Practical Critic Gradient based Actor Critic for On-Policy Reinforcement Learning

Swaminathan Gurumurthy [*] SGURUMUR@ANDREW.CMU.EDU
Carnegie Mellon University

Zachary Manchester ZMANCHES@ANDREW.CMU.EDU
Carnegie Mellon University

J. Zico Kolter ZKOLTER@CS.CMU.EDU
Carnegie Mellon University
Bosch Center for AI

Editors: N. Matni, M. Morari, G. J. Pappas

Abstract

On-policy reinforcement learning algorithms have been shown to be remarkably efficient at learning policies for continuous control robotics tasks. They are highly parallelizable and hence have benefited tremendously from the recent rise in GPU based parallel simulators. The most widely used on-policy reinforcement learning algorithm is proximal policy optimization (PPO) which was introduced in 2017 and was designed for a somewhat different setting with CPU based serial or less parallelizable simulators. However, suprisingly, it has maintained dominance even on tasks based on the highly parallelizable simulators of today. In this paper, we show that a different class of on-policy algorithms based on estimating the policy gradient using the critic-action gradients are better suited when using highly parallelizable simulators. The primary issues for these algorithms arise from the lack of diversity of the on-policy experiences used for the updates and the instabilities arising from the interaction between the biased critic gradients and the rapidly changing policy distribution. We address the former by simply increasing the number of parallel simulation runs (thanks to the GPU based simulators) along with an appropriate schedule on the policy entropy to ensure diversity of samples. We address the latter by adding a policy averaging step and value averaging step (as in off-policy methods). With these modifications, we observe that the critic gradient based on-policy method (CGAC) consistently achieves higher episode returns compared with existing baselines. Furthermore, in environments with high dimensional action space, CGAC also trains much faster (in wall-clock time) than the corresponding baselines.

Keywords: Reinforcement Learning; Actor Critic; Continuous control; Highly parallel Environments

1. Introduction

Over the past few years, we have seen remarkable success of on-policy reinforcement learning algorithms applied to continuous control robotics tasks. Some recent examples include the training of dextrous in hand manipulation (Handa et al., 2022; Chen et al., 2021), training quadruped policies (Chen et al., 2022; Rudin et al., 2021), drone flying (Azar et al., 2021) etc. These successes in large part also seem to have been driven by the recent release of GPU based parallel simulators (Makoviychuk et al., 2021; Xu et al., 2021; Freeman et al., 2021) which have led to vastly faster training and consequently allowed for faster prototyping. Interestingly, despite the change in landscape (i.e, availability of these parallel/differentiable simulators), most of these works (Handa et al., 2022;

[*] Code available at https://github.com/RoboticExplorationLab/CGAC

Chen et al., 2021, 2022; Rudin et al., 2021), still rely on Proximal Policy optimization (PPO), which was released 5 years ago, as their choice of on-policy RL algorithm.

Most existing on-policy algorithms (like PPO (Schulman et al., 2017), TRPO (Schulman et al., 2015a), A2C (Mnih et al., 2016) etc) which are widely used, use the likelihood ratio policy gradient to compute policy updates using a learnt value function. At first glance, given a learnt value function, this is an odd choice given that this only provides a directional derivative for the policy gradient along the sampled transition. As a result, this leads to higher variance estimates of the gradient. The other alternative, which is more common with off-policy algorithms such as DDPG (Silver et al., 2014) and SAC (Haarnoja et al., 2018), is to directly differentiate through a learnt critic Q-function (represented as a function approximator such as a Neural Network). This provides the total derivative for the policy gradient at any sample, albeit, with the catch that the estimate might be more biased. However, with access to highly parallel simulators and a better control of the sampling distribution, it's worth considering if we could train Q-function approximations that aren't as susceptible to the bias and instability issues so that we can leverage the lower variance estimate of the policy gradients that the critic gradients could provide. Furthermore, a critic gradient based on-policy algorithm could help unify the on-policy and off-policy approaches and provide a broader framework that can arbitrarily interpolate between on-policy and off-policy updates.

Thus, in this paper, we propose a practical critic gradient based actor critic (CGAC) method, largely based on algorithms like soft actor-critic (SAC) (Haarnoja et al., 2018) and deterministic policy gradients (DDPG), (Lillicrap et al., 2016) but for on-policy reinforcement learning. One of the primary enablers of this algorithm are the recently released highly parallel simulators (Makoviychuk et al., 2021; Xu et al., 2021; Freeman et al., 2021) which allow us to sample a large batch of environments simultaneously thereby providing a diverse set of samples to perform on-policy updates.

We observe that even when using a large batch of environments, training can still be unstable due to the interaction between the biased critic gradients and the rapidly changing policy distribution. We offer several remedies to fix these issues. First, borrowing from the off-policy RL literature (TD3, SAC etc), we compute the Q-values as a min over two network outputs and use running averages of the Q-networks to compute target values. This helps mitigate the overestimation errors in the Q function. Second, we stabilize the policy sampling distribution by also introducing an averaged policy. The averaged policy is computed by maintaining a running average of the policy parameters and is used to perform rollouts in the environment and to compute the next action for the target value computation. Third, we provide an appropriate schedule for the entropy coefficients to balance the exploration-exploitation trade-off.

We list our key contributions here:

- We propose a practical on policy critic gradient based actor critic (CGAC) method building on previous work on DDPG and SAC.
- We demonstrate its effectiveness in several high dimensional environments such as Ant, Humanoid, SNU Humanoid and the AllegroHand with highly parallelizable simulators. We find that CGAC obtains higher returns compared to existing baselines especially on environments with high dimensional action spaces. Further, CGAC converges 4-5x faster than PPO on a muscle actuated humanoid environment (SNUHumanoid) with 152 dimensional action space.

2. Related Work

Actor critic methods for Reinforcement Learning Actor critic methods have been popular in RL for reducing the variance in the policy gradient updates originating in (Witten, 1977; Barto et al., 1983; Konda and Tsitsiklis, 1999). Many approaches have been proposed since to further reduce the variance of these gradient estimates (Weaver and Tao, 2001; Greensmith et al., 2004; Peters et al., 2008; Bhatnagar et al., 2007; Grondman et al., 2012; Gu et al., 2017a). One way to reduce variance, has been to use the analytical gradients from an action-value critic to estimate the policy gradients (Silver et al., 2014; Lillicrap et al., 2016; Haarnoja et al., 2018). This has been especially popular for off-policy learning as the alternative of using importance weighted likelihood ratio policy gradient estimates (Degris et al., 2012; Graves et al., 2021; Imani et al., 2018) have very high variance and unsuitable for large action spaces (Gu et al., 2017b, 2016). However, most widely used on-policy methods such as PPO (Schulman et al., 2017), TRPO (Schulman et al., 2015a), A2C(Mnih et al., 2016), still use likelihood ratio based policy gradient estimates. This is due to the biased gradient estimates from critic gradients (Silver et al., 2014; Gu et al., 2016). Although there have been previous attempts at using critic gradient based policy gradient estimates for on-policy learning (Silver et al., 2014; Lillicrap et al., 2016), they have mostly served a didactic purpose without actually demonstrating practical merit. They point out the primary impediment as lack of diversity in samples. In this paper, we aim to demonstrate the efficacy of critic gradient based methods in large scale problems leveraging highly parallel simulators.

Reinforcement Learning in highly parallel environments Until recently most work in (deep) reinforcement learning was done with single or few parallel environment executions (Mnih et al., 2015; Schulman et al., 2017). (Nair et al., 2015; Horgan et al., 2018; Mnih et al., 2016; Babaeizadeh et al., 2017; Espeholt et al., 2018; Clemente et al., 2017) moved towards using larger number of environments using distributed systems to simulate multiple parallel environments. However, even the largest of those (Horgan et al., 2018; Clemente et al., 2017) are still only able to scale to 256 environments on CPU clusters. Recent work (OpenAI, 2018; Andrychowicz et al., 2020) further scaled the computation to 1000s of CPU cores and use PPO (Schulman et al., 2017) as the underlying RL algorithm. More recently, we have seen development of a flurry of highly parallelizable GPU based simulators (Makoviychuk et al., 2021; Xu et al., 2021; Freeman et al., 2021) which are capable of simulating tens of 1000s of parallel simulations simultaneously on the GPU. However, most work (Handa et al., 2022; Chen et al., 2022; Rudin et al., 2021; Allshire et al., 2021; Chen et al., 2021; Xu et al., 2021) building on them have limited themselves to using standard reinforcement learning approaches such as PPO (Schulman et al., 2017), SAC (Haarnoja et al., 2018) etc to learn control policies. In this work, we propose a practical critic gradient based on-policy RL algorithm that outperforms these standard approaches when used with highly parallel GPU-based simulators.

3. Preliminaries and Background

In this section, we will introduce the notation used in this paper and then discuss an on policy model free RL algorithm, Proximal Policy Optimization (PPO)(Schulman et al., 2017) and an off policy method called soft actor critic (SAC) (Haarnoja et al., 2018).

3.1. Notation

We primarily focus on on-policy reinforcement learning in continuous action spaces. We will consider Markov decision processes (MDPs) with infinite horizon, but, as with most RL methods, the proposed modifications can easily be extended to finite horizon as well. The MDP state transitions are modelled as a function $f : \mathcal{S} \times \mathcal{A} \rightarrow \mathcal{S}$ and the reward using a bounded reward function $r : \mathcal{S} \times \mathcal{A} \rightarrow [r_{min}, r_{max}]$.

3.2. Critic gradient based off policy actor critic methods

Critic gradient based actor critic algorithms optimize the policy by directly maximizing the Q value estimates computed using the critic :

$$\max_\theta \mathcal{J}_\pi(\theta) = \max_\theta \mathbb{E}_{(s\sim\mathcal{D},a\sim\pi_\theta(.|s))}[Q_\phi(s,a) + \alpha\mathcal{R}(\pi_\theta(a|s))], \tag{1}$$

where, Q_ϕ represents the Q function or the critic and $\mathcal{R}(\pi_\theta(a|s))$ here could be some regularizer (say the entropy of the policy as in SAC). We call these critic gradient based methods as these methods compute the policy gradient only using the gradient of the critic function:

$$\nabla_\theta\mathcal{J}_\pi(\theta) = \mathbb{E}_{s\sim\mathcal{D},\epsilon\sim\mathcal{N}}[\nabla_a Q_\phi(s,a)\nabla_\theta a_\theta(s,\epsilon) + \alpha\nabla_\theta\mathcal{R}(\pi_\theta(a_\theta(s,\epsilon)|s))], \tag{2}$$

where $a_\theta(s,\epsilon)$ could be just the policy outputs (for deterministic policy) or could be a function applying the reparamaterization trick (Kingma and Welling, 2014) to compute the action using the policy output distribution (for stochastic policies) and a randomly sampled noise ϵ from a fixed distribution $\mathcal{N}$.

Critic gradient based off-policy actor critic methods learn the above Q function, Q_ϕ by minimizing the bellman residual over samples drawn from a replay buffer $\mathcal{D}$ of stored samples. SAC and DDPG would be popular examples of such approaches. The bellman residual minimization is given as follows:

$$\min_\phi \mathbb{E}_{(s,a,s',r)\sim\mathcal{D},\epsilon\in\mathcal{N}}[(Q_\phi(s,a) - (R(s,a,s') + \gamma Q_{\bar\phi}(s',a_\theta(s',\epsilon))_\perp)^2] \tag{3}$$

where, $a_\theta(s',\epsilon)$ is obtained using the current policy estimate $\pi_\theta(.|s')$. For SAC, $R(s,a,s') = r + \alpha\mathcal{H}(\pi_\theta(.|s'))$ while DDPG just uses $R(s,a,s') = r$. $\perp$ is the stop gradient operator denoted to mean that the terms within the corresponding parenthesis wouldn't affect the gradient of the loss. Further, $\bar\phi$ are the parameters of the target Q function usually computed as a moving average of ϕ.

While these critic gradient based methods are very popular in off-policy RL, they are seldom used for on-policy RL. This is often attributed to critic gradient providing a biased estimate Silver et al. (2014) of the policy gradient which ends up being particularly problematic for the on-policy case, where the rapidly changing input distribution for the value function can further bias and destabilize training.

3.3. Advantage actor critic methods

Most popular on-policy RL approaches instead use the advantage to compute the likelihood ratio policy gradient. Examples include PPO (Schulman et al., 2017), TRPO (Schulman et al., 2015a), A2C (Mnih et al., 2016) etc. They typically learn a value function, V_ψ, by minimizing the error between $V_\psi(s)$ and a single/multi-step estimate of the discounted return, $R(s,a)$, on the current policy distribution ρ_π. Specifically, $R(s,a)$ is computed using :

$$R(s,a) = A_V^{(\lambda,\gamma,h)}(s,a) + V_\psi(s), \tag{4}$$

where $A^{(\lambda,\gamma,h)}(s,a)$ is the advantage estimate computed over a horizon h, with discount factor γ and a parameter λ. It can be computed using the generalized advantage estimate (Schulman et al., 2015b):

$$A_V^{(\lambda,\gamma,h)}(s_t,a_t) = \sum_{l=0}^{h}(\gamma\lambda)^l\delta_{t+l}^V, \quad \text{where } \delta_{t+l}^V = r_{t+l} + \gamma V_\psi(s_{t+l+1}) - V_\psi(s_{t+l}). \tag{5}$$

Setting $\lambda = 1$ gives us the traditional advantage estimates used in A2C, TRPO etc. The advantage estimates are also used to compute the policy gradients as follows:

$$\nabla_\theta p_\theta(s,a) = \mathbb{E}_{(s,a)\sim\rho_\pi}[A_V^{(\lambda,\gamma,h)}(s,a)\nabla_\theta \log(\pi_\theta(a|s))]. \tag{6}$$

PPO and TRPO use slight variations of Eq 6 to additionally account for the mildly off-policy samples. We observe that the policy gradient estimate in Eq 6 computes a directional derivative along the sampled trajectory. As a result, advantage based estimate of the policy gradient ends up being higher variance than the critic gradient based estimate from Eq 1 (Gu et al., 2017b, 2016). However, they are also less biased.

4. Method

We combine aspects of the above two approaches and propose a critic gradient based actor critic approach for on-policy RL. We expect the resulting algorithm to provide low variance policy gradients and hence converge faster especially on environments with high dimensional action spaces. We will first describe a naive implementation and then discuss the issues faced by this naive implementation. We will then discuss a few fixes that alleviate these problems.

4.1. Critic gradient based actor critic for on-policy RL

We parameterize the Q-function and policy using function approximators such as neural networks. The policy network outputs the mean and standard deviation to parameterize a gaussian distribution. As with other critic gradient based approaches, we maximize the following objective to optimize the policy parameters

$$\max_\theta \mathcal{J}_\pi(\theta) = \max_\theta \mathbb{E}_{(s\sim\mathcal{D},a\sim\pi_\theta(.|s))}[Q_\phi(s,a) - \alpha\log(\pi_\theta(a|s))], \tag{7}$$

where we regularize the entropy of the policy outputs to encourage appropriate exploration. We learn the Q function by supervising it using the single/multi-step discounted return estimate R,

$$\min_\phi \mathcal{J}_\mathcal{Q}(\phi) = \min_\phi \mathbb{E}_{(s\sim\mathcal{D},a\sim\pi_\theta(.|s))}\left[||Q_\phi(s,a) - R(s,a)_\perp||^2\right], \tag{8}$$

$$\text{where,} \quad R(s,a) = A_Q^{(\lambda,\gamma,h)}(s,a) + Q_{\bar\phi}(s,a), \tag{9}$$

where $A_Q^{(\lambda,\gamma,h)}(s,a)$ is a slightly modified generalized advantage estimate (Schulman et al., 2015b) adapted for the Q-function :

$$A_Q^{(\lambda,\gamma,h)}(s_t,a_t) = \sum_{l=0}^{h}(\gamma\lambda)^l\delta_{t+l}^Q, \quad \text{where } \delta_{t+l}^Q = r_{t+l} + \gamma Q_{\bar\phi}(s_{t+l+1},a_{t+l+1}) - Q_{\bar\phi}(s_{t+l},a_{t+l}). \tag{10}$$

Note that we only use $A_Q^{(\lambda,\gamma,h)}(s_t,a_t)$ to compute the returns for value supervision and not to compute the policy gradient estimate.

Like the off-policy counterparts (Haarnoja et al., 2018; Fujimoto et al., 2018), we make use of two Q-functions to mitigate the overestimation issues caused by the above updates. Specifically, we learn two Q-functions, Q_{ϕ_1} and Q_{ϕ_2}, independently to optimize $\mathcal{J}_Q(\phi_i)$. The Q-values in the value targets in Equation 9 and the policy objective in Equation 7 are then computed by taking the minimum of the two Q-functions.

$$Q(s,a) = \min(Q_{\phi_1}(s,a), Q_{\phi_2}(s,a)) \tag{11}$$

Further, we also maintain a moving average of both the Q-functions as is common with off-policy methods and use them while computing the value targets. This also helps mitigate the overestimation issues. Like the on-policy counterparts (Schulman et al., 2017, 2015a; Mnih et al., 2016), our method alternates between collecting experience from the parallel environments with the current policy and using these recent collected experiences to update the Q-functions and policy by optimizing Equations 7 and 8.

This framework is particularly appealing given that off-policy samples can be easily incorporated in the updates by performing a DDPG/SAC type update step and can nicely complement the on-policy updates we perform. Although, we do not explore this in our paper, we believe this ability to conveniently combine off-policy and on-policy updates is one of the primary benefits of this method. Moreover, as discussed previously, we expect this critic gradient based approach for policy optimization to converge faster due to the lower variance of the estimated policy gradients.

However, this naive implementation can be unstable or saturate at lower returns as shown in the ablation experiments in 5.2. This is due to 1) the lack of diversity of the on-policy experiences used for the updates (leading to biased updates) and 2) the instabilities arising from the interaction between the biased critic gradients and the rapidly changing policy distribution. In the next section, we explore these in more detail and propose some candidate solutions to mitigate them.

4.2. Causes for instability and corresponding fixes

State Distribution Diversity One of the motivations for using a replay buffer and performing off-policy RL in the seminal DQN paper (Mnih et al., 2015) was to avoid the problem of correlated samples across timesteps biasing the gradients and value function. However, with the development of highly parallel simulators (Makoviychuk et al., 2021; Xu et al., 2021; Freeman et al., 2021), this problem can be significantly mitigated by sampling from a large batch of environments in parallel. With a decent choice of exploration strategy one can ensure that this sampling distribution is diverse enough to ensure a diverse set of states at each sampling step.

To encourage an appropriate amount of exploration throughout training we simply add a negative penalty on the entropy of the policy outputs as part of the policy objective as in SAC as shown in Eq 7. We use a slightly modified automatic entropy tuning procedure from SAC to tune the entropy coefficient α. We update α to maximize

$$\max_{\alpha} \mathcal{J}_\alpha(\alpha) = \max_{\alpha} \left(\log \pi_\theta(a|s) + \beta n\right) \log \alpha \tag{12}$$

where n is the action space size and β is a hyperparameter. The value of β here chooses the target entropy for the policy that we wish to optimize towards using the entropy regularization. SAC sets this hyperparameter to a fixed value of $\beta = 1$. However, as the policy performance gets better, we typically wish to encourage exploitation. To that end, we propose to gradually decrease the target entropy for policy. Specifically, we use simple linear schedule for β based on the current reward values

$$\beta = \frac{\beta_{final} - \beta_{init}}{\textit{peak reward} - \textit{initial reward}} * (\textit{current reward} - \textit{initial reward}) + \beta_{init} \tag{13}$$

where β_{final} and β_{init} are the final and initial values of β we desire (with $\beta_{final} > \beta_{init}$). *peak reward* and *initial reward* are rough estimates of peak and initial average reward we expect to get in the environment. This allows us to gradually decrease the target entropy for the policy as the policy performance improves during training encouraging further exploitation. Further, while solving Eq 12, we found it helpful to ensure that α monotonically decreases during training. Thus, we clamp the update on α:

$$\alpha = \min(\alpha_{prev}, \alpha_{prev} + \eta \nabla_\alpha \mathcal{J}_\alpha(\alpha_{prev})).$$

to ensure a monotonic decrease in it's value. Intuitively, this helps 1) prevent the policy from arbitrarily exploring regions of the state-action space outside the Q-function's support in the middle of training due to an increase in entropy penalty, 2) lower entropy penalty towards the end of training encourages exploitative behavior in the policy leading to lower variance updates.

Indeed, we observe in the ablation experiments in 5.2 that increasing the number parallel environment executions has a significant impact on the policy performance, both in terms of convergence speeds and the peak returns obtained. Furthermore, the α schedule leads to stable convergence and higher final returns. However, this doesn't completely resolve the issue.

Policy-Value Coupling We observe that, in some cases, a few bad gradient updates can lead to the policy performance suddenly deteriorating and collapsing. This results in a large fraction of parallel episodes abruptly coming to an end and the environments resetting to their initial states. As a consequence a large fraction of the parallel environments at that time step (and a few succeeding steps) collapse to very similar set of states. The sudden change in the input distribution and increase in correlated samples leads to a further deterioration on the value updates consequently going down a spiral of successively worse updates for both the policy and the value function. This issue is less pronounced in existing on-policy approaches such as PPO and TRPO as they are not fully on-policy and use trust region constraints to mitigate rapid changes in policy.

We propose a much simpler solution for this issue. We maintain a running average of the Q-functions to compute value targets, and a running average of the policy to perform rollouts in the environment and compute actions for the target value computation in Equation 9 and 8. Using these running averages of the policy and Q function for rollouts and target value computations ensures that a few bad gradient steps on the original policy and value function don't derail them. This leads to improved stability and prevents the policy distribution from changing too rapidly while keeping our method largely 'on-policy'.

5. Experiments

In this section, we present experimental evidence to demonstrate the effectiveness of CGAC especially in environments with large action spaces. We compare with proximal policy optimization (PPO) (Schulman et al., 2017) and soft actor critic (SAC) (Haarnoja et al., 2018) based on episode returns obtained at convergence or end of training and the time taken to convergence.We also present ablation experiments to analyze and understand where the gains come from. Finally, we also present some initial promising experiments combining off-policy and on-policy updates in CGAC.

Tasks We test the above methods on 4 tasks, namely: Ant, Humanoid, SNUHumanoid and AllegroHand. For the first 3, we use the Dflex simulator(Xu et al., 2021) and use IsaacGym(Makoviychuk et al., 2021) for AllegroHand. The environments have an action space of 8, 21, 152 and 16 dimensions respectively and observation space of 37, 76, 71 and 88 respectively.

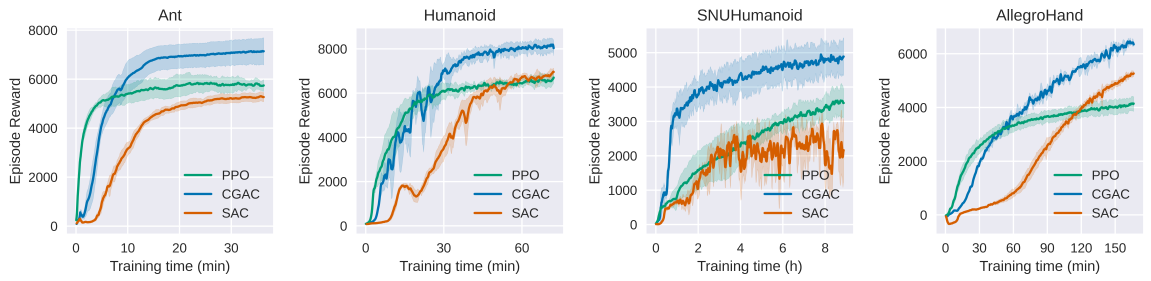


Figure 1: Comparison of on-policy critic gradient based actor critic (CGAC) with PPO and SAC on various high dimensional continuous control tasks.

Hyperparameters We use the hyperparameters provided in the official Isaacgym(Makoviychuk et al., 2021) and Dflex(Xu et al., 2021) repositories for PPO and SAC for all the environments while using the RL-games (Makoviichuk and Makoviychuk, 2022) implementation of the said algorithms. For CGAC, we use largely the same hyperparameters as the SAC implementation and list the ones that differ in Table 1. Although slight improvements in performance are possible in each environment with a different hyperparameter setting, we aim to keep the hyperparameters as similar as possible across environments to demonstrate the robustness to those hyperparameters.

Table 1: CGAC hyperparameters for each task

Hyperparameter	Ant	Humanoid	ANUHumanoid	AllegroHand
Policy Architecture	$[256, 256]$	$[256, 256]$	$[256, 256]$	$[512, 256, 128]$
Critic Architecture	$[256, 256]$	$[256, 256]$	$[256, 256]$	$[512, 256, 128]$
Activation	ELU	ELU	ELU	ELU
Num Actors	4096	4096	4096	16384
Num Critic updates per step	2	2	2	2
(Start) Learning Rate	0.002	0.002	0.002	0.002
Critic avg coeff τ_Q	0.005	0.005	0.005	0.05
Policy avg coeff τ_π	0.001	0.001	0.001	1
final targ ent coeff β_{final}	3.5	7.5	3.5	5
init targ ent coeff β_{init}	0.2	0.2	0.2	0.2
GAE horizon length	32	32	8	8
Batch Size	4096	4096	4096	32768

5.1. Comparisons

Figure 1 shows the average episode return obtained by each method on the 4 continuous control tasks as training progresses. The averages are computed using 5 runs of each algorithm each with a different random seed. We observe that across all the tasks, CGAC outperforms PPO and SAC in terms of the peak episode returns. The difference is especially stark on SNUHumanoid which has a much higher dimensional action space of 152 dimensions. Further we observe that CGAC also converges faster than the baselines in environments with high dimensional action spaces. Again the difference is especially stark on SNUHumanoid, where CGAC converges within 1-2 hours whereas PPO doesn't converged even after 9 hours of training.

Further, we observe some peculiar convergence behaviors which contrasts PPO and CGAC. While CGAC is slower to converge in the initial stages of training, it reaches peak returns much faster than PPO, whereas PPO converges very quickly initially but converges very slowly towards the end. This is simply a consequence of the fact that the policy gradients estimated by CGAC are highly biased and of very poor quality in the initial stages of training when the Q-function is poorly trained but improve quickly in the later stages of training as the Q-function quality improves.

5.2. Ablations and analysis

In this section, we perform various ablations to understand the contribution of the various design decisions made in the algorithm.

Variance analysis Figure 2 shows the signal to noise ratio of the policy gradients computed using the likelihood ratio estimate $A_V^{(\lambda,\gamma,h)}(s,a)\nabla_\theta log\pi_\theta(a|s)$ and the Q function gradients estimate $\nabla_\theta Q_\phi(s,a_\theta(s))$ over the on-policy samples throughout training for the Humanoid Environment. Note that for the likelihood ratio estimate we train a separate value function V_ψ concurrently during training and use it to compute gae advantage estimates $A_V^{(\lambda,\gamma,h)}$. We observe that, as expected, the Q function gradients have a higher signal to noise ratio compared to the advantage based estimates. While it is not possible to conclude from this that the Q function gradients are 'better' as it's difficult to accurately measure the bias of those estimates, we could say that there's some reason to hope that if we mitigate the bias issues in the Q function gradient estimates, we could hope for a better performance. Our modifications to the vanilla algorithm aim to do exactly that by ensuring we use a diverse input distribution to train the networks and that the policy distribution changes smoothly.

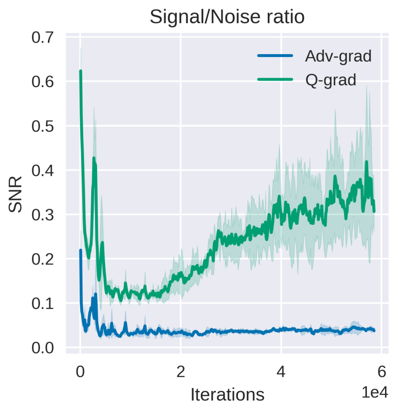


Figure 2: Signal to noise ratio of the policy gradients computed using the likelihood ratio estimate $A_V^{(\lambda,\gamma,h)}(s,a)\nabla_\theta log\pi_\theta(a|s)$ and the Q function gradients $\nabla_\theta Q_\phi(s,a_\theta)$ over the on-policy samples throughout training.

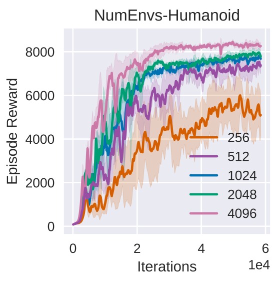
Ablations We inspect the impact of various aspects of the algorithm in this section. Figure 3 shows the corresponding training plots for experiments on the Humanoid and Ant environments.

Figure 3a and 3b shows the training plots with different number of parallel environment runs. We find that the large number of parallel runs enabled by the highly parallel GPU based simulators are immensely useful as the performance improves significantly with the number of parallel runs both in terms of convergence speeds and the average returns at the end of training. This is one of the biggest contributing factors enabling the on-policy nature of the algorithm. To put it in context, before the availability of these GPU based simulators, DDPG/SAC based algorithms using the maximum number of parallel runs was (Horgan et al., 2018) with just 64 parallel runs for the continuous control experiments.

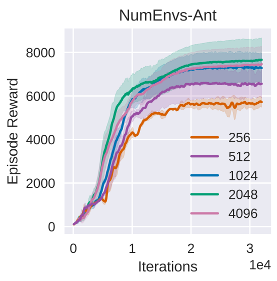
Further, we experiment with changing various aspects of the algorithm one at a time to illustrate their impact in Figures 3c and 3d. We observe that removing the value function averaging (no-val-avg) to compute the target values has a small effect on the training stability and performance as long as we perform policy averaging. This is an interesting contrast with off-policy methods, where the value function averaging plays a crucial role in combating the overestimation issues. However, we observe that removing policy averaging (no-pol-avg) alone has a significant effect on the stability and convergence speed of the method especially in the initial stages of training. Further removing both value and policy averaging (no-polval-avg) leads to a further increase in instability and degradation of performance as it worsens the coupling between a update quality and the policy distribution. As a result, in the initial stages of training (when the Q-function quality is bad, resulting in bad quality updates), the algorithm becomes very unstable. The sudden drops in the episode rewards seen in the plots happen when a few bad updates deteriorate the policy resulting in a bunch of runs suddenly collapsing leading to a temporary collapse in the policy distribution.

Further, we also observe that the exploration parameter alpha often affects the exploration behavior of the agents. As shown in figure , ensuring a strictly monotonic decrease in the alpha values throughout training leads to better final performance as well as more stability during training.

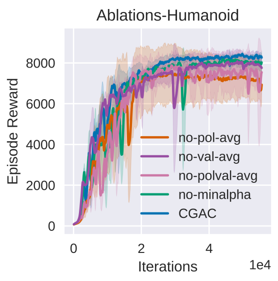
We note that although these phenomenon might not be observed universally across all environments, we do observe that incorporating these changes more often than not do help to improve stability and peak performance and make the algorithm more robust to hyperparameter values. They become especially important as we go to environments with larger action spaces.



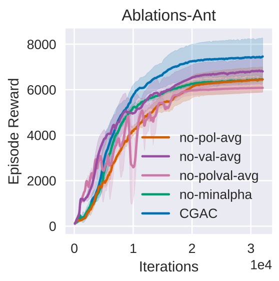
(a)



(b)



(c)



(d)

Figure 3: (a) and (b) : number of parallel environment runs. (c) and (d) : Ablations with switching on/off various other aspects of the algorithm to understand their impact.

6. Discussions and Conclusion

We present a critic gradient based on-policy reinforcement learning algorithm that outperforms popular on-policy and off-policy reinforcement learning algorithms such as PPO and SAC. We show that one of the primary factors enabling the superior performance is the availability of highly parallelizable GPU based simulators which allow us to run a large number of simulation runs in parallel. We also propose several other improvements to the naive version of the algorithm such as maintaining an averaged policy and value averaging for the policy rollouts and target value computations, and, a specific schedule on the entropy coefficient to manage the exploration-exploitation tradeoff. With these changes incorporated, we show that CGAC converges to much higher returns compared to PPO and SAC across all the environments. Further, CGAC also converges much faster than the baselines especially on environments with high dimensional action spaces.

In future work, we believe it's important to explore using off-policy experiences to complement the on-policy experiences for more stable implementations. Given the suitability of this algorithm to incorporate off-policy experiences, we believe that this would be a promising avenue of future research. Further, we also believe that the critic based policy gradient estimates are complementary to the advantage based estimates in algorithms like PPO. Thus, we believe exploring interpolations of these objectives such as in (Gu et al., 2016, 2017b) can also be important avenues of future work.

Acknowledgments

Swaminathan Gurumurthy is supported by a grant from the Bosch Center for Artificial Intelligence. We thank Karnik Ram for help with the experimental setup. Further, we thank various members in the lab for discussions and feedback at various points during the course of the project

References

Arthur Allshire, Mayank Mittal, Varun Lodaya, Viktor Makoviychuk, Denys Makoviichuk, Felix Widmaier, Manuel Wüthrich, Stefan Bauer, Ankur Handa, and Animesh Garg. Transferring dexterous manipulation from gpu simulation to a remote real-world trifinger. *ArXiv*, abs/2108.09779, 2021.

OpenAI: Marcin Andrychowicz, Bowen Baker, Maciek Chociej, Rafal Jozefowicz, Bob McGrew, Jakub Pachocki, Arthur Petron, Matthias Plappert, Glenn Powell, Alex Ray, et al. Learning dexterous in-hand manipulation. *The International Journal of Robotics Research*, 39(1):3–20, 2020.

Ahmad Taher Azar, Anis Koubaa, Nada Ali Mohamed, Habiba A Ibrahim, Zahra Fathy Ibrahim, Muhammad Kazim, Adel Ammar, Bilel Benjdira, Alaa M Khamis, Ibrahim A Hameed, et al. Drone deep reinforcement learning: A review. *Electronics*, 10(9):999, 2021.

Mohammad Babaeizadeh, Iuri Frosio, Stephen Tyree, Jason Clemons, and Jan Kautz. Reinforcement learning through asynchronous advantage actor-critic on a GPU. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=r1VGvBcxl.

Andrew G. Barto, Richard S. Sutton, and Charles W. Anderson. Neuronlike adaptive elements that can solve difficult learning control problems. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-13:834–846, 1983.

Shalabh Bhatnagar, Richard S. Sutton, Mohammad Ghavamzadeh, and Mark Lee. Incremental natural actor-critic algorithms. In *NIPS*, 2007.

Shuxiao Chen, Bike Zhang, Mark Wilfried Mueller, Akshara Rai, and Koushil Sreenath. Learning torque control for quadrupedal locomotion. *ArXiv*, abs/2203.05194, 2022.

Tao Chen, Jie Xu, and Pulkit Agrawal. A system for general in-hand object re-orientation. In *5th Annual Conference on Robot Learning*, 2021. URL https://openreview.net/forum?id=7uSBJDoP7tY.

Alfredo V Clemente, Humberto N Castejón, and Arjun Chandra. Efficient parallel methods for deep reinforcement learning. *arXiv preprint arXiv:1705.04862*, 2017.

Thomas Degris, Patrick M. Pilarski, and Richard S. Sutton. Model-free reinforcement learning with continuous action in practice. *2012 American Control Conference (ACC)*, pages 2177–2182, 2012.

Lasse Espeholt, Hubert Soyer, Remi Munos, Karen Simonyan, Vlad Mnih, Tom Ward, Yotam Doron, Vlad Firoiu, Tim Harley, Iain Dunning, et al. Impala: Scalable distributed deep-rl with importance weighted actor-learner architectures. In *International conference on machine learning*, pages 1407–1416. PMLR, 2018.

C. Daniel Freeman, Erik Frey, Anton Raichuk, Sertan Girgin, Igor Mordatch, and Olivier Bachem. Brax - a differentiable physics engine for large scale rigid body simulation, 2021. URL http://github.com/google/brax.

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

Eric Graves, Ehsan Imani, Raksha Kumaraswamy, and Martha White. Off-policy actor-critic with emphatic weightings. *ArXiv*, abs/2111.08172, 2021.

Evan Greensmith, Peter L. Bartlett, and Jonathan Baxter. Variance reduction techniques for gradient estimates in reinforcement learning. In *Journal of machine learning research*, 2004.

Ivo Grondman, Lucian Buşoniu, Gabriel A. D. Lopes, and Robert Babuka. A survey of actor-critic reinforcement learning: Standard and natural policy gradients. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 42:1291–1307, 2012.

Shixiang Gu, Timothy Lillicrap, Zoubin Ghahramani, Richard E Turner, and Sergey Levine. Q-prop: Sample-efficient policy gradient with an off-policy critic. *arXiv preprint arXiv:1611.02247*, 2016.

Shixiang Shane Gu, Timothy P. Lillicrap, Zoubin Ghahramani, Richard E. Turner, and Sergey Levine. Q-prop: Sample-efficient policy gradient with an off-policy critic. *ArXiv*, abs/1611.02247, 2017a.

Shixiang Shane Gu, Timothy P. Lillicrap, Zoubin Ghahramani, Richard E. Turner, Bernhard Scholkopf, and Sergey Levine. Interpolated policy gradient: Merging on-policy and off-policy gradient estimation for deep reinforcement learning. In *NIPS*, 2017b.

Tuomas Haarnoja, Aurick Zhou, P. Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International Conference on Machine Learning*, 2018.

Ankur Handa, Arthur Allshire, Viktor Makoviychuk, Aleksei Petrenko, Ritvik Singh, Jingzhou Liu, Denys Makoviichuk, Karl Van Wyk, Alexander Zhurkevich, Balakumar Sundaralingam, Yashraj S. Narang, Jean-Francois Lafleche, Dieter Fox, and Gavriel State. Dextreme: Transfer of agile in-hand manipulation from simulation to reality. *ArXiv*, abs/2210.13702, 2022.

Dan Horgan, John Quan, David Budden, Gabriel Barth-Maron, Matteo Hessel, Hado van Hasselt, and David Silver. Distributed prioritized experience replay. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=H1Dy---0Z.

Ehsan Imani, Eric Graves, and Martha White. An off-policy policy gradient theorem using emphatic weightings. In *NeurIPS*, 2018.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *International Conference on Learning Representations*, 2014.

Vijay R. Konda and John N. Tsitsiklis. Actor-critic algorithms. In *NIPS*, 1999.

Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Manfred Otto Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *CoRR*, abs/1509.02971, 2016.

Denys Makoviichuk and Viktor Makoviychuk. rl-games: A high-performance framework for reinforcement learning. https://github.com/Denys88/rl_games, May 2022.

Viktor Makoviychuk, Lukasz Wawrzyniak, Yunrong Guo, Michelle Lu, Kier Storey, Miles Macklin, David Hoeller, Nikita Rudin, Arthur Allshire, Ankur Handa, and Gavriel State. Isaac gym: High performance GPU based physics simulation for robot learning. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*, 2021. URL https://openreview.net/forum?id=fgFBtYgJQX_.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.

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

Arun Nair, Praveen Srinivasan, Sam Blackwell, Cagdas Alcicek, Rory Fearon, Alessandro De Maria, Vedavyas Panneershelvam, Mustafa Suleyman, Charles Beattie, Stig Petersen, et al. Massively parallel methods for deep reinforcement learning. *arXiv preprint arXiv:1507.04296*, 2015.

OpenAI. Openai five. https://blog.openai.com/openai-five/, 2018.

Jan Peters, Sethu Vijayakumar, and Stefan Schaal. Natural actor-critic. *Neurocomputing*, 71:1180–1190, 2008.

N. Rudin, David Hoeller, Philipp Reist, and Marco Hutter. Learning to walk in minutes using massively parallel deep reinforcement learning. *ArXiv*, abs/2109.11978, 2021.

John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In Francis Bach and David Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 1889–1897, Lille, France, 07–09 Jul 2015a. PMLR.

John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*, 2015b.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *ArXiv*, abs/1707.06347, 2017.

David Silver, Guy Lever, Nicolas Manfred Otto Heess, Thomas Degris, Daan Wierstra, and Martin A. Riedmiller. Deterministic policy gradient algorithms. In *International Conference on Machine Learning*, 2014.

Lex Weaver and Nigel Tao. The optimal reward baseline for gradient-based reinforcement learning. *ArXiv*, abs/1301.2315, 2001.

Ian H. Witten. An adaptive optimal controller for discrete-time markov environments. *Inf. Control.*, 34:286–295, 1977.

Jie Xu, Viktor Makoviychuk, Yashraj Narang, Fabio Ramos, Wojciech Matusik, Animesh Garg, and Miles Macklin. Accelerated policy learning with parallel differentiable simulation. In *International Conference on Learning Representations*, 2021.