Improving Exploration in Soft-Actor-Critic with Normalizing Flows Policies

Patrick Nadeem Ward [*1,2] **Ariella Smofsky** [*1,2] **Avishek Joey Bose** [1,2]

Abstract

Deep Reinforcement Learning (DRL) algorithms for continuous action spaces are known to be brittle toward hyperparameters as well as sample inefficient. Soft Actor Critic (SAC) proposes an off-policy deep actor critic algorithm within the maximum entropy RL framework which offers greater stability and empirical gains. The choice of policy distribution, a factored Gaussian, is motivated by its easy re-parametrization rather than its modeling power. We introduce Normalizing Flow policies within the SAC framework that learn more expressive classes of policies than simple factored Gaussians. We show empirically on continuous grid world tasks that our approach increases stability and is better suited to difficult exploration in sparse reward settings.

1. Introduction

Policy Gradient methods have been the defacto gold standard for scaling up Reinforcement Learning with non-linear function approximation. In continuous control tasks stochastic policies have the added benefit of permitting on-policy exploration and learning from sampled experience in an off-policy manner (Heess et al., 2015). Soft Actor Critic presents a stable model free DRL algorithm which augments the standard maximum reward Reinforcement Learning objective with an entropy maximization term (Haarnoja et al., 2018). SAC allows for the learning of policies that maximize both expected reward and policy entropy which aids in stability, exploration and robustness (Ziebart et al., 2008; Ziebart, 2010). Nevertheless, most stochastic policies are Gaussian policies that make use of the reparameterization trick in generative modeling to directly optimize distributional parameters via the pathwise gradient estimator (Schulman et al., 2015). While computationally attractive, Gaussian policies have limited modeling expressivity i.e. only policies that are diagonal Gaussians. In many learning tasks the choice of target policy can be a limiting assumption. For instance, in real world robotics tasks where actions may correspond to bounded joint angles due to physical constraints, a Beta policy has been shown to converge faster and to better policies (Chou et al., 2017).

In this paper, we analyze the utility of modeling stochastic policies using different probability distributions for continuous control tasks. In most cases the form of the optimal policy is not known *a priori* and thus picking an appropriate distribution for modeling stochastic policies can be difficult without domain specific knowledge. We propose to bypass this modeling bottleneck by introducing Normalizing Flow policies that define a sequence of invertible transformations that map a sample from a simple density to a more complex density via the change of variable formula for probability distributions (Rezende & Mohamed, 2015). In short, these policies can learn arbitrarily complex distributions but still retain the attractive properties of sampling from a simple distribution such as a reparameterized Gaussian and having a defined density making it amenable to the maximum entropy RL framework. While past work applies Normalizing flows for on-policy learning (Tang & Agrawal, 2018), we instead apply Normalizing Flows to Soft Actor Critic. We argue the use of Normalizing Flows is a natural choice as the squashing function used in SAC to enforce action bounds can be viewed as a simple one layer flow. Extending SAC with multiple and more complex flow models is the core contribution of this work. We empirically evaluate our work on continuous grid world environments with both dense and sparse rewards and find that Normalizing Flow policies converge faster and to higher rewards.

2. Preliminaries

Maximum Entropy Reinforcement Learning. The Reinforcement Learning problem can be formulated as a Markov decision process (MDP) (Sutton & Barto, 1998) which is defined by a tuple $< \mathcal{S}, \mathcal{A}, p, r, \gamma >$ where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the continuous action space, $p : \mathcal{S} \times \mathcal{A} \to \mathbb{P}_{\mathcal{S}}$ is the state transition distribution, $r : \mathcal{S} \times \mathcal{A} \to [r_{\min}, r_{\max}]$ is a bounded reward emitted by the environment, and finally γ is the discount factor. Maximum entropy RL adds an entropy term, $\mathbb{H}(\pi(\cdot|s))$, as a regularizer (Ziebart et al., 2008)

[*]Equal contribution [1]McGill University [2]Mila. Correspondence to: Patrick Nadeem Ward <patrick.ward@mail.mcgill.ca>, Ariella Smofsky <ariella.smofsky@mail.mcgill.ca>.

First workshop on *Invertible Neural Networks and Normalizing Flows* (ICML 2019), Long Beach, CA, USA

and leads to a modified formulation of the policy gradient objective

$$J(\pi) = \sum_{t=0}^{T} \mathop{\mathbb{E}}_{\substack{s_t \sim p \\ a_t \sim \pi}} [r(s_t, a_t) + \alpha \mathbb{H}(\pi(\cdot|s))].$$

Soft Actor Critic. Soft Actor-Critic (SAC) is a policy gradient algorithm that combines several recent approaches in RL including function approximation using Neural Networks, Double Q-learning (Van Hasselt et al., 2016) and entropy-regularized rewards to produce an off-policy actor-critic algorithm that is both sample efficient and emphasizes exploration (Haarnoja et al., 2018).

SAC learns three functions; two value functions $Q_\phi^{\pi_\theta}(s,a)$ and $V_\psi^{\pi_\theta}(s)$ playing the role of critics, and the target policy π_θ, referred to as the actor. The Q_ϕ network can be learnt by minimizing a mean squared bootstrapped estimate (MSBE) while the value network, V_ψ, is learnt by minimizing the squared residual error. The policy in SAC is a reparametrized Gaussian with a squashing function: $a_t = f_\theta(s_t; \epsilon_t) = \tanh(\mu_\theta(s_t) + \sigma_\theta(s_t) \cdot \epsilon_t)$ where $\epsilon_t \sim \mathcal{N}(0, I)$. Finally, the policy parameters can be learnt by maximizing the boostrapped entropy regularized reward.

Stochastic Computation Graphs. Stochastic Computation Graphs (John Schulman, 2015) provide a formalism for estimating the gradient of a stochastic function with learnable parameters by constructing directed acyclic graphs (DAGs). Consider for example a continuous random variable Z parameterized by θ, where the objective is to minimize the expected cost $L(\theta) = \mathbb{E}_{z \sim p_\theta}[f(z)]$ for some cost function $f(z)$. The gradient is defined as $\nabla_\theta L(\theta) = \nabla_\theta \mathbb{E}_{z \sim p_\theta}[f(z)]$. The REINFORCE estimator uses the property $\nabla_\theta p_\theta(z) = p_\theta(z) \nabla_\theta \log(p_\theta(z))$ to re-write the gradient of the expected cost as $\nabla_\theta L(\theta) = \mathbb{E}_{z \sim p_\theta}[f(z) \nabla_\theta \log(p_\theta(z))]$ which we can estimate using Monte Carlo estimation (Williams, 1992). In cases where f is differentiable and p_θ can be *reparameterized* through a deterministic function, $z = g(\epsilon, \theta)$, a low variance gradient estimate can be be computed by shifting the stochasticity from the distributional parameters to a standardized noise model, ϵ (Kingma & Welling, 2013; Rezende et al., 2014). Specifically, we can rewrite the gradient computation as $\nabla_\theta L(\theta) = \mathbb{E}_\epsilon[\nabla_g f(g(\epsilon, \theta)) \nabla_\theta g(\epsilon, \theta)]$.

Normalizing Flows. Given a parametrized density a *normalizing flow* defines a sequence of invertible transformations to a more complex density over the same space via the change of variable formula for probability distributions (Rezende & Mohamed, 2015). Starting from a sample from a base distribution, $z_0 \sim p(z)$, a mapping $f : \mathbb{R}^d \to \mathbb{R}^d$, with parameters θ that is both invertible and smooth, the log density of $z' = f(z_0)$ is defined as $\log p_\theta(z') = \log p(z_0) - \log \det \left| \frac{\partial f}{\partial z} \right|$.

Here, $p_\theta(z')$ is the probability of the transformed sample and $\partial f / \partial x$ is the Jacobian of f. To construct arbitrarily complex densities a sequence of functions, a flow, is defined and the change of density for each of these invertible transformations is applied. Thus the final sample from a flow is given by $z_k = f_K \circ f_{K-1} ... \circ f_1(z_0)$ and its corresponding density can be determined simply by $\ln p_\theta(z_K) = \ln p(z_0) - \sum_{k=1}^{K} \ln \det \left| \frac{\partial f_k}{\partial z_{k-1}} \right|$. While there are many different choices for the transformation function, f, in this work we consider only RealNVP based flows as presented in (Dinh et al., 2016) due to their simplicity.

3. Method

Connection with SAC. We motivate our approach by observing the existing relationship between sampling an action in SAC and the change of variable formula for probability distributions. As outlined in Section 2, a sampled action in SAC is taken from a reparametrized Gaussian distribution after a squashing function is applied. The squashing function, tanh, is a bijective mapping, transforming a sample from a Gaussian distribution with infinite support to one with bounded support in $(-1, 1)$. Let $u \in \mathbb{R}^D$ be the sample from the Gaussian distribution conditioned on some state s, then the log density of the action is given by $\log \pi(a|s) = u(a|s) - \sum_{i=1}^{D} \log(1 - \tanh(u_i)^2)$, where the right term is the log determinant of the Jacobian of tanh applied component-wise. Consequently, the squashing function is a one layer Normalizing Flow.

Normalizing Flow Policies. We now present how to construct Normalizing Flow policies within the SAC framework. To define the policy, $\pi(a|s)$, we first encode the state using a neural network which is then used to define the base distribution z_0. In principle, z_0 can be any probability distribution and learning can use the general REINFORCE gradient estimator. However, choosing a distribution with an available reparametrization allows for empirically lower variance gradient estimates and increased stability, a fact we verify in Section 4. Similar to SAC we choose z_0 to be a Gaussian distribution whose parameters are defined by a state-dependent neural network. Unlike SAC, we do not apply a squashing function after sampling from z_0, but instead apply a sequence of invertible transformations that define the flow; after which we apply the squashing function. We use a modified RealNVP style invertible transformations to define each flow layer f_i. Thus, the final sampled action is given by $\pi(a|s) = \tanh(f_k \circ ... \circ f_1(\mu_\theta(s) + \sigma_\theta(s) \cdot \epsilon))$. Let $u(a|s) = z_0$ be the sampled intermediate action from the reparametrized Gaussian. The corresponding log density of a, under the Normalizing Flow is simply $\log \pi(a|s) = \log p(z_0) - \sum_{i=1}^{k} (\log \det \left| \frac{\partial f_k}{\partial z_{i-1}} \right|) - \sum_{i=1}^{D} log(1 - \tanh(z_{k_i})^2)$. As the final output action has a well defined density it is easy to

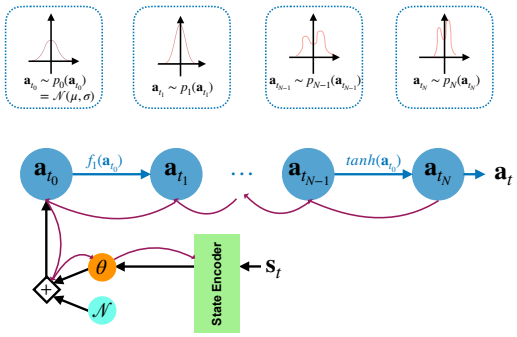

Figure 1. A sampled action from a Normalizing Flow policies

compute the entropy of the policy through sampling. This fact allows Normalizing Flow policies to be a convenient replacement for Gaussian policies within SAC and maximum entropy RL.

Stabilizing Tricks. While the formulation above is not the only choice for defining a Normalizing Flow policy it is significantly more stable than the one presented in (Tang & Agrawal, 2018). We implement the state encoder as a reparametrized Gaussian which consistently leads to stable behavior after applying a few simple tricks that we now detail. We apply a series of tricks taken from the generative modeling literature to further stabilize training. Specifically, we use weight clipping for the policy parameters to enforce 1-Lipschitzness as demonstrated in Wasserstein GAN (Arjovsky et al., 2017). From a practical perspective weight clipping prevents any parameter in the policy from overflowing due to numerical precision. We hypothesize such phenomena is exacerbated in the RL setting because using a more expressive policy enables sampling extreme values before the squashing function. Furthermore, RealNVP uses a BatchNorm layer (Ioffe & Szegedy, 2015) as part of its flow. Empirically, we observe the log standard deviation of the batch can also overflow. Thus we omit the BatchNorm layer completely from our flow.

4. Experiments

We investigate the limitations of imposing a fixed family of probability distributions for continuous control tasks in Reinforcement Learning [1]. To this end, we compare various implementations of SAC that modify the family of distributions used. Through our experiments we seek to answer the following questions:

(Q1) What is the impact of picking a specific family of distributions on performance?

(Q2) What is the effect of different gradient estimators on the learned policy and return?

(Q3) How does exploration behavior change when using a more expressive class of policy distributions?

Setup and Environment. We compare variations of SAC that include modeling the policy distribution by a Normalizing Flow, a factored Gaussian (the current method used in practice) and different reparametrizable distributions such as the Exponential distribution. We also consider two variations of parameter updates; the REINFORCE update and the pathwise gradient estimator. The chosen RL environment is a continuous two-dimensional grid world introduced by (Oh et al., 2017). The goal of the continuous grid world task is for the agent, whose trajectory begins at a fixed starting point, to maximize its discounted reward for the duration of a 500 timestep episode. We experiment on two versions of this environment; a continuous grid world of size 150×150 units with sparse and dense rewards. Both environments have the same start state indicated by the green point and grey walls that the agent cannot pass as shown in Fig. 2. The sparse reward environment is composed of a single goal state of value 100.0, the red point, whereas the dense reward environment includes additional sub-goal states of value 5.0 indicated as yellow points. These rewards are per timestep, meaning an agent can stay in a reward state until the end of the episode consisting of 500 timesteps.

Results. We now address questions (**Q1-Q3**).

Q1. Empirically we observe that the choice of distribution significantly impacts model performance Fig. 3. In dense reward settings, the Gaussian policy consistently converges quickly to an optimal policy while on sparse rewards it fails to find a solution within the first $100k$ steps. The Exponential policy does significantly worse than the Gaussian and only begins to learn near the end of training, indicating the choice of distribution does matter for this task. Overall, we find that the Normalizing flows achieves a reasonable trade-off when considering both environments since it manages to get comparable performance to the Gaussian policy in the dense rewards settings but, more importantly, performs significantly better, i.e. reaching the goal state before any other distribution, when run with sparse rewards.

Q2. In all of our experiments, we observe that the REINFORCE gradient estimate leads to higher variance in performance Fig. 3. However both REINFORCE and reparametrized policies converge to a similar reward.

Q3. In Dense reward settings all three policy variations converge to the first sub-goal Fig. 2. We notice the Normalizing Flow policy does not explore the environment far beyond the first and second sub-goals while both the Gaussian Policies tend to explore regions where no rewards are present. In Sparse reward settings we see that Normalizing Flow policies explore regions close to the goal state while

[1] Code: https://github.com/joeybose/FloRL

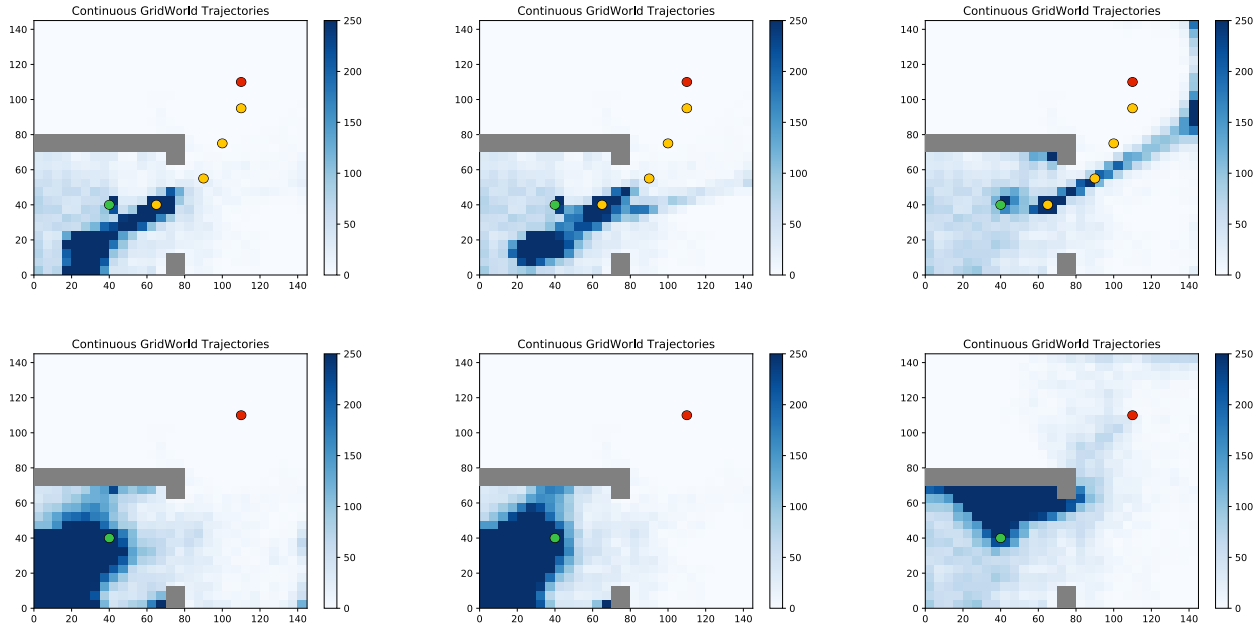

Figure 2. Heatmaps for various policies in SAC on Dense (**Top**) and Sparse (**Bottom**) reward settings. The grey areas indicate impassable walls. The start and goal states are shown as green and red points respectively. Sub-goal states are depicted as yellow points. **Left** Gaussian Policy with REINFORCE. **Middle** Gaussian Policy with Reparametrization. **Right** Normalizing Flow Policy

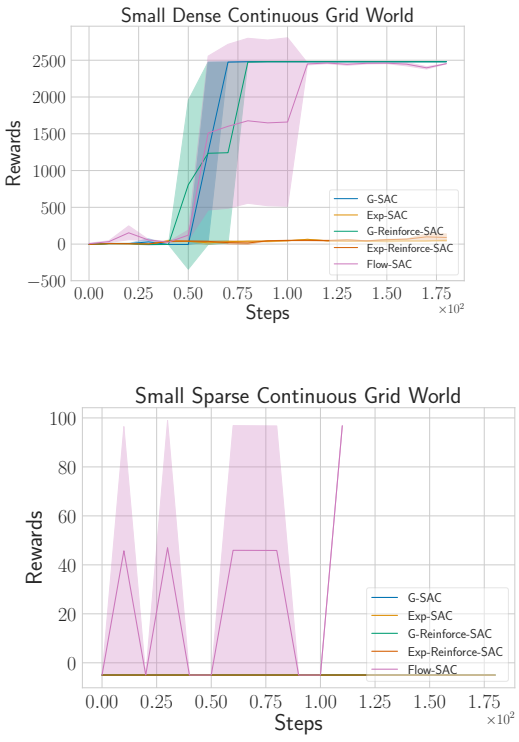

Figure 3. Performance of SAC with various policies and gradient estimators on Continuous Grid World. All runs are averaged over three runs and the shaded region represents one standard deviation.

both Gaussian policies only explore locally and far from the goal state, as substantiated in Fig. 3.

5. Discussion and conclusion

We introduce Normalizing Flow policies within the maximum entropy RL framework. Specifically, we replace Gaussian policies in Soft Actor Critic with modified RealNVP flows. In essence, Normalizing Flow policies have the ability to model more expressive classes of policies while maintaining the benefit of Gaussian policies, i.e. easy to sample from and having a defined probability density. We also present a few stabilizing tricks that enable training Normalizing Flow policies in the RL setting. Empirically, we observe that our approach has the ability to explore sparse reward continuous gridworld settings in a more efficient manner than Gaussian or Exponential policies, in some cases finding the optimal reward almost immediately. Additionally, our Normalizing Flow approach comes with minimal added cost on dense reward settings since our approach converges to the optimal reward at a rate similar to the Gaussian policy. In terms of directions for future work, one important design decision is the choice of architecture when defining the Normalizing Flow. In this work, we considered Flows based on the RealNVP architecture but applying more recent variants such as Glow (Kingma & Dhariwal, 2018), NAF (Huang et al., 2018), and FFJORD (Grathwohl et al., 2018) to the RL setting is a natural direction for future work.

References

Arjovsky, M., Chintala, S., and Bottou, L. Wasserstein gan. *arXiv preprint arXiv:1701.07875*, 2017.

Chou, P.-W., Maturana, D., and Scherer, S. Improving stochastic policy gradients in continuous control with deep reinforcement learning using the beta distribution. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 834–843. JMLR. org, 2017.

Dinh, L., Sohl-Dickstein, J., and Bengio, S. Density estimation using real nvp. *arXiv preprint arXiv:1605.08803*, 2016.

Grathwohl, W., Chen, R. T., Betterncourt, J., Sutskever, I., and Duvenaud, D. Ffjord: Free-form continuous dynamics for scalable reversible generative models. *arXiv preprint arXiv:1810.01367*, 2018.

Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. *arXiv preprint arXiv:1801.01290*, 2018.

Heess, N., Wayne, G., Silver, D., Lillicrap, T., Erez, T., and Tassa, Y. Learning continuous control policies by stochastic value gradients. In *Advances in Neural Information Processing Systems*, pp. 2944–2952, 2015.

Huang, C.-W., Krueger, D., Lacoste, A., and Courville, A. Neural autoregressive flows. *arXiv preprint arXiv:1804.00779*, 2018.

Ioffe, S. and Szegedy, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

John Schulman, Nicolas Heess, T. W. P. A. Gradient estimation using stochastic computation graphs. In *Advances in Neural Information Processing Systems*, pp. 3528–3536, 2015.

Kingma, D. P. and Dhariwal, P. Glow: Generative flow with invertible 1x1 convolutions. In *Advances in Neural Information Processing Systems*, pp. 10215–10224, 2018.

Kingma, D. P. and Welling, M. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Oh, J., Singh, S., and Lee, H. Value prediction network. In *Advances in Neural Information Processing Systems*, pp. 6118–6128, 2017.

Rezende, D. J. and Mohamed, S. Variational inference with normalizing flows. *arXiv preprint arXiv:1505.05770*, 2015.

Rezende, D. J., Mohamed, S., and Wierstra, D. Stochastic backpropagation and approximate inference in deep generative models. *31st International Conference on Machine Learning*, 2014.

Schulman, J., Heess, N., Weber, T., and Abbeel, P. Gradient estimation using stochastic computation graphs. In *Advances in Neural Information Processing Systems*, pp. 3528–3536, 2015.

Sutton, R. S. and Barto, A. G. *Reinforcement learning - an introduction*. Adaptive computation and machine learning. MIT Press, 1998. ISBN 0262193981. URL http://www.worldcat.org/oclc/37293240.

Tang, Y. and Agrawal, S. Boosting trust region policy optimization by normalizing flows policy. *arXiv preprint arXiv:1809.10326*, 2018.

Van Hasselt, H., Guez, A., and Silver, D. Deep reinforcement learning with double q-learning. In *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

Williams, R. J. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 1992.

Ziebart, B. D. Modeling purposeful adaptive behavior with the principle of maximum causal entropy. 2010.

Ziebart, B. D., Maas, A. L., Bagnell, J. A., and Dey, A. K. Maximum entropy inverse reinforcement learning. In *Aaai*, volume 8, pp. 1433–1438. Chicago, IL, USA, 2008.