Variance Reduction in Off-Policy Deep Reinforcement Learning using Spectral Normalization

Anonymous Author(s)
Affiliation
Address
email

Abstract

Off-policy deep reinforcement learning algorithms like Soft Actor Critic (SAC) have achieved state-of-the-art results in several high dimensional continuous control tasks. Despite their success, they are prone to instability due to the *deadly triad* of off-policy training, function approximation, and bootstrapping. Unstable training of off-policy algorithms leads to sample inefficient and sub-optimal asymptotic performance, thus preventing their real-world deployment. To mitigate these issues, previously proposed solutions have focused on advances like target networks to alleviate instability and the introduction of twin critics to address overestimation bias. However, these modifications fail to address the issue of noisy gradient estimation with excessive variance, resulting in instability and slow convergence. Our proposed method, Spectral Normalized Actor Critic (SNAC), regularizes the actor and the critics using spectral normalization to systematically bound the gradient norm. Spectral normalization constrains the magnitudes of the gradients resulting in smoother actor-critics with robust and sample-efficient performance thus making them suitable for deployment in stability-critical and compute-constrained applications. We present empirical results on several challenging reinforcement learning benchmarks and extensive ablation studies to demonstrate the effectiveness of our proposed method.

1 Introduction

Model-free reinforcement learning (RL) algorithms have achieved impressive results in various difficult tasks, such as games (Silver et al., 2017, 2018) and robotic control (Gu et al., 2017; Kalashnikov et al., 2018). In some of the most challenging RL settings, high-dimensional continuous control problems, off-policy actor-critic methods constitute some of the most successful approaches so far (Lillicrap et al., 2016; Fujimoto et al., 2018; Haarnoja et al., 2018). A state-of-the-art (SOTA) member of the off-policy RL family is Soft Actor Critic (SAC) (Haarnoja et al., 2018). SAC augments the standard reinforcement learning objective of maximum reward with a maximum entropy objective that allows it to learn policies that maximize both expected reward and policy entropy, aiding in exploration. However, off-policy reinforcement leaning algorithms like SAC are known to be unstable and sample inefficient, requiring millions of interactions with the environment to learn a well functioning policy. The sample inefficient and unstable sub-optimal polices prevent their wider adoption and deployment in stability-critical and resource-constrained applications.

Off-policy Reinforcement Leaning algorithms are known to be prone to instability due to the *deadly triad* (Sutton and Barto, 2018; Van Hasselt et al., 2018), a combination of function approximation, off-policy learning, and bootstrapping. To address this issue, algorithms use a target network to bootstrap. A pair of Q networks are used to estimate the Q values, overcoming overestimation bias. The target update for learning is the minimum of the two Q networks. These modifications, however,

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

do not fully resolve the issues caused by the deadly triad. The use of function approximators, a finite experience replay buffer and bootstrapping result in inaccurate gradient estimation with excessive variance. The excessive variance then leads to off-policy RL algorithms converging to unstable, sample inefficient, sub-optimal policies.

While the issue of excessive variance in gradient estimates has been well studied in policy-gradient algorithms (policy-only architectures) (Sutton and Barto, 2018; Mnih et al., 2016; Schulman et al., 2017) and Deep Q-learning (Zhao et al., 2019; Jia et al., 2020) (value function only), we focus on off-policy actor-critic methods, which allows us to address this issue in both the policy and the value function approximation. In particular, we analyse the SAC algorithm, as a representative baseline, given its success in continuous control, and investigate spectral normalization as a variance reduction technique. Spectral normalization has been originally proposed and extensively studied in the context of Generative Adversarial Network (GAN) training (Yoshida and Miyato, 2017; Farnia et al., 2018). In essence, this technique regularizes the weight matrix of the layers of the networks to ensure a spectral norm of one, thus bounding the Lipschitz constant of the network. Controlling the Lipschitz constant constrains the magnitudes of the gradients, which enforces smoothness and stabilizes training. In SAC, spectral normalization regularises both the actor and the critic networks, resulting in an approach we call *Spectral Normalized Actor Critic* (SNAC). Our contributions can be summarized as follows:

1. We empirically demonstrate poor gradient estimates and dramatic instability in off-policy actor critic algorithms.
2. We extensively evaluate the effectiveness of spectral normalization in the context of off-policy actor critic RL for continuous control.
3. We present results on complex continuous control benchmarks (Todorov et al., 2012) and ablation studies. The results show SNAC significantly outperforms SAC, especially on high-dimensional tasks, with much higher sample efficiency and a highly stable and robust policy. We demonstrate that reduced variance in gradient estimation leads to reduced variance in performance with no additional fine-tuning required.

2 Background

We consider the standard reinforcement learning framework of a Markov Decision Process defined by a tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, r, \gamma)$ where $\mathcal{S}$ is state space, $\mathcal{A}$ is action space, $\mathcal{P}$ is the transition probability, r is the reward function, and $\gamma \in [0, 1]$ is the discount factor. The agent interacts with the environment at discrete time steps. At each time step t, the agent in its current state s_t executes an action a_t in the environment based on a policy π. The environment returns a reward r_t, and the agent transitions to the next state s_{t+1}. The goal of the agent is to learn the optimal policy π^* that maximizes the expected return $\mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r_t(s_t, a_t)\right]$.

2.1 Soft Actor Critic

The goal in standard reinforcement learning is to learn a policy $\pi(a_t \mid s_t; \phi)$ that maximizes the expected long-term reward objective $\sum_t \mathbb{E}_{(s_t,a_t)\sim\rho^\pi}[r_t(s_t, a_t)]$. SAC is an off-policy maximum-entropy reinforcement learning (Ziebart, 2010) algorithm that augments the standard objective with an entropy term:

$$\pi^* = \arg\max_{\pi} \sum_t \mathbb{E}_{(s_t,a_t)\sim\rho^\pi}[r_t(s_t, a_t) + \alpha\mathcal{H}(\cdot \mid s_t)] \tag{6}$$

where $\alpha > 0$ is a constant. The entropy term ensures that the policy maximizes its entropy along with the reward at each state. Optimal policies are then learned using soft policy iteration. Soft policy iteration learns optimal maximum-entropy policies by alternating between soft policy evaluation and soft policy improvement. In the policy evaluation step, the parameters θ of the soft Q-function, modeled as a neural network, are learned by minimizing the soft Bellman residual:

$$J_Q(\theta) = \mathbb{E}_{(s_t,a_t\sim\mathcal{B})}[\frac{1}{2}(Q_\theta(s_t, a_t) - (r_t(s_t, a_t) + \gamma\, \mathbb{E}_{(s_{t+1})\sim p} V_{\bar{\theta}}(s_{t+1})))^2], \tag{7}$$

where $\mathcal{B}$ is experience buffer, $V_{\bar{\theta}}(s_t) = \mathbb{E}_{a_t\sim\pi}[Q_{\bar{\theta}}(s_t, a_t) - \alpha \log \pi(a_t \mid s_t)]$ and $\bar{\theta}$ are the delayed parameters of the target soft Q-function. In the policy improvement step, the policy is updated by

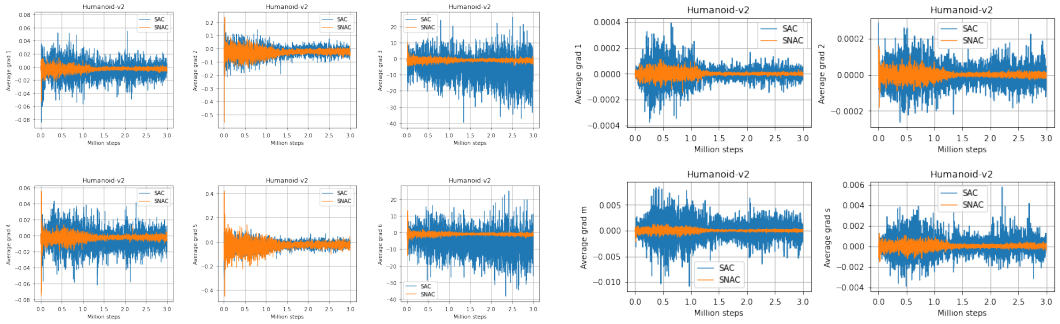

Figure 1: Training dynamics of SAC and SNAC. Solid curves correspond to mean gradient estimates and shaded region correspond to one standard deviation over five random seeds of Humanoid task. **Left** : Top 3 columns correspond to the three layers of first Q network. Bottom 3 columns correspond to the three layers of second Q network. **Right** : Top 2 columns correspond to the input and hidden layer of actor network. Bottom 2 columns correspond to mean and log standard deviation of output Gaussian policy. SAC has large gradient estimates with excessive noise. When spectral normalization is applied (SNAC), the magnitudes of averaged gradient estimation is largely reduced and less variability is observed.

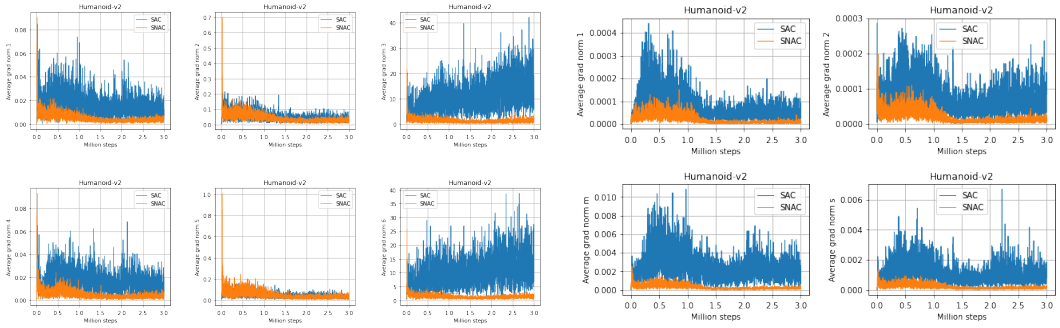

Figure 2: Training dynamics of SAC and SNAC. Solid curves correspond to mean gradient norm and shaded region correspond to one standard deviation over five random seeds of Humanoid task. **Left** : Top 3 columns correspond to the the three layers of first Q network. Bottom 3 columns correspond to the the three layers of second Q network. **Right** : Top 2 columns correspond to the input and hidden layer of actor network. Bottom 2 columns correspond to mean and log standard deviation of output Gaussian policy. SAC has large gradient norms. When spectral normalization is applied (SNAC), the norms are bounded and less variable.

minimizing the KL divergence:

$$J_\pi(\phi) = \mathbb{E}_{s_t \sim \mathcal{B}} \left[D_{KL} \left(\pi_\phi(\cdot \mid s_t) \,\middle\|\, \frac{\exp(\frac{1}{\alpha} Q_\theta(s_t, \cdot))}{Z_\theta(s_t)} \right) \right] \tag{8}$$

where $Z_\theta(s_t)$ is the partition function. The policy is modeled as a Gaussian distribution. The actor network outputs the mean and log standard deviation of the Gaussian policy.

3 Noisy Gradient Estimation and Instability in Off-Policy Actor Critic

To understand how noisy gradient estimates destabilize off-policy actor critic algorithms like SAC, we investigate the training dynamics of SAC. Unlike supervised learning, the off-policy RL training procedure of SAC involves bootstrapping on the target network, which moderately improves stability, but does not fully resolve the statistical estimation issues. The use of function approximators constrains the representation capacity of SAC, further exacerbating gradient estimation. The off-policy training also contributes to distortion in gradient estimation. Off-policy algorithms like SAC are implemented using Adam (Kingma and Ba, 2014) optimizer. Although Adam has shown

robust performance across tasks, it uses first-order gradient information, such as gradient magnitude and gradient variance to update parameters. The use of large gradients estimated from off-policy sample batch (replay buffer) as inputs destabilizes optimization in first-order algorithms and hurts performance.

Figures 1 and 2 show the training dynamics of SAC. We train each algorithm with five different random seeds. We calculate the average gradient and gradient norm of each layer every 1000 environment steps. Figure 1 shows the average gradient of each layer of the actor and the twin critics during training on Humanoid task. Figure 2 show the corresponding gradient norms. The solid curves correspond to the mean and the shaded region corresponds to one standard deviation over the five trials. For SAC, training is highly unstable with large gradients and frequent spikes throughout training of actor and the twin critics. The output layer of the two critics has large and growing gradient norm. The large estimation variance of SAC in Figures 1 and 2 matches the wildly oscillating policy and large variability in rewards in Figure 3.

4 Spectral Normalized Actor Critic (SNAC)

We start the description of our method by first defining Lipschitz continuity, which is a critical concept in the method.

Definition 4.1. *Given two metric spaces $(\mathcal{X}, d_{\mathcal{X}})$ and $(\mathcal{Y}, d_{\mathcal{Y}})$ consisting of a space and a distance metric, a function $f: \mathcal{X} \to \mathcal{Y}$ is Lipschitz continuous if the Lipschitz constant K, defined as:*

$$K_{d_{\mathcal{X}}, d_{\mathcal{Y}}}(f) := \sup_{x_1 \in \mathcal{X}, x_2 \in \mathcal{Y}} \frac{d_{\mathcal{Y}}(f(x_1), f(x_2))}{d_{\mathcal{X}}(x_1, x_2)}$$

is finite.

Equivalently for a Lipschitz f:

$$\forall_{x_1}, \forall_{x_2} d_{\mathcal{Y}}(f(x_1), f(x_2)) \leq K d_{\mathcal{X}}(x_1, x_2).$$

The Lipschitz constant of a composition of functions is bounded by the product of their respective constants:

$$\|f_1 \circ f_2\|_{Lip} \leq \|f_1\|_{Lip} \cdot \|f_2\|_{Lip} \,,$$

where $\|f_1\|_{Lip}$ and $|f_2\|_{Lip}$ are the Lipschitz constants of functions f_1 and f_2 respectively.

For a linear map $g(x) = \mathbf{W}x$ with weight matrix $\mathbf{W}$ and input x, a 1- Lipschitz affine transformation is given by $\|\mathbf{W}x\|_p \leq \|x\|_p$. This is equivalent to constraining the matrix p-norm to at most 1:

$$\|\mathbf{W}\|_p = \sup_{\|x\|_p = 1} \|\mathbf{W}x\|_p.$$

An example of matrix norm is the spectral norm or matrix 2-norm. For the linear map g, the spectral norm is:

$$\sigma(\mathbf{W}) = \max_{x: x \neq 0} \frac{\|\mathbf{W}x\|_2}{\|x\|_2},$$

which is equivalent to largest singular value of the matrix $\mathbf{W}$.

Since 1-Lipschitz functions are closed under composition, we can build a 1-Lipschitz neural network by constraining the Lipschitz constant of all the layers and activation functions of the network to 1. Most activation functions, e.g., ReLU and Tanh, are 1-Lipschitz ($\|a\|_{Lip}$ =1) when scaled. For linear layers of the neural network, we can use spectral normalization to control the Lipschitz constant of a layer by constraining the spectral norm of the layer. When applied to every layer g_n of a network f with N layers, spectral normalization bounds the Lipschitz constant of the network to 1:

$$\|f\|_{Lip} \leq \|g_1\|_{Lip} \cdot \|a_1\|_{Lip} \cdot \|g_2\|_{Lip} \cdots \|g_{N-1}\|_{Lip} \cdot \|a_{N-1}\|_{Lip} \cdot \|g_N\|_{Lip} = \prod_{n=1}^{n=N} \|g\|_n = \prod_{n=1}^{n=N} \sigma(\mathbf{W}_{SN})$$

Our proposed approach, SNAC, built on SAC, adds spectral normalization to the hidden layers of both the actor and the twin critics. Spectral normalization normalizes the spectral norm of weight matrix $\mathbf{W}$:

$$\mathbf{W}_{SN} = \frac{\mathbf{W}}{\sigma(\mathbf{W})},$$

so as to have a Lipschitz constraint of one ($\sigma(\mathbf{W}_{SN}) = 1$). Constraining the spectral norm of hidden layer to one bounds the Lipschitz constant of the actor and the critic networks. Lipschitz constant control the magnitude and variance of the gradients flowing through these networks. The bounded gradients ensure smoother networks with bounded parameter updates. The smoother critics, especially at the beginning of the training, hinder unrealistic Q-value estimates. The bounded value estimates in combination with smoother actor lead to stabilized training and fast convergence of the policy.

Although SNAC is built on top of SAC, since SAC is a representative baseline for continuous control off-policy RL, our approach is algorithm-agnostic and can easily be extended to other existing off-policy actor critic methods, such as DDPG and TD3. We provide pseudo-code for SNAC in the Appendix, based on the original SAC implementation, alongside Pytorch code for reproducibility.

Figure 1 and 2 show the training dynamics of SNAC. Unlike SAC, SNAC achieves well-behaved gradients. The gradient norms of all layers of actor and critics are bounded and decrease as training progresses, leading to a stable result.

5 Experiments

To evaluate spectral normalisation as a variance reduction technique, we assess SNAC's performance on a suite of MuJoCo continuous control tasks (Todorov et al., 2012). We use the original set of tasks without any modifications, applying the same hyperparameters as the original implementation of SAC (Haarnoja et al., 2018), and compare the results using author-provided implementations. The temperature hyperparameter is fixed to $\alpha = 0.2$, and the gradient step is fixed to 1 for SNAC. We train each algorithm with 5 different random seeds and perform 10 evaluation rollouts every 1000 environment steps. All experiments were conducted on an NVIDIA GTX 1050 Ti GPU.

Env	**SNAC**	**SAC**	**Welch t-test**		**Wilcoxon Rank Test**	
			Statistic	p-value	Statistic	p-value
Hopper	**3260.2**$\pm$ 46.5	2980.6$\pm$ 546.1	-9.2	< 0.001	0	< 0.001
Walker2d	5428.6 $\pm$ 169.7	**5524.9**$\pm$ 726.9	1.9	0.063	325.0	< 0.001
HalfCheetah	**15212.8** $\pm$ 697.3	14263.0 $\pm$ 608.1	34.7	< 0.001	1	< 0.001
Ant	**6716.6** $\pm$ 225.41	4911.7 $\pm$ 1638.6	-35.5	< 0.001	0	< 0.001
Humanoid	**7765.1** $\pm$ 256.42	5701.4 $\pm$ 1286.0	-30.9	< 0.001	35.0	< 0.001

Table 1: Comparison of algorithms across tasks from the MujoCo benchmark after 3M timesteps. We compare SNAC with baseline SAC. By using spectral normalization, SNAC learns an extremely stable policy and achieves statistically significant decrease in variance. The stable policy helps SNAC outperform SAC.

5.1 Comparative Evaluation

Figure 3 shows the learning curves of our algorithm and the baseline on complex tasks. SNAC achieves significantly stable learning curves on all tasks. In contrast, SAC is highly unstable with wildly oscillating curves on all tasks except HalfCheetah-v2. We performed two-tailed Welch's t-test (Welch, 1947) to determine whether final performance of SAC and SNAC is statistically significantly different. We also performed Wilcoxon Sign-Ranked test (Wilcoxon, 1945) to analyze the effect of spectral normalization on variance.

Observing Table 1, SNAC outperforms SAC on high dimensional tasks like Ant-v2 (action space dimensionality: 8, state space dimensionality: 111) and Humanoid-v2 (action space dimensionality: 17, state space dimensionality: 376), achieving state-of-the-art final performance and sample efficiency on Humanoid-v2. On low-dimensional tasks like HalfCheetah-v2 (action space dimensionality: 6, state space dimensionality: 17) and Hopper-v2 (action space dimensionality: 3, state space dimensionality: 11), SNAC also performs better than SAC. Walker-2d (action space dimensionality: 6, state space dimensionality: 17) is a highly unstable environment. As a result, SNAC traded off marginal performance gains for stability, but the difference in performance is statistically insignificant as shown by Welch's t-test for Walker-2d. In the highly unstable Hopper task, SNAC also prioritizes

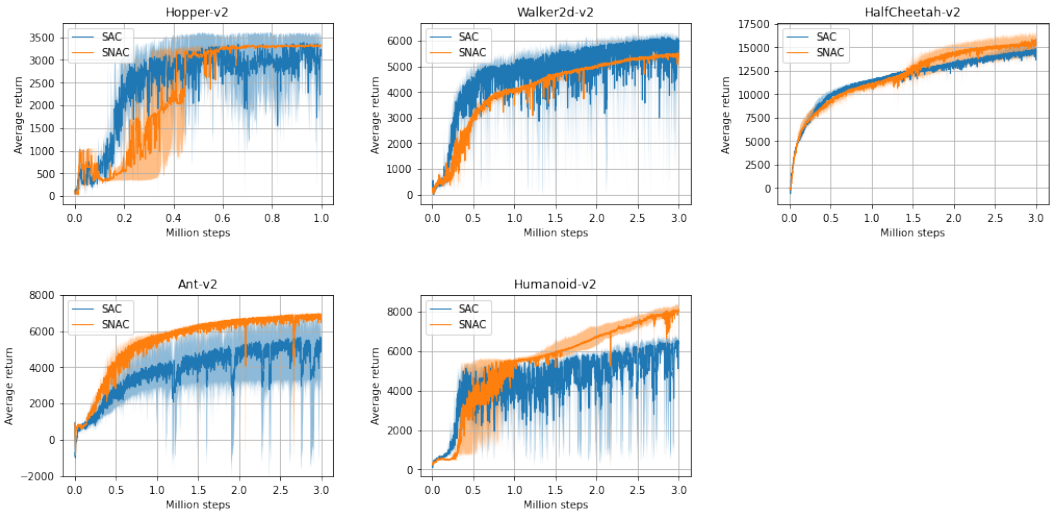

Figure 3: Learning curves of SAC and SNAC. Solid curves correspond to mean returns and shaded region correspond to minimum and maximum returns over five random seeds. SNAC outperforms SAC, especially on high dimensional tasks. The learning curve of SNAC are highly stable with statistically significant decrease in variance compared to SAC.

stability, but since the environment is simple enough, SNAC ultimately converges and achieves a better result than the baseline in less than one million steps. Overall, SNAC achieves a statistically significant decrease in variance across most tasks, as shown by Wilcoxon Ranked Test, except for the highly stable HalfCheetah environment.

5.2 Ablation Studies

SNAC with spectral normalization on a single hidden layer outperforms SAC on multiple tasks and achieves state-of-the-art results on high-dimensional problems. We next investigate if better performance can be achieved with larger networks having spectral normalization applied to different layers.

5.2.1 Effect of Spectral Normalization on Different Layers

In the original implementation of SNAC, we applied spectral normalization to the single hidden layer of the actor and the twin critics. We next investigate the effect of spectral normalization on different layers of the neural networks by implementing SNAC with two hidden layers (SNAC_2). We use Python list indexing to specify layer number. The output layer is specified by index -1 and index -2 stands for the penultimate layer. We apply spectral normalization to the output layer (SNAC_2[-1]), to the hidden layer closer to the output layer (SNAC_2[-2]), to the hidden layer closer to the input layer (SNAC_2[-3]) and to both the hidden layers (SNAC_2[-3, -2]).

Figure 4 shows the learning curves for SAC, SNAC and the different implementations of SNAC_2 on Humanoid-v2 and Walker2d-v2 tasks. All versions of SNAC_2 learn a reasonably performant policy faster than SAC and SNAC. However, except for SNAC_2[-3,-2], all implementations of SNAC_2 have poorer asymptotic performance compared to SNAC. SNAC_2[-3,-2] with spectral normalization on both hidden layers greatly outperform other implementations. From Table 2 we can conclude that, for small-sized networks, constraining all hidden layers of actor and critics improves performance.

5.2.2 Spectral Normalization and Deeper Networks

Computer vision and natural language processing tasks have been shown to greatly benefit from larger and deeper networks, with deeper networks learning more generalizable representations. In contrast, RL often relies on smaller networks. Deeper networks in RL are known to be highly unstable with performance deteriorating drastically with depth. The inability of RL neural networks to leverage

size and depth has been well studied (Sinha et al., 2020; Andrychowicz et al., 2020; Bjorck et al., 2021). We investigate the effect of spectral normalization on deeper networks in the RL context.

Algorithm	Humanoid	Walker2d
SAC	5701.4 ± 1286.0	5524.9 ± 726.9
SNAC	7765.1 ± 256.42	5428.6 ± 169.7
SNAC-2[-1]	7571.8 ± 312.5	5325.7 ± 233.5
SNAC_2[-2]	7761.3 ± 161.3	5384.9 ± 479.1
SNAC_2[-3]	7552.0 ± 179.5	5214.3 ± 243.4
SNAC_2[-3,-2]	**8390.4** ± 187.1	**5560.1** ± 204.6

Table 2: Comparison of algorithms on Humanoid and Walker2d tasks from the MujoCo benchmark. We compare implementations of SNAC with spectral normalization on different layers. SNAC_2 has two hidden layers. All implementations of SNAC_2 achieve similar asymptotic performance except SNAC_2[-3,-2]. SNAC_2[-3,-2] outperforms all algorithms.

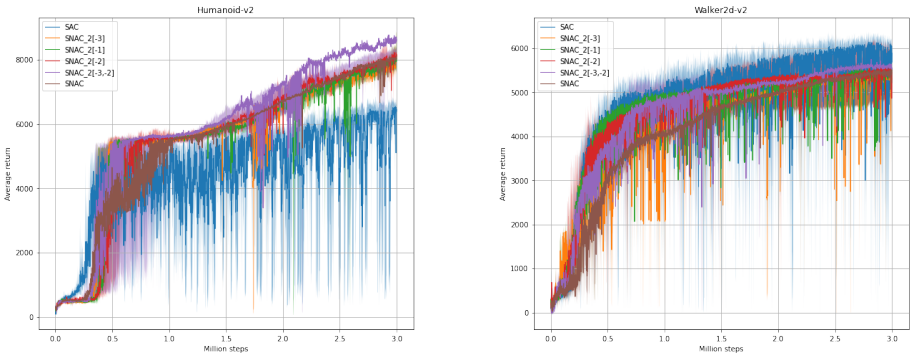

Figure 4: Learning curves of SAC and implementations of SNAC with spectral normalization on different layers. Solid curves correspond to mean returns and the shaded region corresponds to minimum and maximum returns over five random seeds. SNAC_2[-3,-2] with constraints on both hidden layers outperforms other implementations.

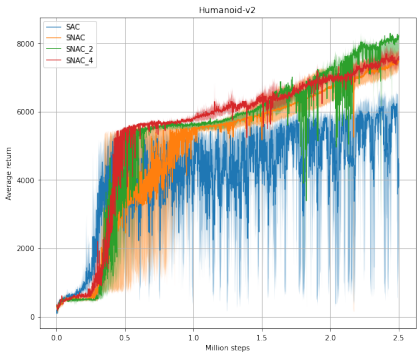

Figure 5: Learning curves of SAC and implementations of SNAC with spectral normalization on increasing number of hidden layers. Solid curves correspond to mean returns and the shaded region corresponds to minimum and maximum returns over five random seeds. SNAC_2 with constraints on both hidden layers outperforms other implementations. Deeper network like SNAC_4 quickly learns a more performant and stable policy. However, it is unable to leverage its early performance with the learning curve plateauing and performing worse than SNAC_2.

Figure 5 shows the learning curves for SNAC with one, two (SNAC_2) and four (SNAC_4) hidden layers. We have applied normalization to all the hidden layers of each implementation. SNAC_2 has

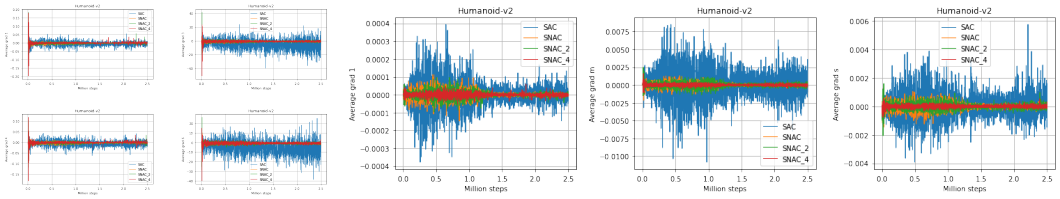

Figure 6: Training dynamics of SAC and different implementations of SNAC. Solid curves correspond to mean gradient and shaded region correspond to one standard deviation over five random seeds of Humanoid task. **Left** : Top 2 columns correspond to the the first and last layers of first Q network. Bottom 2 columns correspond to the the first and last layers of second Q network. **Right** : The three columns correspond to the input layer, mean and log standard deviation layer of output Gaussian policy (actor). All implementations of SNAC have statistically indistinguishable average gradient and variance throughout the training of the critic networks. However, the average gradients of the policy network for SNAC_4 are smaller than the those for SNAC and SNAC_2.

the best asymptotic performance among different implementations of SNAC. SNAC_4 quickly learns meaningful behaviour and is more sample efficient and stable than the the other two implementations during the early part of the training (500k steps). However, the implementation is unable to leverage its early performance with the learning curve plateauing in the later half of the training and approaching performance of SNAC. Bjorck et al. (2021) propose that smoothing with SN improves performance when using multiple hidden layer MLPs. We, on the other hand, find that naively using SN with deeper MLPs does not work. The above contrast in results could be due to Bjorck et al. (2021) conducting ablation on simpler tasks, like pendulum, hopper, walker, cheetah, thus overestimating the role of SN in performance increase with deeper networks.

Figure 6 shows the gradients for the first and last layers of actors and critics of SNAC implementations with different depths. Unlike Gogianu et al. (2021), all implementations of SNAC have statistically indistinguishable average gradient and variance throughout the training of the critic networks. However, the average gradients of the policy network for SNAC_4 are smaller than those of SNAC and SNAC_2. The small gradients are the result of strong regularization, especially at the beginning of the training. Constraining all the hidden layers of the deeper actor network of SNAC_4 makes the network extremely smooth and significantly reduces the magnitudes of the gradients. As a result, the Gaussian policy becomes too concentrated too early in the training causing under-exploration. The under-exploration leads the actor to never sample actions that may improve performance and instead converges to a sub-optimal policy. Relaxing the constraint and applying spectral normalization to few layers instead of all layers of deeper actor networks, would perhaps lead to a better trade-off between stability and performance.

5.3 Spectral Normalization and Over-estimation Bias

We next investigate whether spectral normalization is enough to mitigate overestimation bias in off-policy actor critic networks. Off-policy RL algorithms are known to suffer from consistent over estimation bias especially at the early stage of learning. Typical off-policy algorithms apply the max operator over the TD estimates of action value functions. But these estimators are prone to estimation errors (Thrun and Schwartz, 1993) which can arise due to noisy environments, use of function approximators or presence of any kind of stochasticity. Taking the maximum of these noisy action value estimates results in positively biased Q-value estimates leading to instability and sub-optimal policies.

Algorithms like SAC employ a pair of critics to mitigate over-estimation bias. The target update for learning is the minimum of the two Q functions. Taking the minimum prevents introduction of additional overestimation over the standard Q-learning target. To study the effect of spectral normalization on overestimation bias, we implemented SAC with just one Q network and spectral normalization. We call this architecture SNAC_One. To evaluate SNAC_One, we measure its performance on Walker2d and Humanoid environments. Table 3 shows comparison of SNAC_One with SNAC and SAC. SNAC_One performs worse than SNAC and SAC on all tasks and is highly unstable compared to them. The ablative study provides further evidence that spectral normalization primarily targets noisy gradients to stabilize training.

Algorithm	Humanoid	Walker2d
SAC	5701.4 ± 1286.0	**5524.9** ± 726.9
SNAC	**7765.1** ± 256.42	5428.6 ± 169.7
SNAC_One	4524.8 ± 1312.2	4391.5 ± 618.4

Table 3: Comparison of algorithms on Humanoid and Walker2d tasks from the MujoCo benchmark. We compare SNAC_One (SNAC with one Q network) with SNAC and SAC. SNAC_One performs worse than SAC and SNAC on both tasks.

6 Related Work

6.1 Stabilized Q-learning

Over the past few years, several algorithms have been proposed to address instability in Q learning. TD3 and SAC mitigate the overestimation bias by using twin critics. Softmax Deep Double Deterministic Policy Gradients (SD2) (Pan et al., 2020), Optimistic Actor Critic (OAC) (Ciosek et al., 2019), Truncated Quantile Critics (TQC) (Kuznetsov et al., 2020), Bagged Critic for Continuous control (BC3) (Bawa and Ramos, 2021) reduce overestimation bias. SUNRISE (Lee et al., 2021) uses an ensemble-based weighted Bellman backups to estimate Q-values, and upper confidence bounds for efficient exploration. Cautious Actor Critic (CAC) (Zhu et al., 2021) combines a conservative actor with a conservative critic to address the oscillating performance of off-policy learning. Crossnorm (Bhatt et al., 2019) uses a mixture of on- and off-policy transitions to mitigate divergence and to improve returns in deep off-policy learning without requiring target networks. Our approach, in contrast, targets noisy gradients to stabilize training.

6.1.1 Spectral Normalization

Spectral Normalization has been widely studied in the context of Generative Adversarial Network (GAN) training and stabilization (Yoshida and Miyato, 2017; Farnia et al., 2018; Gouk et al., 2021; Lin et al., 2021). SVD parameterization has also been used to stabilize training in RNNs (Zhang et al., 2018). Asadi et al. (2018) studied the impact of Lipschitz continuity in the context of model based reinforcement learning. Yu et al. (2020) used spectral normalization to improve uncertainty estimates in offline model based reinforcement learning setting. Gogianu et al. (2021) showed performance improvement in Categorical-DQN due to spectral normalization and its effect on the optimisation dynamics. Bjorck et al. (2021) showed increased stability in SAC with Transformer-inspired (Vaswani et al., 2017) architectures with residual connections, layer normalization and spectral normalization. Our paper, in contrast investigates the effectiveness and limitations of spectral normalization in the context of off-policy actor critic algorithms with conventional multi-layer perceptron architectures.

7 Conclusion

In this paper, we investigated the issue of noisy gradient estimation in off-policy actor critic algorithms and proposed spectral normalization based variance reduction. Our proposed approach, SNAC, outperforms state-of-the-art on high dimensional continuous control tasks. Even with deeper networks, SNAC continues to be extremely stable throughout training and quickly learns a performant policy. However, constraining all hidden layers of deep actor networks negatively affects the asymptotic performance with SNAC failing to leverage the expressive power of deeper networks. We hypothesize that relaxing the constraint on actor networks would lead to a better trade-off between stability and performance. In addition, fine-tuning the Lipschitz constant or implementing a scheduler for Lipschitz constant would enable use of deeper networks.

In the future, we would like to investigate advantages of the spectral normalization technique from a theoretical perspective. We would also like to adapt other variance reduction techniques from GANs and the policy gradient literature which can help mitigate noisy gradients in deeper off-policy RL algorithms.

References

Marcin Andrychowicz, Anton Raichuk, Piotr Sta'nczyk, Manu Orsini, Sertan Girgin, Raphaël Marinier, L'eonard Hussenot, Matthieu Geist, Olivier Pietquin, Marcin Michalski, Sylvain Gelly, and Olivier Bachem. What matters in on-policy reinforcement learning? a large-scale empirical study. *ArXiv*, abs/2006.05990, 2020.

Kavosh Asadi, Dipendra Misra, and Michael Littman. Lipschitz continuity in model-based reinforcement learning. In *International Conference on Machine Learning*, pages 264–273. PMLR, 2018.

Payal Bawa and Fabio Ramos. Bagged critic for continuous control. *ICML 2021 Workshop on Reinforcement Learning Theory*, 2021.

Aditya Bhatt, Max Argus, Artemij Amiranashvili, and Thomas Brox. Crossnorm: Normalization for off-policy td reinforcement learning. *arXiv preprint arXiv:1902.05605*, 2019.

Nils Bjorck, Carla P Gomes, and Kilian Q Weinberger. Towards deeper deep reinforcement learning with spectral normalization. *Advances in Neural Information Processing Systems*, 34, 2021.

Kamil Ciosek, Quan Vuong, Robert Loftin, and Katja Hofmann. Better exploration with optimistic actor critic. *Advances in Neural Information Processing Systems*, 32, 2019.

Farzan Farnia, Jesse M Zhang, and David Tse. Generalizable adversarial training via spectral normalization. *arXiv preprint arXiv:1811.07457*, 2018.

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.

Florin Gogianu, Tudor Berariu, Mihaela C Rosca, Claudia Clopath, Lucian Busoniu, and Razvan Pascanu. Spectral normalisation for deep reinforcement learning: an optimisation perspective. In *International Conference on Machine Learning*, pages 3734–3744. PMLR, 2021.

Henry Gouk, Eibe Frank, Bernhard Pfahringer, and Michael J Cree. Regularisation of neural networks by enforcing lipschitz continuity. *Machine Learning*, 110(2):393–416, 2021.

Shixiang Gu, Ethan Holly, Timothy Lillicrap, and Sergey Levine. Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates. In *2017 IEEE international conference on robotics and automation (ICRA)*, pages 3389–3396. IEEE, 2017.

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

Haonan Jia, Xiao Zhang, Jun Xu, Wei Zeng, Hao Jiang, Xiaohui Yan, and Ji-Rong Wen. Variance reduction for deep q-learning using stochastic recursive gradient. *arXiv preprint arXiv:2007.12817*, 2020.

Dmitry Kalashnikov, Alex Irpan, Peter Pastor, Julian Ibarz, Alexander Herzog, Eric Jang, Deirdre Quillen, Ethan Holly, Mrinal Kalakrishnan, Vincent Vanhoucke, et al. Scalable deep reinforcement learning for vision-based robotic manipulation. In *Conference on Robot Learning*, pages 651–673. PMLR, 2018.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Arsenii Kuznetsov, Pavel Shvechikov, Alexander Grishin, and Dmitry Vetrov. Controlling overestimation bias with truncated mixture of continuous distributional quantile critics. In *International Conference on Machine Learning*, pages 5556–5566. PMLR, 2020.

Kimin Lee, Michael Laskin, Aravind Srinivas, and Pieter Abbeel. Sunrise: A simple unified framework for ensemble learning in deep reinforcement learning. In *International Conference on Machine Learning*, pages 6131–6141. PMLR, 2021.

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.

Zinan Lin, Vyas Sekar, and Giulia Fanti. Why spectral normalization stabilizes gans: Analysis and improvements. *Advances in neural information processing systems*, 34, 2021.

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.

Ling Pan, Qingpeng Cai, and Longbo Huang. Softmax deep double deterministic policy gradients. *Advances in Neural Information Processing Systems*, 33:11767–11777, 2020.

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

David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, Yutian Chen, Timothy P. Lillicrap, Fan Hui, Laurent Sifre, George van den Driessche, Thore Graepel, and Demis Hassabis. Mastering the game of go without human knowledge. *Nat.*, 550(7676):354–359, 2017. URL http://dblp.uni-trier.de/db/journals/nature/nature550.html#SilverSSAHGHBLB17.

David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, Timothy Lillicrap, Karen Simonyan, and Demis Hassabis. A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science*, 362(6419):1140–1144, 2018. doi: 10.1126/science.aar6404. URL https://www.science.org/doi/abs/10.1126/science.aar6404.

Samarth Sinha, Homanga Bharadhwaj, Aravind Srinivas, and Animesh Garg. D2rl: Deep dense architectures in reinforcement learning. *arXiv preprint arXiv:2010.09163*, 2020.

Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

Sebastian Thrun and Anton Schwartz. Issues in using function approximation for reinforcement learning. In *Proceedings of the 1993 Connectionist Models Summer School*, pages 255–263, 1993.

E. Todorov, T. Erez, and Y. Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems, (IROS 2012)*, page 5026–5033, 2012.

Hado Van Hasselt, Yotam Doron, Florian Strub, Matteo Hessel, Nicolas Sonnerat, and Joseph Modayil. Deep reinforcement learning and the deadly triad. *arXiv preprint arXiv:1812.02648*, 2018.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

B. L. Welch. The generalization of 'student's' problem when several different population variances are involved. *Biometrika*, 34(1/2):28–35, 1947. ISSN 00063444.

Frank Wilcoxon. Individual comparisons by ranking methods. *Biometrics Bulletin*, 1(6):80–83, 1945. ISSN 00994987.

Yuichi Yoshida and Takeru Miyato. Spectral norm regularization for improving the generalizability of deep learning. *arXiv preprint arXiv:1705.10941*, 2017.

Tianhe Yu, Garrett Thomas, Lantao Yu, Stefano Ermon, James Y Zou, Sergey Levine, Chelsea Finn, and Tengyu Ma. Mopo: Model-based offline policy optimization. *Advances in Neural Information Processing Systems*, 33:14129–14142, 2020.

Jiong Zhang, Qi Lei, and Inderjit Dhillon. Stabilizing gradients for deep neural networks via efficient svd parameterization. In *International Conference on Machine Learning*, pages 5806–5814. PMLR, 2018.

Wei-Ye Zhao, Xiya Guan, Yang Liu, Xiaoming Zhao, and Jian Peng. Stochastic variance reduction for deep q-learning. In *AAMAS*, 2019.

Lingwei Zhu, Toshinori Kitamura, and Matsubara Takamitsu. Cautious actor-critic. In *Asian Conference on Machine Learning*, pages 220–235. PMLR, 2021.

B. D. Ziebart. *Modeling purposeful adaptive behavior with the principle of maximum causal entropy*. PhD thesis, Carnegie Mellon University, 2010.

```
1. #Intialize Q networks (Q_MLP),target networks and actor network (
                                    A_MLP).
2 # Sample state, action from the replay buffer
3 state, action = replay_buffer.sample()
4 # Feed state, action into the first linear layer of a Q-network
5 q_input = concatenate(state, action)
6 h = Q_MLP_input_layer(q_input)
7 # Spectral normalize the weights of the hidden layer
8 h = spectral_normalize(h)
9 # Feed the representation into the output layer
10 o = Q_MLP_output_layer(h)
11 # Feed state into the first linear layer of  Actor-network
12 h = A_MLP_input_layer(state)
13 # Spectral normalize the weights of the hidden layer
14  h = spectral_normalize(h)
15 # Feed the representation into the output layer of Actor network
16 o = A_MLP_output_layer(h)
17 # Update the target networks
```

Figure 7: Pseudo-code for SNAC

A Pseudo-code

B Hyperparameters

The table 4 lists the parameters used to implement SNAC and SAC.

Hyperparameter	SNAC	SAC
Optimizer	Adam	Adam
Learning rate	10^{-4}	10^{-4}
Discount γ	0.99	0.99
Replay buffer size	10^6	10^6
Number of hidden layers (all networks)	1	1
Number of hidden units per layer	256	256
Minibatch size	256	256
Nonlinearity	ReLU	ReLU
Target smoothing coefficient τ	0.005	0.005
Target update interval	1	1
Gradient steps per iteration	1	1
Environment steps per iteration	1	1
Temperature coefficient α	0.2	-dim($\mathcal{A}$)

Table 4: Hyperparameters for SNAC and SAC.

C Pytorch Code

```
class QNetwork(nn.Module):
    def __init__(self, num_inputs, num_actions, hidden_dim):
        super(QNetwork, self).__init__()

        # Q1 architecture
        self.linear1 = nn.Linear(num_inputs + num_actions, hidden_dim)
        self.linear12 = U.spectral_norm(nn.Linear(hidden_dim,
                                          hidden_dim))
        self.linear2 = U.spectral_norm(nn.Linear(hidden_dim,
                                          hidden_dim))
        self.linear3 = nn.Linear(hidden_dim, 1)
```

```python

        # Q2 architecture
        self.linear4 = nn.Linear(num_inputs + num_actions, hidden_dim)
        self.linear45 = U.spectral_norm(nn.Linear(hidden_dim,
                                        hidden_dim))
        self.linear5 = U.spectral_norm(nn.Linear(hidden_dim,
                                        hidden_dim))
        self.linear6 = nn.Linear(hidden_dim, 1)

        self.apply(weights_init_)

class GaussianPolicy(nn.Module):
    def __init__(self, num_inputs, num_actions, hidden_dim,
                                    action_space=None):
        super(GaussianPolicy, self).__init__()

        self.linear1 = nn.Linear(num_inputs, hidden_dim)
        self.linear12 = U.spectral_norm(nn.Linear(hidden_dim,
                                        hidden_dim))
        self.linear2 = U.spectral_norm(nn.Linear(hidden_dim,
                                        hidden_dim))

        self.mean_linear = nn.Linear(hidden_dim, num_actions)
        self.log_std_linear = nn.Linear(hidden_dim, num_actions)

        self.apply(weights_init_)
```