (Besbes et al. 2014; Anava and Karnin 2016; Luo et al. 2018; Ortner et al. 2019; Chen et al. 2019; Cheung et al. 2019a, b; Russac et al. 2019). The change can occur continually or as a slow transition between stationary plateaus, making it considerably more difficult to deal with, theoretically and empirically. In a non-stationary MDP, the non-stationarities can manifest in the dynamics p (Nilim and El Ghaoui 2005; Da Silva et al. 2006; Xu and Mannor 2007; Lim et al. 2013; Abdallah and Kaisers 2016), in the reward process r (Even-dar et al. 2005; Dick et al. 2014), or in both conjointly (Yu and Mannor 2009a, b; Abbasi-Yadkori et al. 2013; Gajane et al. 2018; Padakandla et al. 2019; Yu and Sra 2019; Lecarpentier et al. 2019). The adversarial bilevel optimization problem—guiding the adaptive tuning of the reward for every policy update—present in this work is reminiscent of the stream of research pioneered by Auer et al. (1995) in which the reward is generated by an omniscient *adversary*, either arbitrarily or adaptively with potentially malevolent drive (Yu and Mannor 2009a, b; Lim et al. 2013; Gajane et al. 2018; Yu and Sra 2019). Non-stationary environments are almost exclusively tackled from a theoretical perspective in the literature (cf. previous references in this section). Specifically, in the *drifting* case, the non-stationarities are traditionally dealt with via the use of sliding windows. The accompanying (dynamic) regret analyses all rely on strict assumptions. In the switching case, one needs to know the number of occurring switches beforehand, while in the drifting case, the change variation need be upper-bounded. Specifically, (Besbes et al. 2014; Cheung et al. 2019a) assume the total change to be upper-bounded by some preset variation budget, while (Cheung et al. 2019b) assumes the variations are uniformly bounded in time. (Ortner et al. 2019) assumes that the *incremental* variation [as opposed to *total* in (Besbes et al. 2014; Cheung et al. 2019a)] is upper-bounded by a *per-change* threshold. Finally, in the same vein, (Lecarpentier et al. 2019) posits *regular evolution*, by making the assumption that both the transition and reward functions are Lipschitz-continuous *w.r.t.* time.

Appendix 3: Adaptive Policy Update based on Gradient Similarities

See Appendix Fig. 12.



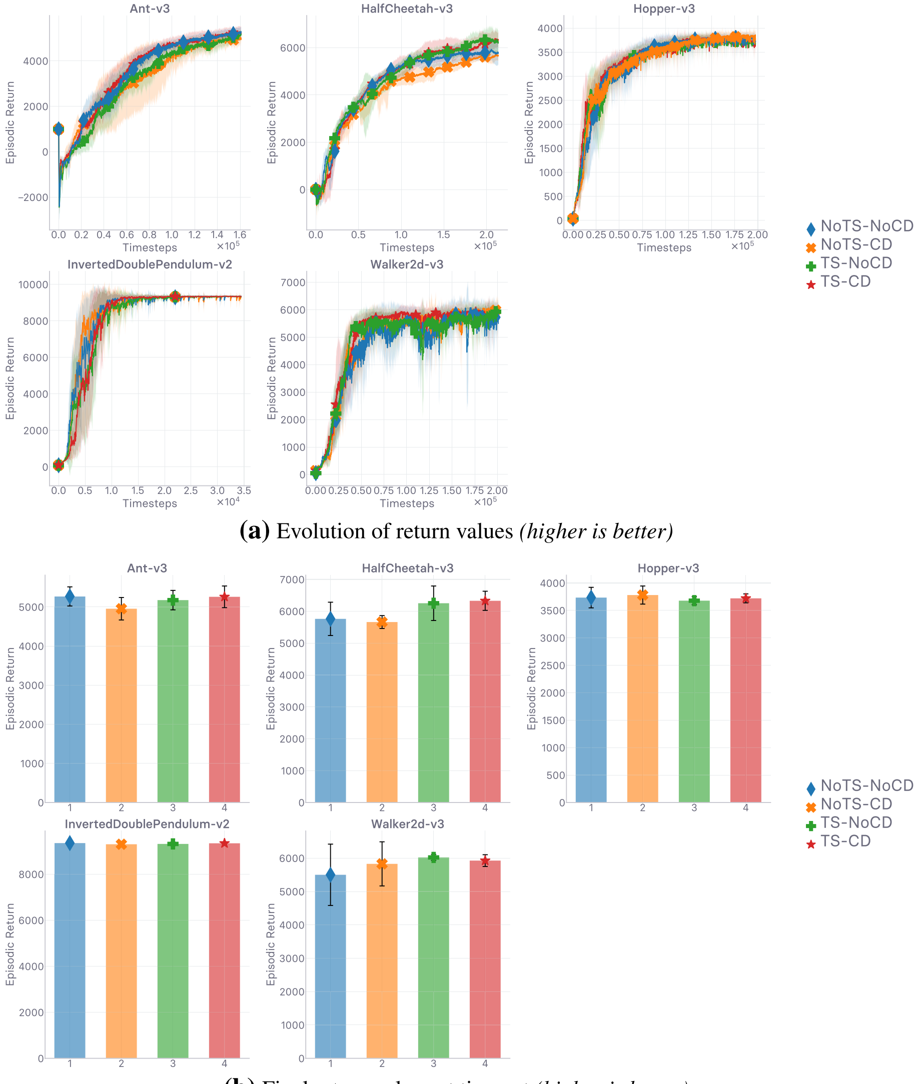
(a) Evolution of return values *(higher is better)*

(b) Final return values at timeout *(higher is better)*

Fig. 12 Comparison of the gradient used to update the policy in this work, involving the gradient of the state-action value, against an adaptive hybrid method involving *also* the gradient of the discriminator, and combining both gradients based on their cosine similarity. Runtime is 12 h

Appendix 4: Clipped double-Q learning and target policy smoothing

See Appendix Fig. 13.



(a) Evolution of return values *(higher is better)*

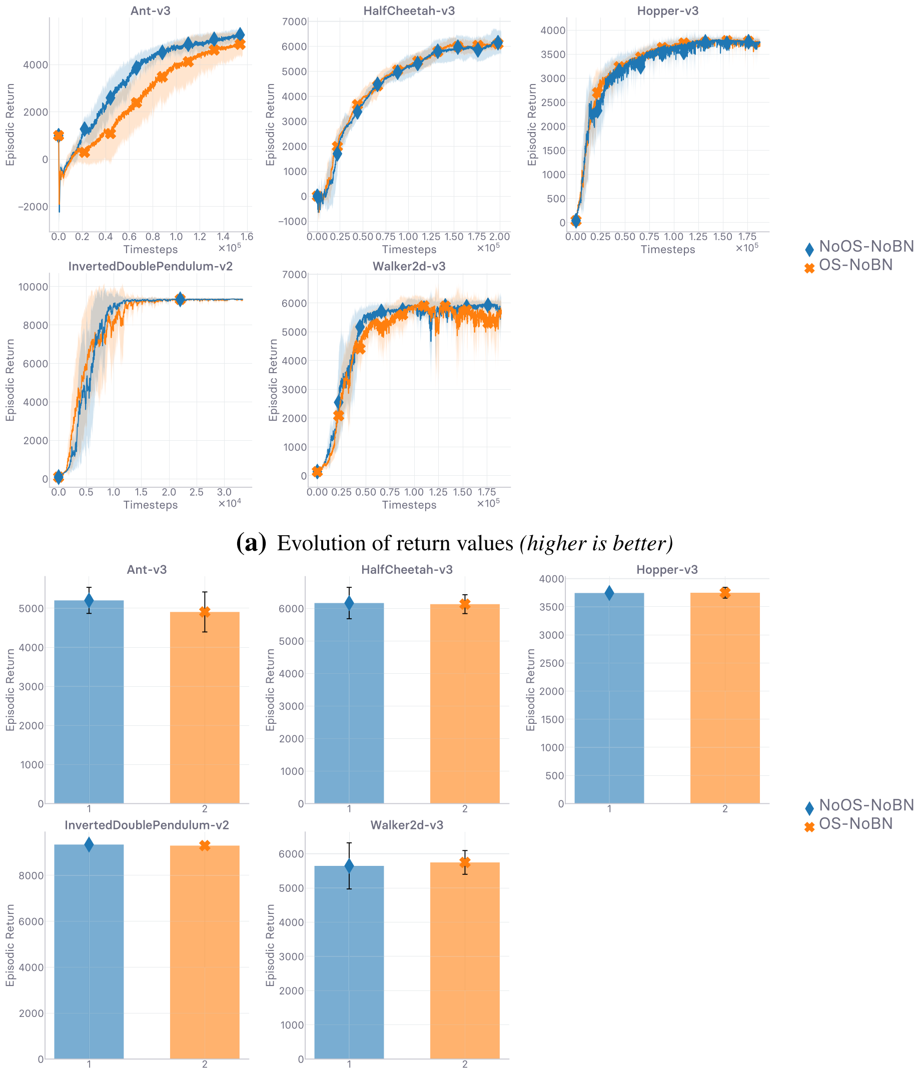
(b) Final return values at timeout *(higher is better)*

Fig. 13 Ablation study on the use of the clipped double Q-Learning (CD) and target smoothing (TS) techniques, both from (Fujimoto et al. 2018), *with* gradient penalty regularization (Gulrajani et al. 2017). Runtime is 48 h

Appendix 5: Gradient penalty

5.1 One-sided gradient penalty

See Appendix Fig. 14.



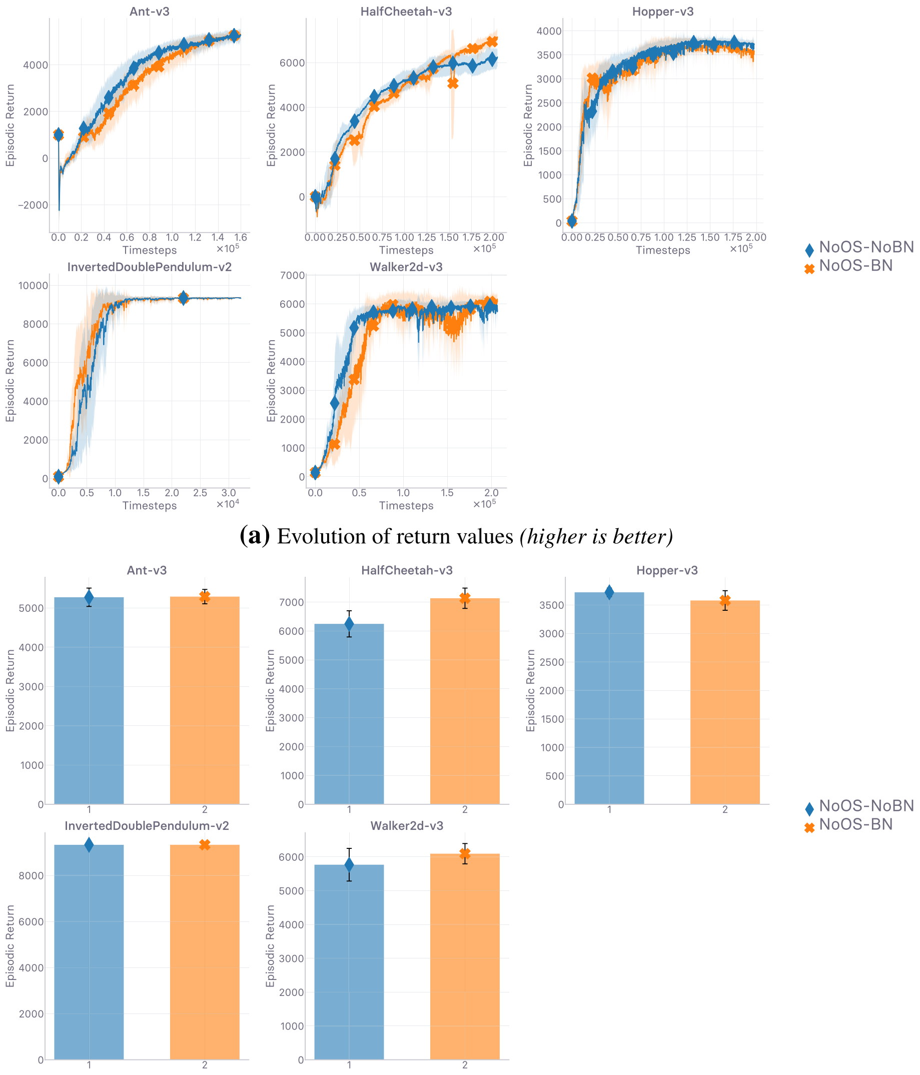
(a) Evolution of return values *(higher is better)*

(b) Final return values at timeout *(higher is better)*

Fig. 14 Ablation study on the use of the one-sided (OS) penalty variant (Gulrajani et al. 2017). Runtime is 48 h

5.2 Online batch normalization in discriminator

See Appendix Fig. 15.



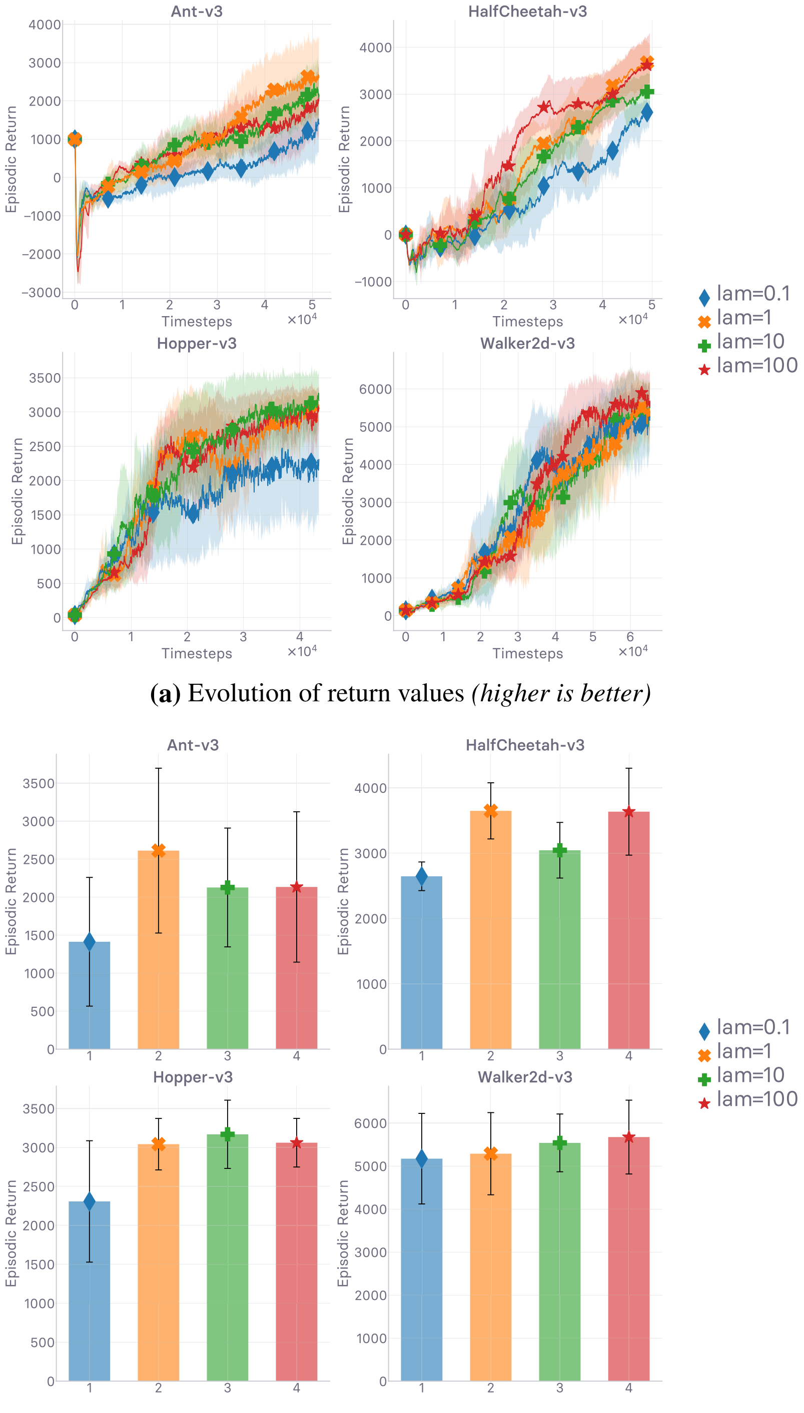
(a) Evolution of return values *(higher is better)*

(b) Final return values at timeout *(higher is better)*

Fig. 15 Ablation study on the use of online batch normalization (BN) in the discriminator for its impact on the gradient penalization (Gulrajani et al. 2017). Runtime is 48 hours

5.3 Target *k* and coefficient *λ* grid search

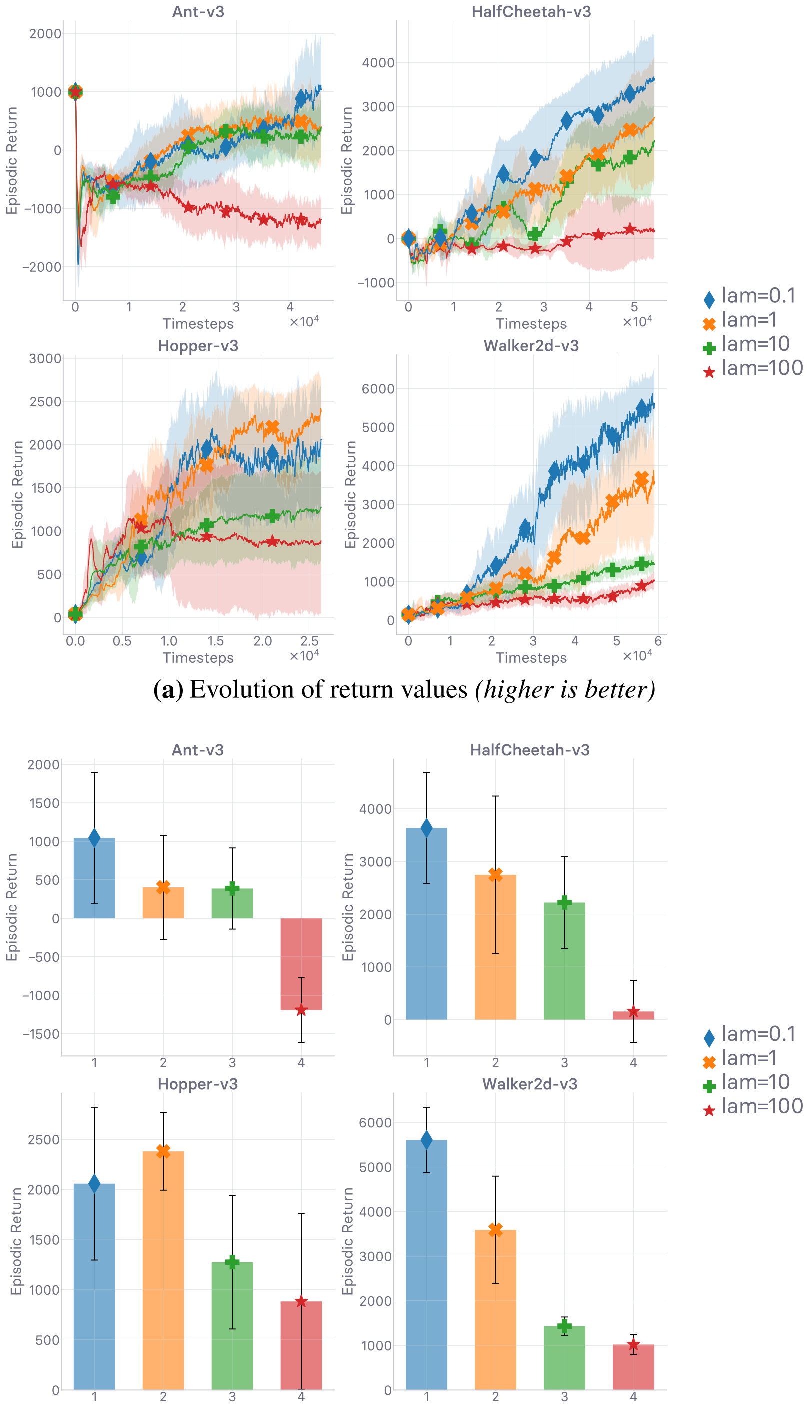
See Appendix Figs. 16 and 17.



(a) Evolution of return values *(higher is better)*

(b) Final return values at timeout *(higher is better)*

Fig. 16 Grid search over the hyper-parameter λ when $k = 1$. Runtime is 12 h



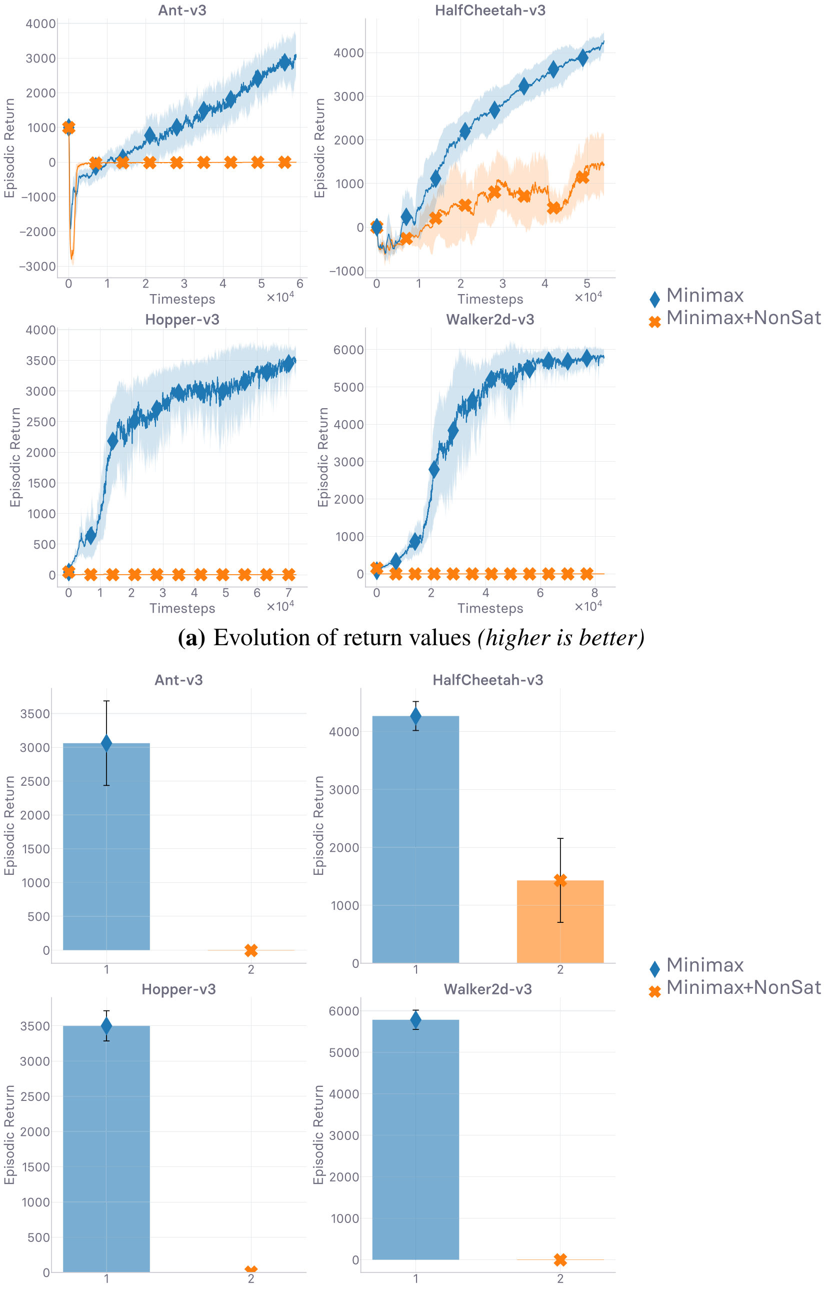
(a) Evolution of return values *(higher is better)*

(b) Final return values at timeout *(higher is better)*

Fig. 17 Grid search over the hyper-parameter λ when $k = 0$. Runtime is 12 h

Appendix 6: Reward formulation

See Appendix Fig. 18.



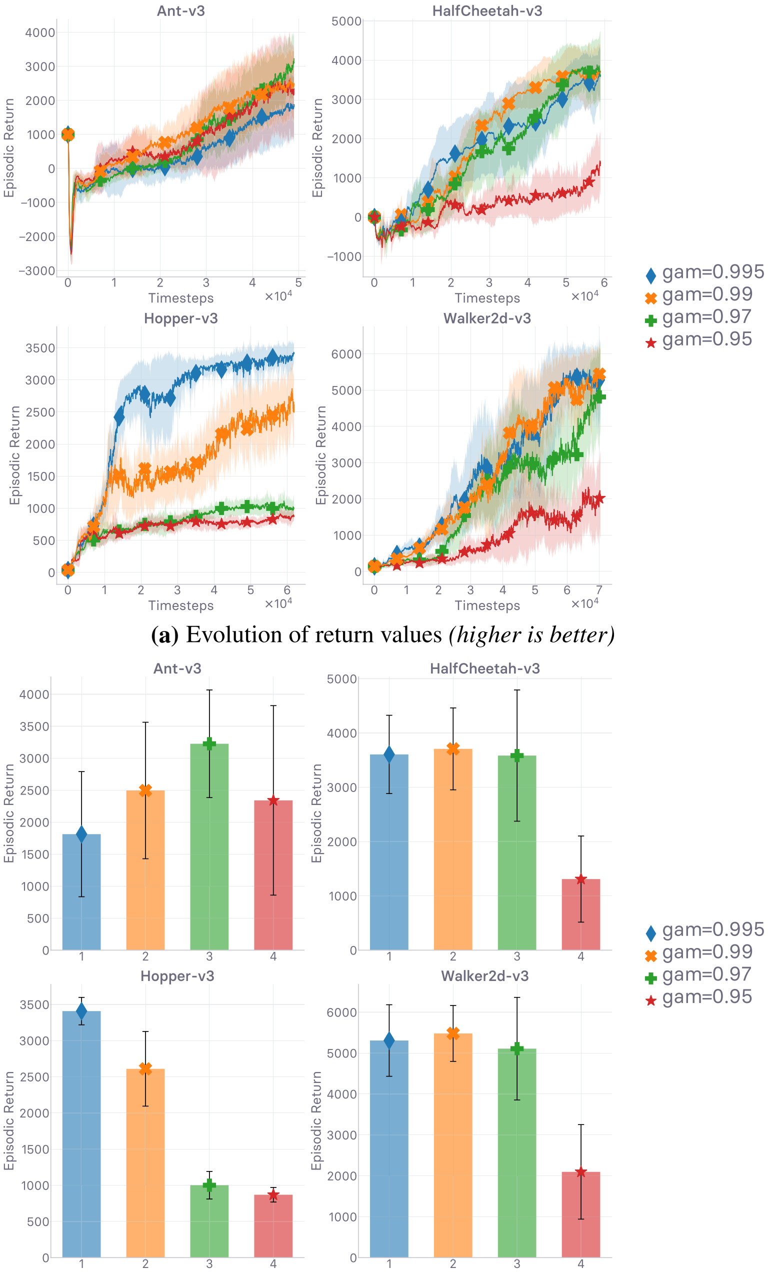
(a) Evolution of return values *(higher is better)*

(b) Final return values at timeout *(higher is better)*

Fig. 18 Comparison of two ways to define the surrogate imitation reward r_φ from the discriminator D_φ. *"Minimax"* corresponds to $r_\varphi^{\mathrm{MM}} := -\log(1 - D_\varphi)$, while *"Minimax + Non-Saturating"* denotes the use of $r_\varphi^{\mathrm{NS}} := -\log(1 - D_\varphi) + \log(D_\varphi)$, as described in Sect. 4. Runtime is 12 h

Appendix 7: Discount factor

See Appendix Fig. 19.



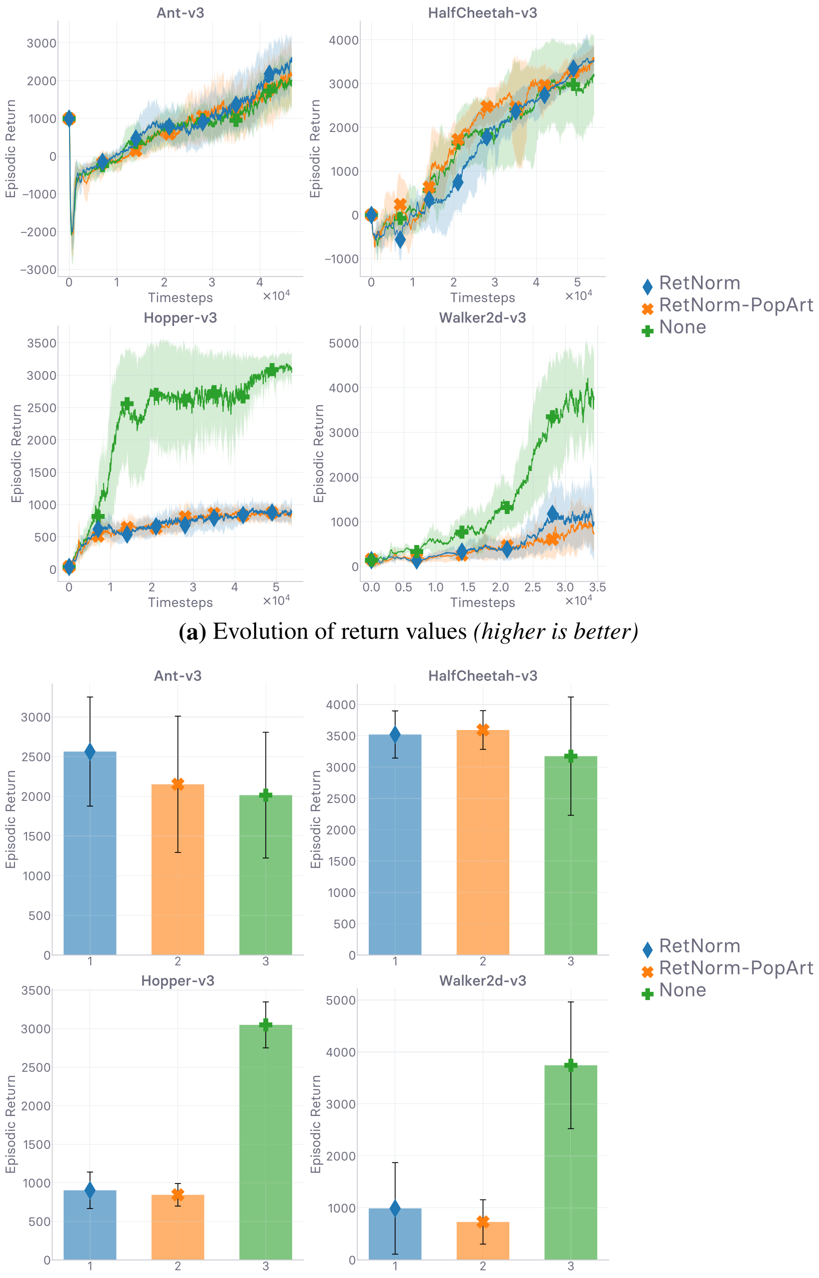
(a) Evolution of return values *(higher is better)*

(b) Final return values at timeout *(higher is better)*

Fig. 19 Grid search over the discount factor γ. Runtime is 12 h

Appendix 8: Return normalization

See Appendix Fig. 20.



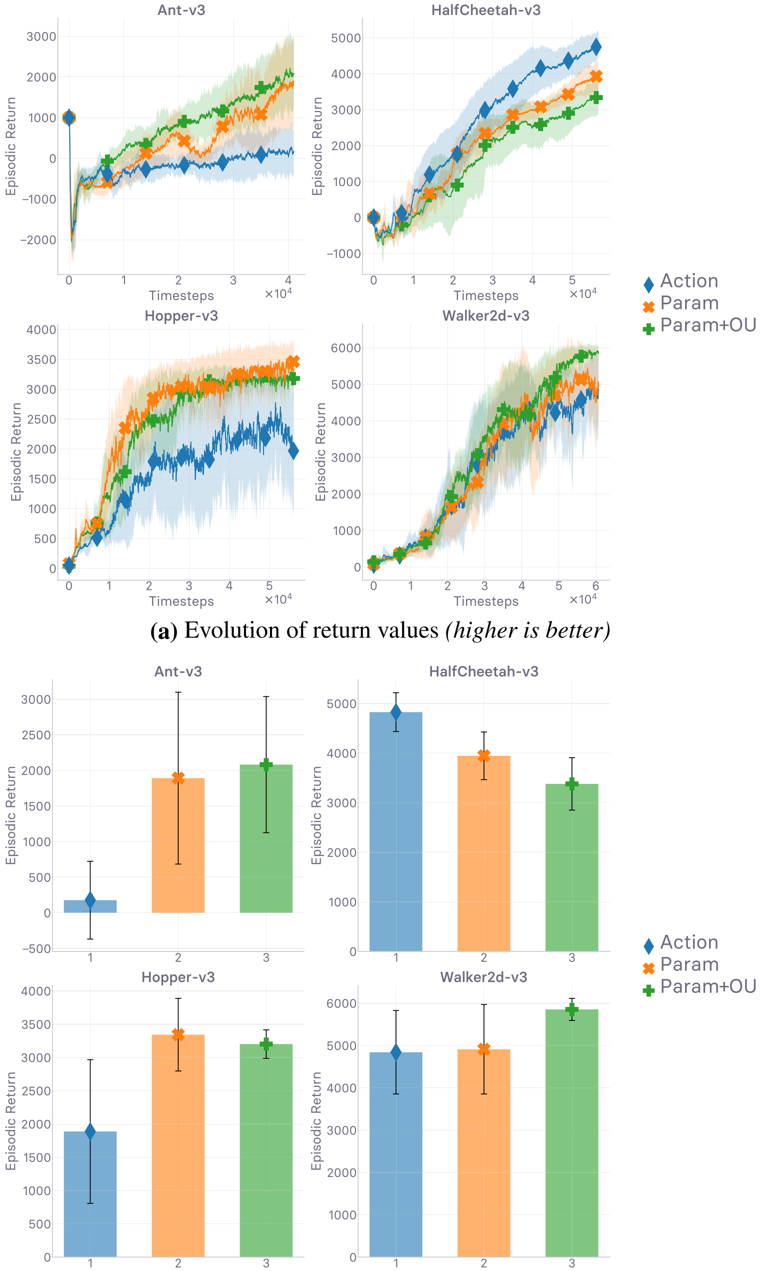
(a) Evolution of return values *(higher is better)*

(b) Final return values at timeout *(higher is better)*

Fig. 20 Ablation study on return normalization and Pop-Art (van Hasselt et al. 2016). Runtime is 12 h

Appendix 9: Exploration

See Appendix Fig. 21.



(a) Evolution of return values *(higher is better)*

(b) Final return values at timeout *(higher is better)*

◀**Fig. 21** Evaluation of the considered method under several exploration strategies. *"Action"* corresponds to defining π_θ by directly applying additive Gaussian noise to the *action* returned by μ_θ. As such, $\pi_\theta(\cdot, s_t) = \mu_\theta(s_t) + \epsilon$, where $\epsilon \sim \mathcal{N}(0, \sigma)$, with $\sigma = 0.2$. *"Param"* denotes the application of additive noise in the network *parameters* directly, and *"Param + OU"* corresponds to the additional application of temporally correlated noise, generated sequentially by a Ornstein-Uhlenbeck process, on the action (cf. Sect. 4 for a description of these two last approaches, and Table 2 for the associated hyper-parameters). Despite the absence of a clear winner, we use the combination of parameter noise and temporally correlated action noise in every experiment reported in this work, as it seems to yield the best results. Runtime is 12 h

Author contributions Conceptualization: Lionel Blondé; Methodology: Lionel Blondé; Formal analysis and investigation: Lionel Blondé; Writing—original draft preparation: Lionel Blondé; Writing—review and editing: Lionel Blondé, Pablo Strasser, Alexandros Kalousis; Funding acquisition: Alexandros Kalousis; Resources: Lionel Blondé, Alexandros Kalousis; Supervision: Alexandros Kalousis.

Funding Open access funding provided by University of Geneva. This work was supported by the Swiss National Science Foundation grant number CSSII5_177179 "Modeling pathological gait resulting from motor impairment".

Availability of data and materials The simulated robotics, continuous control environments considered in this work are built with the MuJoCo (Todorov et al. 2012) physics engine, and provided to the community through the OpenAI Gym API (Brockman et al. 2016). Note, to use these environments, one needs a MuJoCo license, which can be obtained from https://www.roboti.us/license.html.

Code availability SAM (Blondé and Kalousis 2019) implementations: https://github.com/lionelblonde/sam-tf (TensorFlow), https://github.com/lionelblonde/sam-pytorch (PyTorch). SAM augmented with the extensions implemented in this work: https://github.com/lionelblonde/liayn-pytorch (PyTorch).

Declarations

Conflict of interest The authors declare that they have no competing interests.

References

Abbasi-Yadkori, Y., Bartlett, P. L., & Szepesvari, C. (2013). Online learning in Markov decision processes with adversarially chosen transition probability distributions. Preprint retrieved from arXiv:1303.3055

Abbeel, P., & Ng, A. Y. (2004). Apprenticeship learning via inverse reinforcement learning. In *It international conference on machine learning (ICML)*. https://doi.org/10.1145/1015330.1015430

Abdallah, S., & Kaisers, M. (2016). Addressing environment non-stationarity by repeating Q-learning updates. *Journal of Machine Learning Research (JMLR), 17*(1), 1582–1612. https://doi.org/10.5555/2946645.2946691

Achiam, J., Knight, E., & Abbeel, P. (2019). Towards characterizing divergence in deep q-learning. Preprint retrieved from arXiv:1903.08894

Anava, O., & Karnin, Z. (2016). Multi-armed Bandits: Competing with optimal sequences. In *Neural information processing systems (NeurIPS)*. https://papers.nips.cc/paper/6341-multi-armed-bandits-competing-with-optimal-sequences.pdf

Arjovsky, M., & Bottou, L. (2017). Towards principled methods for training generative adversarial networks. In *International conference on learning representations (ICLR)*. Preprint retrieved from arXiv:1701.04862

Arjovsky, M., Chintala, S., & Bottou, L. (2017). Wasserstein GAN. Preprint retrieved from arXiv:1701.07875

Atkeson, C. G., & Schaal, S. (1997). Robot learning from demonstration. In *International conference on machine learning (ICML)* (Vol. 97, pp. 12–20). https://pdfs.semanticscholar.org/a0e2/dd2f6b116a12b235caa9b8ef8960f7afc585.pdf

Auer, P., Cesa-Bianchi, N., Freund, Y., & Schapire, R. E. (1995). The non-stochastic multi-armed bandit problem. *Symposium on Foundations of Computer Science*. https://cseweb.ucsd.edu/~yfreund/papers/bandits.pdf

Auer, P., Gajane, P., & Ortner, R. (2019). Adaptively tracking the best bandit arm with an unknown number of distribution changes. In *Conference on learning theory (COLT)*. http://proceedings.mlr.press/v99/auer19a/auer19a.pdf

Ba, J. L., Kiros, J. R., & Hinton, G. E. (2016). Layer normalization. Preprint retrieved from arXiv:1607.06450

Bagnell, J. A. (2015). An invitation to imitation. Tech. rep., Carnegie Mellon, Robotics Institute. https://www.ri.cmu.edu/pub_files/2015/3/InvitationToImitation_3_1415.pdf

Baram, N., Anschel, O., Caspi, I., & Mannor, S. (2017). End-to-end differentiable adversarial imitation learning. In *International conference on machine learning (ICML)* (pp. 390–399). http://proceedings.mlr.press/v70/baram17a/baram17a.pdf

Besbes, O., Gur, Y., & Zeevi, A. (2014). Stochastic multi-armed-bandit problem with non-stationary rewards. In *Neural information processing systems (NeurIPS)*. http://papers.nips.cc/paper/5378-stochastic-multi-armed-bandit-problem-with-non-stationary-rewards.pdf

Biewald, L. (2020). Experiment tracking with weights and biases. https://www.wandb.com/

Billard, A., Calinon, S., Dillmann, R., & Schaal, S. (2008). Robot programming by demonstration. In Bruno, S., & Oussama, K. (Eds.) *Springer handbook of robotics* (pp. 1371–1394). Springer. http://link.springer.com/referenceworkentry/10.1007/978-3-540-30301-5_60

Bishop, C. M. (1995). Training with noise is equivalent to Tikhonov regularization. *Neural Computation, 7*(1), 108–116. https://doi.org/10.1162/neco.1995.7.1.108

Blondé, L., & Kalousis, A. (2019). Sample-efficient imitation learning via generative adversarial nets. In *International conference on artificial intelligence and statistics (AISTATS)*. Preprint retrieved from arXiv:1809.02064

Borsa, D., Piot, B., Munos, R., & Pietquin, O. (2017). Observational learning by reinforcement learning. In *Neural information processing systems (NeurIPS)*. Preprint retrieved from arXiv:1706.06617

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., & Zaremba, W. (2016). OpenAI Gym. Preprint retrieved from arXiv:1606.01540

Burda, Y., Edwards, H., Storkey, A., & Klimov, O. (2018). Exploration by random network distillation. Preprint retrieved from arXiv:1810.12894

Carpenter, G. A., & Grossberg, S. (1987). A massively parallel architecture for a self-organizing neural pattern recognition machine. *Computer Vision, Graphics, and Image Processing*, *37*(1), 54–115. http://techlab.bu.edu/files/resources/articles_cns/CarpenterGrossberg1987.pdf

Chen, Y., Lee, C. W., Luo, H., & Wei, C. Y. (2019). A new algorithm for non-stationary contextual bandits: Efficient, optimal, and parameter-free. In *Conference on learning theory (COLT)*. Preprint retrieved from arXiv:1902.00980

Cheung, W. C., Simchi-Levi, D., & Zhu, R. (2019). Learning to optimize under non-stationarity. In *International conference on artificial intelligence and statistics (AISTATS)*. Preprint retrieved from arXiv:1810.03024

Cheung, W. C., Simchi-Levi, D., & Zhu, R. (2019). Reinforcement learning under drift. Preprint retrieved from arXiv:1906.02922

Cisse, M., Bojanowski, P., Grave, E., Dauphin, Y., & Usunier, N. (2017). Parseval networks: Improving robustness to adversarial examples. Preprint retrieved from arXiv:1704.08847

Da Silva, B. C., Basso, E. W., Bazzan, A. L. C., & Engel, P. M. (2006). Dealing with non-stationary environments using context detection. In *International conference on machine learning (ICML)*. https://dl.acm.org/citation.cfm?id=1143872

Dick, T., Gyorgy, A., & Szepesvari, C. (2014). Online learning in Markov decision processes with changing cost sequences. In *International conference on machine learning (ICML)*. http://proceedings.mlr.press/v32/dick14.html

Dinh, L., Pascanu, R., Bengio, S., & Bengio, Y. (2017). Sharp minima can generalize for deep nets. In *International conference on machine learning (ICML)*. Preprint retrieved from arXiv:1703.04933

Doersch, C., Gupta, A., Efros, A. A. (2015). Unsupervised visual representation learning by context prediction. In *International conference on computer vision (ICCV)*. Preprint retrieved from arXiv:1505.05192

Du, Y., Czarnecki, W. M., Jayakumar, S. M., Pascanu, R., & Lakshminarayanan, B. (2018). Adapting auxiliary losses using gradient similarity. Preprint retrieved from arXiv:1812.02224

Duan, Y., Andrychowicz, M., Stadie, B. C., Ho, J., Schneider, J., Sutskever, I., Abbeel, P., & Zaremba, W. (2017). One-shot imitation learning. In *Neural Information Processing Systems (NeurIPS)*. Preprint retrieved from arXiv:1703.07326

Even-dar, E., Kakade, S. M., & Mansour, Y. (2005). Experts in a Markov decision process. In *Neural Information Processing Systems (NeurIPS)*. http://papers.nips.cc/paper/2730-experts-in-a-markov-decision-process.pdf

Everitt, T., Krakovna, V., Orseau, L., Hutter, M., & Legg, S. (2017). Reinforcement learning with a corrupted reward channel. In *International joint conference on artificial intelligence (IJCAI)*. Preprint retrieved from arXiv:1705.08417.

Fernando Hernandez-Garcia, J., & Sutton, R. S. (2019). Understanding multi-step deep reinforcement learning: A systematic study of the DQN target. Preprint retrieved from arXiv:1901.07510

Finlay, C., Calder, J., Abbasi, B., & Oberman, A. (2018). Lipschitz regularized Deep Neural Networks generalize and are adversarially robust. Preprint retrieved from arXiv:1808.09540

Fonteneau, R., Murphy, S. A., Wehenkel, L., & Ernst, D. (2010). Model-free Monte Carlo–like policy evaluation. In *International conference on artificial intelligence and statistics (AISTATS)*. http://proceedings.mlr.press/v9/fonteneau10a/fonteneau10a.pdf

Fonteneau, R., Murphy, S. A., Wehenkel, L., & Ernst, D. (2013). Batch mode reinforcement learning based on the synthesis of artificial trajectories. *Annals of Operations Research*, *208*(1), 383–416. http://dx.doi.org/10.1007/s10479-012-1248-5

Fortunato, M., Azar, M. G., Piot, B., Menick, J., Osband, I., Graves, A., Mnih, V., Munos, R., Hassabis, D., Pietquin, O., Blundell, C., & Legg, S. (2017). Noisy networks for exploration. Preprint retrieved from arXiv:1706.10295

Fu, J., Kumar, A., Soh, M., & Levine, S. (2019). Diagnosing bottlenecks in deep Q-learning algorithms. In *International conference on machine learning (ICML)*. Preprint retrieved from arXiv:1902.10250

Fujimoto, S., van Hoof, H., & Meger, D. (2018). Addressing function approximation error in actor-critic methods. In *International conference on machine learning (ICML)*. Preprint retrieved from arXiv:1802.09477

Gajane, P., Ortner, R., & Auer, P. (2018). A sliding-window algorithm for Markov decision processes with arbitrarily changing rewards and transitions. Preprint retrieved from arXiv:1805.10066

Gama, J., Žliobaitė, I., Bifet, A., Pechenizkiy, M., & Bouchachia, A. (2014). A survey on concept drift adaptation. *ACM Computing Surveys (CSUR)*. https://www.win.tue.nl/~mpechen/publications/pubs/Gama_ACMCS_AdaptationCD_accepted.pdf

Garivier, A., & Moulines, E. (2011). On upper-confidence bound policies for switching bandit problems. In *Algorithmic learning theory (ALT)* (pp. 174–188). Springer. http://dx.doi.org/10.1007/978-3-642-24412-4_16

Goodfellow, I. (2017). NIPS 2016 tutorial: Generative adversarial networks. Preprint retrieved from arXiv:1701.00160

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., & Bengio, Y. (2014). Generative adversarial nets. In *Neural information processing systems (NIPS)*. http://papers.nips.cc/paper/5423-generative-adversarial-nets.pdf

Gouk, H., Frank, E., Pfahringer, B., & Cree, M. J. (2021). Regularisation of neural networks by enforcing Lipschitz continuity. *Machine Learning*, *110*(2), 393–416. https://doi.org/10.1007/s10994-020-05929-w

Gu, Z., Li, Z., Di, X., & Shi, R. (2020). An LSTM-based autonomous driving model using Waymo open dataset. Preprint retrieved from arXiv:2002.05878

Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., & Courville, A. (2017). Improved training of Wasserstein GANs. In *Neural information processing systems (NIPS)*. Preprint retrieved from arXiv:1704.00028v3

Ha, S., Xu, P., Tan, Z., Levine, S., & Tan, J. (2020). Learning to walk in the real world with minimal human effort. Preprint retrieved from arXiv:2002.08550

Hafner, R., & Riedmiller, M. (2011). Reinforcement learning in feedback control. *Machine Learning*, *84*(1–2), 137–169. https://link.springer.com/article/10.1007/s10994-011-5235-x

Hanna, J. P., & Stone, P. (2017). Grounded action transformation for robot learning in simulation. In *AAAI conference on artificial intelligence*. https://www.cs.utexas.edu/~jphanna/gsl.pdf

Harada, D. (1997). Reinforcement learning with time. In *Conference on artificial intelligence (AAAI)*. https://www.aaai.org/Papers/AAAI/1997/AAAI97-090.pdf

Hardt, M., Recht, B., & Singer, Y. (2015). Train faster, generalize better: Stability of stochastic gradient descent. Preprint retrieved from arXiv:1509.01240

Heckman, J. J. (1979). Sample selection bias as a specification error. *Econometrica*, **47**(1), 153–161. http://www.jstor.org/stable/1912352

Held, D., McCarthy, Z., Zhang, M., Shentu, F., & Abbeel, P. (2017). Probabilistically safe policy transfer. Preprint retrieved from arXiv:1705.05394

Hernandez, D., & Brown, T. B. (2020). Measuring the algorithmic efficiency of neural networks. https://cdn.openai.com/papers/ai_and_efficiency.pdf

Hessel, M., Modayil, J., van Hasselt, H., Schaul, T., Ostrovski, G., Dabney, W., Horgan, D., Piot, B., Azar, M., & Silver, D. (2017). Rainbow: Combining improvements in deep reinforcement learning. Preprint retrieved from arXiv:1710.02298

Ho, J., & Ermon, S. (2016). Generative adversarial imitation learning. In *Neural information processing systems (NIPS)*. Preprint retrieved from arXiv:1606.03476

Ho, J., Gupta, J. K., & Ermon, S. (2016). Model-free imitation learning with policy optimization. In *International conference on machine learning (ICML)*. Preprint retrieved from arXiv:1605.08478

Hochreiter, S., & Schmidhuber, J. (1997). Flat minima. *Neural Computation*, *9*(1), 1–42. https://www.ncbi.nlm.nih.gov/pubmed/9117894

Ioffe, S., & Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. Preprint retrieved from arXiv:1502.03167

Jaderberg, M., Mnih, V., Czarnecki, W. M., Schaul, T., Leibo, J. Z., Silver, D., & Kavukcuoglu, K. (2016). Reinforcement learning with unsupervised auxiliary tasks. Preprint retrieved from arXiv:1611.05397

Jaksch, T., Ortner, R., & Auer, P. (2010). Near-optimal regret bounds for reinforcement learning. *Journal of Machine Learning Research (JMLR)*, *11*, 1563–1600 . http://www.jmlr.org/papers/volume11/jaksch10a/jaksch10a.pdf

Kaelbling, L. P. (1993). Learning to achieve goals. In *International joint conference on artificial intelligence (IJCAI)*. https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.51.3077&rep=rep1&type=pdf

Kahn, G., Zhang, T., Levine, S., & Abbeel, P. (2016). PLATO: Policy learning using adaptive trajectory optimization. Preprint retrieved from arXiv:1603.00622

Karpathy, A., & Van De Panne, M. (2012). Curriculum learning for motor skills. In *Advances in artificial intelligence (Canadian conference on artificial intelligence)* (pp. 325–330). http://link.springer.com/content/pdf/10.1007/978-3-642-30353-1.pdf#page=339

Keskar, N. S., Mudigere, D., Nocedal, J., Smelyanskiy, M., & Tang, P. T. P. (2017). On large-batch training for deep learning: Generalization gap and sharp minima. In *International conference on learning representations (ICLR)*. Preprint retrieved from arXiv:1609.04836

Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. Preprint retrieved from arXiv:1412.6980

Kodali, N., Abernethy, J., Hays, J., & Kira, Z. (2017). How to train your DRAGAN. Preprint retrieved from arXiv:1705.07215

Kostrikov, I., Agrawal, K. K., Dwibedi, D., Levine, S., & Tompson, J. (2019). Discriminator-actor-critic: Addressing sample inefficiency and reward bias in adversarial imitation learning. In *International conference on learning representations (ICLR)*. Preprint retrieved from arXiv:1809.02925

Kurach, K., Lucic, M., Zhai, X., Michalski, M., & Gelly, S. (2018). The GAN landscape: Losses, architectures, regularization, and normalization. Preprint retrieved from arXiv:1807.04720

Lange, S., Gabel, T., & Riedmiller, M. (2012). Batch reinforcement learning. In Wiering, M., & van Otterlo, M. (Eds.) *Reinforcement learning: State-of-the-art* (pp. 45–73). Springer. https://doi.org/10.1007/978-3-642-27645-3_2

Lecarpentier, E., & Rachelson, E. (2019). Non-stationary Markov decision processes, a worst-case approach using model-based reinforcement learning. In *Neural information processing systems (NeurIPS)*. Preprint retrieved from arXiv:1904.10090

Li, H., Xu, Z., Taylor, G., Studer, C., & Goldstein, T. (2018). Visualizing the loss landscape of neural nets. In *Neural information processing systems (NeurIPS)*. Preprint retrieved from arXiv:1712.09913

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., & Wierstra, D. (2016). Continuous control with deep reinforcement learning. In *International conference on learning representations (ICLR)*. Preprint retrieved from arXiv:1509.02971

Lim, S.H., Xu, H., & Mannor, S. (2013). Reinforcement learning in robust Markov decision processes. In *Neural information processing systems (NeurIPS)*. http://papers.nips.cc/paper/5183-reinforcement-learning-in-robust-markov-decision-processes.pdf

Lin, L. J. (1992). Self-improving reactive agents based on reinforcement learning, planning and teaching. *Machine Learning, 8*(3), 293–321. https://doi.org/10.1007/BF00992699

Loshchilov, I., & Hutter, F. (2017). Fixing weight decay regularization in Adam. Preprint retrieved from arXiv:1711.05101

Lucic, M., Kurach, K., Michalski, M., Gelly, S., & Bousquet, O. (2017). Are GANs created equal? A large-scale study. Preprint retrieved from arXiv:1711.10337

Luo, H., Wei, C. Y., Agarwal, A., & Langford, J. (2018). Efficient contextual bandits in non-stationary worlds. In *Conference on learning theory (COLT)*. Preprint retrieved from arXiv:1708.01799

Maas, A. L., Hannun, A. Y., & Ng, A. Y. (2013). Rectifier nonlinearities improve neural network acoustic models. In *International conference on machine learning (ICML)*. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.693.1422&rep=rep1&type=pdf

Mescheder, L., Geiger, A., & Nowozin, S. (2018). Which training methods for GANs do actually converge? In *International conference on machine learning (ICML)*. Preprint retrieved from arXiv:1801.04406

Mirowski, P., Pascanu, R., Viola, F., Soyer, H., Ballard, A. J., Banino, A., Denil, M., Goroshin, R., Sifre, L., Kavukcuoglu, K., Kumaran, D., & Hadsell, R. (2016). Learning to navigate in complex environments. Preprint retrieved from arXiv:1611.03673

Miyato, T., Kataoka, T., Koyama, M., & Yoshida, Y. (2018). Spectral normalization for generative adversarial networks. In *International conference on learning representations (ICLR)*. https://openreview.net/pdf?id=B1QRgziT-

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., & Riedmiller, M. (2013). Playing Atari with deep reinforcement learning. Preprint retrieved from arXiv:1312.5602

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., & Hassabis, D. (2015). Human-level control through deep reinforcement learning. *Nature, 518*(7540), 529–533. https://doi.org/10.1038/nature14236

Müller, R., Kornblith, S., & Hinton, G. E. (2019). When does label smoothing help? In *Neural information processing systems (NeurIPS)*. http://papers.nips.cc/paper/8717-when-does-label-smoothing-help.pdf

Ng, A. Y., Harada, D., & Russell, S. (1999). Policy invariance under reward transformations: Theory and application to reward shaping. In *International conference on machine learning (ICML)* (pp. 278–287). http://www.robotics.stanford.edu/~ang/papers/shaping-icml99.pdf

Ng, A. Y., & Russell, S. J. (2000). Algorithms for inverse reinforcement learning. In *International conference on machine learning (ICML)* (pp. 663–670). http://ai.stanford.edu/~ang/papers/icml00-irl.pdf

Nilim, A., & El Ghaoui, L. (2005). Robust control of Markov decision processes with uncertain transition matrices. *Operations Research, 53*(5), 780–798. https://doi.org/10.1287/opre.1050.0216

OpenAI: Solving Rubik's cube with a robot Hand (2019). https://d4mucfpksywv.cloudfront.net/papers/solving-rubiks-cube.pdf

Orsini, M., Raichuk, A., Hussenot, L., Vincent, D., Dadashi, R., Girgin, S., Geist, M., Bachem, O., Pietquin, O., & Andrychowicz, M. (2021). What matters for adversarial imitation learning? Preprint retrieved from arXiv:2106.00672

Ortner, R., Gajane, P., & Auer, P. (2019). Variational regret bounds for reinforcement learning. Preprint retrieved from arXiv:1905.05857

Padakandla, S., Prabuchandran, K. J., & Bhatnagar, S. (2019). Reinforcement learning in non-stationary environments. Preprint retrieved from arXiv:1905.03970

Pardo, F., Tavakoli, A., Levdik, V., & Kormushev, P. (2018). Time limits in reinforcement learning. In *International conference on machine learning (ICML)*. Preprint retrieved from arXiv:1712.00378

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., & Chintala, S. (2019). PyTorch: An imperative style, high-performance deep learning library. In *Neural information processing systems (NeurIPS)*. http://papers.nips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf

Peng, J., & Williams, R. J. (1996). Incremental multi-step Q-learning. *Machine Learning, 22*(1-3), 283–290. https://link.springer.com/article/10.1023/A:1018076709321

Peng, X. B., Kanazawa, A., Toyer, S., Abbeel, P., & Levine, S. (2018). Variational discriminator bottleneck: Improving imitation learning, inverse RL, and GANs by constraining information flow. Preprint retrieved from arXiv:1810.00821

Pfau, D., & Vinyals, O. (2016). Connecting generative adversarial networks and actor-critic methods. Preprint retrieved from arXiv:1610.01945

Plappert, M., Houthooft, R., Dhariwal, P., Sidor, S., Chen, R.Y., Chen, X., Asfour, T., Abbeel, P., & Andrychowicz, M. (2018). Parameter space noise for exploration. In *International conference on learning representations (ICLR)*. Preprint retrieved from arXiv:1706.01905

Pomerleau, D. (1989). ALVINN: An autonomous land vehicle in a neural network. In *Neural information processing systems (NIPS)* (pp. 305–313). http://papers.nips.cc/paper/95-alvinn-an-autonomous-land-vehicle-in-a-neural-network.pdf

Pomerleau, D. (1990). Rapidly adapting artificial neural networks for autonomous navigation. In *Neural information processing systems (NIPS)* (pp. 429–435). http://papers.nips.cc/paper/432-rapidly-adapting-artificial-neural-networks-for-autonomous-navigation.pdf

Puterman, M. L. (1994). *Markov decision processes: Discrete stochastic dynamic programming*. Wiley.

Ratliff, N., Bagnell, J. A., Srinivasa, S. S. (2007). Imitation learning for locomotion and manipulation. In *IEEE-RAS international conference on humanoid robots* (pp. 392–397). https://doi.org/10.1109/ICHR.2007.4813899

Ray, A., Achiam, J., & Amodei, D. (2019). Benchmarking safe exploration in deep reinforcement learning. https://d4mucfpksywv.cloudfront.net/safexp-short.pdf

Reed, S., Aytar, Y., Wang, Z., Paine, T., van den Oord, A., Pfaff, T., Gomez, S., Novikov, A., Budden, D., & Vinyals, O. (2018). *Visual imitation with a minimal adversary*. Tech. rep., Deepmind.

Robbins, H. (1952). Some aspects of the sequential design of experiments. *Bulletin of the American Mathematical Society, 58*(5), 527–535. https://projecteuclid.org/download/pdf_1/euclid.bams/1183517370

Romoff, J., Henderson, P., Piché, A., Francois-Lavet, V., & Pineau, J. (2018). Reward estimation for variance reduction in deep reinforcement learning. In *Conference on robot learning (CoRL)*. Preprint retrieved from arXiv:1805.03359

Rosca, M., Weber, T., Gretton, A., & Mohamed, S. (2020). A case for new neural network smoothness constraints. In *NeurIPS workshop "I Can't Believe It's Not Better"*. http://proceedings.mlr.press/v137/rosca20a/rosca20a.pdf

Ross, S., & Bagnell, J. A. (2010). Efficient reductions for imitation learning. In *International conference on artificial intelligence and statistics (AISTATS)*. http://www.jmlr.org/proceedings/papers/v9/ross10a/ross10a.pdf

Roth, K., Lucchi, A., Nowozin, S., & Hofmann, T. (2017) Stabilizing training of generative adversarial networks through regularization. In *Neural information processing systems (NeurIPS)*. Preprint retrieved from arXiv:1705.09367v2

Russac, Y., Vernade, C., & Cappé, O. (2019). Weighted linear bandits for non-stationary environments. In *Neural information processing systems (NeurIPS)* . Preprint retrieved from arXiv:1909.09146

Rusu, A. A., Colmenarejo, S. G., Gulcehre, C., Desjardins, G., Kirkpatrick, J., Pascanu, R., Mnih, V., Kavukcuoglu, K., & Hadsell, R. (2015). Policy distillation. Preprint retrieved from arXiv:1511.06295

Saxe, A. M., McClelland, J. L., & Ganguli, S. (2013). Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. Preprint retrieved from arXiv:1312.6120

Schaal, S. (1997)(1997) Learning from demonstration. In *Neural information processing systems (NeurIPS)*. http://papers.nips.cc/paper/1224-learning-from-demonstration.pdf

Schlimmer, J. C., & Granger Jr., R. H. (1986). Incremental learning from noisy data. *Machine Learning*. https://link.springer.com/article/10.1007/BF00116895

Schulman, J., Levine, S., Moritz, P., Jordan, M. I., & Abbeel, P. (2015). Trust region policy optimization. In *International conference on machine learning (ICML)*. Preprint retrieved from arXiv:1502.05477

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., & Oleg, K. (2017). Proximal policy optimization algorithms. https://openai-public.s3-us-west-2.amazonaws.com/blog/2017-07/ppo/ppo-arxiv.pdf

Shelhamer, E., Mahmoudieh, P., Argus, M., & Darrell, T. (2016). Loss is its own reward: Self-supervision for reinforcement learning. Preprint retrieved from arXiv:1612.07307

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., van den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., Dieleman, S., Grewe, D., Nham, J., Kalchbrenner, N., Sutskever, I., Lillicrap, T., Leach, M., Kavukcuoglu, K., Graepel, T., & Hassabis, D. (2016). Mastering the game of go with deep neural networks and tree search. *Nature, 529*(7587), 484–489. https://doi.org/10.1038/nature16961

Silver, D., Lever, G., Heess, N., Degris, T., Wierstra, D., & Riedmiller, M. (2014). Deterministic policy gradient algorithms. In *International conference on machine learning (ICML)* (pp. 387–395). http://proceedings.mlr.press/v32/silver14.html

Singh, S., Lewis, R. L., & Barto, A. G. (2009). Where do rewards come from? http://www-anw.cs.umass.edu/pubs/2009/singh_l_b_09.pdf

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research, 15*(56), 1929–1958. https://jmlr.org/papers/v15/srivastava14a.html

Sun, P., Kretzschmar, H., Dotiwalla, X., Chouard, A., Patnaik, V., Tsui, P., Guo, J., Zhou, Y., Chai, Y., Caine, B., Vasudevan, V., Han, W., Ngiam, J., Zhao, H., Timofeev, A., Ettinger, S., Krivokon, M., Gao, A., Joshi, A., Zhao, S., Cheng, S., Zhang, Y., Shlens, J., Chen, Z., & Anguelov, D. (2019). Scalability in perception for autonomous driving: Waymo open dataset. Preprint retrieved from arXiv:1912.04838

Sutton, R. S. (1988). Learning to predict by the methods of temporal differences. *Machine Learning, 3*(1), 9–44. https://link.springer.com/article/10.1007/BF00115009

Sutton, R. S., & Barto, A. G. (1998). *Reinforcement learning: An introduction*. MIT Press.

Sutton, R. S., McAllester, D. A., Singh, S. P., & Mansour, Y. (1999). Policy gradient methods for reinforcement learning with function approximation. In *Neural information processing systems (NIPS)* (pp. 1057–1063). http://papers.nips.cc/paper/1713-policy-gradient-methods-for-reinforcement-learning-with-function-approximation.pdf

Syed, U., Bowling, M., & Schapire, R. E. (2008). Apprenticeship learning using linear programming. In *International conference on machine learning (ICML)* (pp. 1032–1039). https://doi.org/10.1145/1390156.1390286

Syed, U., & Schapire, R. E. (2008). A game-theoretic approach to apprenticeship learning. In *Neural information processing systems (NIPS)* (pp. 1449–1456). http://papers.nips.cc/paper/3293-a-game-theoretic-approach-to-apprenticeship-learning.pdf

Thrun, S., & Schwartz, A. (1993). Issues in using function approximation for reinforcement learning. In *Proceedings of the 1993 connectionist models summer school*. Lawrence Erlbaum. https://www.ri.cmu.edu/pub_files/pub1/thrun_sebastian_1993_1/thrun_sebastian_1993_1.pdf

Todorov, E., Erez, T., & Tassa, Y. (2012). MuJoCo: A physics engine for model-based control. In *IEEE/RSJ international conference on intelligent robots and systems (IROS)* (pp. 5026–5033). https://doi.org/10.1109/IROS.2012.6386109

Uhlenbeck, G. E., & Ornstein, L. S. (1930). On the theory of the Brownian motion. *Physical Reviews, 36*(5), 823–841. https://doi.org/10.1103/PhysRev.36.823

van Hasselt, H. (2010). Double Q-learning. In *Neural information processing systems (NeurIPS)*. https://papers.nips.cc/paper/3964-double-q-learning

van Hasselt, H., Doron, Y., Strub, F., Hessel, M., Sonnerat, N., & Modayil, J. (2018). Deep reinforcement learning and the deadly triad. Preprint retrieved from arXiv:1812.02648

van Hasselt, H., Guez, A., Hessel, M., Mnih, V., & Silver, D. (2016). Learning values across many orders of magnitude. In *Neural information processing systems (NeurIPS)*. Preprint retrieved from arXiv:1602.07714

van Hasselt, H., Guez, A., & Silver, D. (2015). Deep reinforcement learning with double Q-learning. In *AAAI conference on artificial intelligence* (pp. 2094–2100). Preprint retrieved from arXiv:1509.06461

Vinyals, O., Babuschkin, I., Czarnecki, W. M., Mathieu, M., Dudzik, A., Chung, J., Choi, D. H., Powell, R., Ewalds, T., Georgiev, P., Oh, J., Horgan, D., Kroiss, M., Danihelka, I., Huang, A., Sifre, L., Cai, T., Agapiou, J. P., Jaderberg, M., & Silver, D. (2019). Grandmaster level in StarCraft II using multi-agent reinforcement learning. *Nature*. https://doi.org/10.1038/s41586-019-1724-z

Wang, R., Ciliberto, C., Amadori, P., & Demiris, Y. (2019). Random expert distillation: Imitation learning via expert policy support estimation. In *International conference on machine learning (ICML)*. Preprint retrieved from arXiv:1905.06750

Wang, Z., Bapst, V., Heess, N., Mnih, V., Munos, R., Kavukcuoglu, K., & de Freitas, N. (2016). Sample efficient actor-critic with experience replay. Preprint retrieved from arXiv:1611.01224

Wang, Z., Merel, J., Reed, S., Wayne, G., de Freitas, N., & Heess, N. (2017). Robust imitation of diverse behaviors. In *Neural information processing systems (NIPS)*. Preprint retrieved from arXiv:1707.02747

Watkins, C. J. C. H. (1989). *Learning from delayed rewards*. Ph.D. thesis, King's College. http://www.cs.rhul.ac.uk/~chrisw/new_thesis.pdf

Watkins, C. J. C. H., & Dayan, P. (1992). Technical note: Q-learning. *Machine Learning, 8*(3), 279–292. https://doi.org/10.1023/A:1022676722315

Webb, A. R. (1994). Functional approximation by feed-forward networks: a least-squares approach to generalization. *IEEE Transactions on Neural Networks, 5*(3), 363–371. https://doi.org/10.1109/72.286908

Xu, D., & Denil, M. (2019). Positive-unlabeled reward learning. Preprint retrieved from arXiv:1911.00459

Xu, H., & Mannor, S. (2007). The robustness-performance tradeoff in Markov decision processes. In *Neural information processing systems (NeurIPS)*. http://papers.nips.cc/paper/3053-the-robustness-performance-tradeoff-in-markov-decision-processes.pdf

Yang, Y. Y., Rashtchian, C., Zhang, H., Salakhutdinov, R., & Chaudhuri, K. (2020). Adversarial robustness through local lipschitzness. Preprint retrieved from arXiv:2003.02460

Yu, J. Y., & Mannor, S. (2009). Arbitrarily modulated Markov decision processes. In *Conference on decision and control (CDC)* (pp. 2946–2953). https://doi.org/10.1109/CDC.2009.5400662

Yu, J. Y., & Mannor, S. (2009). Online learning in Markov decision processes with arbitrarily changing rewards and transitions. In *International conference on game theory for networks* (pp. 314–322). https://doi.org/10.1109/GAMENETS.2009.5137416

Yu, T., & Sra, S. (2019). Efficient policy learning for non-stationary MDPs under adversarial manipulation. Preprint retrieved from arXiv:1907.09350

Zhang, H., Cisse, M., Dauphin, Y. N., & Lopez-Paz, D. (2017). mixup: Beyond empirical risk minimization. Preprint retrieved from arXiv:1710.09412

Zhao, J., Mathieu, M., & LeCun, Y. (2017). Energy-based generative adversarial network. In *International conference on learning representations (ICLR)*. Preprint retrieved from arXiv:1609.03126

Ziebart, B. D., Maas, A. L., Bagnell, J. A., & Dey, A. K. (2008). Maximum entropy inverse reinforcement learning. In *AAAI conference on artificial intelligence* (pp. 1433–1438). http://www.aaai.org/Papers/AAAI/2008/AAAI08-227.pdf

Zolna, K., Reed, S., Novikov, A., Colmenarej, S. G., Budden, D., Cabi, S., Denil, M., de Freitas, N., & Wang, Z. (2019). Task-relevant adversarial imitation learning. Preprint retrieved from arXiv:1910.01077

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.