Maximum Entropy Reinforcement Learning With Evolution Strategies

Longxiang Shi[†], Shijian Li[†], Qian Zheng[‡], Longbing Cao[§], Long Yang[†], Min Yao[†], Gang Pan[†]
[†]*College of Computer Science and Technology, Zhejiang University, China*
[‡]*School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore*
[§]*Advanced Analytics Institute, University of Technology Sydney, Australia*
{shilongxiang,shijianli,yanglong,myao, gpan}@zju.edu.cn,
zhengqian@ntu.edu.sg, longbing.cao@uts.edu.au

***Abstract*—Evolution strategies (ES) have recently raised attention in solving challenging tasks with low computation costs and high scalability. However, it is well-known that evolution strategies reinforcement learning (RL) methods suffer from low stability. Without careful consideration, ES methods are sensitive to local optima and are unstable in learning. Therefore, there is an urgent need for improving the stability of ES methods in solving RL problems. In this paper, we propose a simple yet efficient ES method to stabilize the learning. Specifically, we propose a framework to incorporate the maximum entropy reinforcement learning with evolution strategies and derive an efficient entropy calculation method for linear policies. We further present a practical algorithm called maximum entropy evolution policy search based on the proposed framework, which is efficient and stable for policy search in continuous control. Our algorithm shows high stability across different random seeds and can obtain comparable results in performance against some existing derivative-free RL methods on several of the well-known benchmark MuJoCo robotic control tasks.**

***Index Terms*—reinforcement learning, evolution strategies, policy search**

I. Introduction

In recent years, evolution strategies (ES) have raised attention in solving challenging tasks of modern reinforcement learning (RL) with low computation cost and high scalability [1]–[3]. As a powerful alternative approach to the conventional Markov Decision Process (MDP) based methods [4], ES methods optimize the RL objective by regarding the environment as a "blackbox", and use the perturbed policies to explore the environment.

One popular version of ES methods [1] optimizes the *Gaussian smoothing* of the RL objective function:

$$J(\theta) = \mathbb{E}_{\epsilon \in N(0,I)}[J(\theta + \sigma\epsilon)],$$

where $J(\theta)$ is the objective function which represents the expected total reward obtained from the environment, θ encodes the policy, $\epsilon \in N(0, I)$ is a random variable drawn from Gaussian distribution and $\sigma > 0$ is a hyperparameter denoting the exploration range. The gradient of the objective function w.r.t. θ can be estimated using N sampled ϵ:

Corresponding author: Shijian Li (shijianli@zju.edu.cn).

$$\nabla J(\theta) = \frac{1}{\sigma N} \sum_{i=1}^{N} J(\theta + \sigma\epsilon_i)\epsilon_i$$

The policy can be optimized by performing gradient ascent with the sampled data from the environments. However, one major drawback for ES methods is the instability and high variance in learning. The ES gradient estimator may introduce high variance and without careful consideration, ES methods are brittle to local optima and are fragile in learning [5].

In order to improve the stability of ES methods, one common way is to reduce the estimation variance in ES gradient estimator. For example, control variables (also known as baseline functions) are often used in ES-based methods [6]–[8]. Alternatively, general Monte Carlo methods such as antithetic sampling are also popular adopted in blackbox optimization methods [1], [2]. Those methods can substantially improve the performance of ES. However, most of them are not stable and easy to trap in the local optima when facing high dimensional, complex tasks [9], [10]. New sampling methods such as Quasi Monte Carlo methods and orthogonal sampling [11] can obtain robust results, but need extra computation cost in generating perturbation samples, which are not efficient when dealing with high-dimensional tasks. Moreover, reusing the trajectories generated by ES with off-policy deep reinforcement learning algorithms are also adopted in stable the ES algorithm. Prominent examples such as evolution reinforcement learning (ERL) [12], CEM-RL [13] improve the stability of ES with the cost of extra computation cost and low scalability.

Recently, several works based on maximum entropy reinforcement learning show that optimizing a non-deterministic policy with maximum entropy regularizer can also improve the stability and robustness of RL algorithms [14], [15]. The involvement of maximum entropy RL can improve exploration by introducing diverse behaviours to the policy and can substantially improve the stability in learning. However, for ES methods the maximum entropy reinforcement learning framework is seldom used.

Inspired by those prominent works, in this paper we attempt to incorporate evolution strategies with maximum entropy reinforcement learning. Specifically, we propose a maximum re-

inforcement learning framework for evolution strategies, which enables maximum entropy reinforcement learning in evolution strategies. We also derive an efficient entropy calculation method for linear policies. In addition, we propose a practical algorithm called *maximum entropy evolution policy search*, which can retain the efficiency and scalability of ES methods. We evaluate our method on the MuJoCo continuous control benchmark environments [16]. The empirical results show that our method is capable for dealing with high-dimensional control tasks, and is very stable across different random seeds, and can obtain competitive results in performance against some of the existing methods.

To summarize, our contributions are as follows:

- We propose a maximum entropy reinforcement learning framework, which additionally considers the maximum entropy regularizer to ES methods.
- We present a practical policy search algorithm for continuous control based on the proposed framework, which improves the performance and stability of the existing ES methods.

II. Related Works

Here, we summarized some related works, including the stability improvement in evolution strategies and maximum entropy reinforcement learning.

A. Stability Improvement in Evolution Strategies

Existing methods to improve the stability of ES for policy search can be divided into two categories:

Variance reduction of ES gradient estimator: ES gradient estimator are always high variance in practice, which may cause ES unstable and can easily breakdown when facing a stochastic environments. Therefore, many existing works have studied the variance reduction of ES gradient estimator. One common approach is to use control variables (also refer to baselines). The control variables can be either learned or calculated during learning. Such method can be found in [6], [7], [17] and [8]. In addition, methods for general purpose Monte Carlo approach are also widely used, for example antithetic sampling is very popular among the ES policy search methods [1], [2], [7]. Although those methods can achieve a considerable performance on RL tasks, most of them are not stable and easy to trap in the local optima when facing high dimensional tasks [9], [10]. Recently, [11] proposes variance reduction through Quasi Monte Carlo sampling and orthogonal sampling. [18] propose a sampling method based on geometrically coupling. However, those methods involve extra computation cost in generating samples, which are not efficient when dealing with high-dimension control tasks.

Reuse of the ES generated samples: Several methods attempt to increase the stability of ES by reusing the samples generated by ES. The incorporation of off-policy methods to re-utilize the samples generated by ES can improve the stability of ES. For example, In [19], They propose POWER algorithm, which uses expectation-maximization algorithm for calculating the policy update. However, the involvement of importance sampling may introduce high variance in estimation. Evolution RL (ERL) [12] introduces a hybrid algorithm that periodically inserts the deep deterministic policy gradient [20] agent to the evolution optimization process, and improves the stability and efficiency in learning and exploration. The goal exploration process-policy gradient (GEP-PG) [21] adopts a goal exploration process to fill the replay buffer and then uses DDPG [20] to learn the policies. [5] proposes robust blackbox optimization(RBO), which uses off-policy samples to improve the stability of ES and generalize the orthogonal sampling. In addition, the CEM-RL [13] combines TD3 to ES methods to relearn the samples. Their work, the cross-entropy method (CEM) and gradient-based method DDPG/TD3 [22] are combined. Their result shows that stability of ES can be improved. In [23], they combine ES with TRPO [24] to make monotonic update the policies.

To summarize, although those sample-based methods can improve the data efficiency and stability of ES, they are always computation costly in gradient computing and not easy to parallelize.

B. Maximum Entropy Reinforcement Learning

While the conventional RL optimizes the policies to maximize the expected return, maximum entropy RL aims at maximizing both the expected return and the expected entropy of the policy [25]. The maximum entropy RL can provide a substantial improvement of exploration and robustness comparing to conventional RL [15], [25]. Such a framework has been used in inverse RL [26], optimal control [27], [28], and guided policy search [29], [30].

Recently, several methods use deep RL combined with maximum entropy RL. [14] proposes soft Q-learning algorithms that uses a stochastic sampling network to perform exploration, and can cope with arbitrary policy distributions. Subsequently, the Soft actor-critic [15] adopts maximum entropy RL in actor-critic algorithms. In addition, [31] proposes Trust-PCL to utilize the entropy regularizer in TRPO [24] and improves the stability and robustness of TRPO.

Although the state-of-the-art maximum entropy RL methods achieve prominent results with high stability, they have not combined with ES framework, since ES always optimize a deterministic policy and there is no sense for measuring the entropy of a deterministic policy.

III. Preliminary

We first give some basic notations and preliminaries on evolution methods for RL and maximum entropy RL.

A. Notations

A reinforcement learning problem can be formulated as a Markovian decision process (MDP) [4]: (S, A, γ, R, P), where S is the state space, A is the action space, $\gamma \in [0, 1]$ is the discount factor, R is the reward function: $R : S \times A \to \mathbb{R}$, P is the transition probability denotes the probability density of the next state $s_{t+1} \in S$ when executing action $a \in A$ under

observation $s_t \in S$: $S \times S \times A \rightarrow [0, 1]$. A policy π is a probability distribution that describes the agent action $a \in A$ under state $s \in S$: $\pi(a|s) : S \times A \rightarrow [0, 1]$. We use ρ_π to refer to the state distributions under the policy π.

The return of policy π of one rollout is defined as the total discount reward achieved when executing π in the environment:

$$G(\pi) = \sum_{i=0}^{T} \gamma^i R(s_t, a_t) \tag{1}$$

The goal of RL is to find an optimal policy that maximizes the expected discounted return:

$$J(\pi) = \mathbb{E}_{(s_t,a_t)\sim\rho_\pi} G(\pi) \tag{2}$$

In this paper, we will focus on linear policies, where the action $a \in \mathbb{R}^{n\times 1}$ can be calculated through a parameterized policy matrix $\theta \in \mathbb{R}^{m\times n}$ under the observation $s \in \mathbb{R}^{m\times 1}$:

$$a = \theta^T \cdot s \tag{3}$$

B. Evolution Strategies for Reinforcement Learning

Evolution strategies optimize the policy by regarding the environment as a blackbox. With Gaussian smoothing [11], the RL optimization objective can be rewrite as:

$$J_\sigma(\theta) = \mathbb{E}_{\epsilon\sim\mathcal{N}(0,I)}[J(\theta + \sigma\epsilon)] \tag{4}$$

where σ is hyperparameter about exploration range, and $\epsilon \sim \mathcal{N}(0, I)$ is a Gaussian random variable. The gradient of parameter θ of policy can be calculated by:

$$\nabla J(\theta) = \frac{1}{\sigma}\mathbb{E}_{\epsilon\sim\mathcal{N}(0,I)}[J(\theta + \sigma\epsilon)] \tag{5}$$

The above gradient estimator can be estimated using a Monte Carlo estimator by sampling N samples from a standard normal distribution:

$$\nabla J(\theta) = \frac{1}{\sigma N} \sum_{i=1}^{N} G(\pi_{\theta+\sigma\epsilon_i})\epsilon_i \tag{6}$$

Additionally, $G(\pi)$ can be also evaluated by executing policy π in the environment. We refer the gradient estimator in 6 as vanilla ES estimator. The policy can be then optimized by performing gradient ascent with estimated gradient:

Usually, antithetic sampling is used to reduce variance:

$$\nabla J(\theta) = \frac{1}{2\sigma N} \sum_{i=1}^{N} [G(\pi_{\theta+\sigma\epsilon_i}) - G(\pi_{\theta-\sigma\epsilon_i})]\epsilon_i \tag{7}$$

With the ES gradient estimator, the policy can be optimized through executing the perturbed policies into the environment.

C. Maximum Entropy Reinforcement Learning

In the maximum entropy RL, an entropy regularizer is augmented to the standard RL objective function:

$$J_e(\pi) = \mathbb{E}_{s_t\sim\rho_\pi}[\sum_{i=0}^{T} \gamma^i R(s_t, a_t) + \alpha H(\pi(\cdot|s_t)] \tag{8}$$

where α is a temperature parameter denoting the relative importance of the entropy regularizer with regard to the reward [26], and $H(\pi(\cdot|s_t))$ is the entropy over policy π under observation s_t. The temperature parameter α controls the stochasticity of the optimal policy [15]. Although in principle $1/\alpha$ can be folded into the reward function [14], for convenient we will treat α as a fixed hyper-parameter in learning.

Note that the maximum entropy RL framework can only used for non-deterministic polices since the entropy for deterministic policies is always 0. The involvement of an entropy regularizer in the optimization objective can incentive the policy to explore more widely while giving up on clearly unpromising choices [15]. The non-deterministic policy can also capture the multiple modes of near-optimal behaviours, which can reduce the difficulty in finding one of them. Moreover, prior work has observed the stability of learning can be substantially improved by the entropy regularizer [14], [15]. While the prior work of maximum entropy RL focuses on derivative-based methods, we here focus on derivative-free method using a maximum entropy learning framework, especially for the evolution strategies.

IV. Method

In this section we introduce our method in detail. We will first describe the overall framework, give derivations of entropy calculation, and introduce maximum entropy evolution policy search algorithm in detail.

A. Overall Framework

As illustrated above, the conventional ES for RL optimizes a deterministic policy during learning, which is not able to apply maximum entropy RL framework. To address this issue, in this work we explore to optimize a non-deterministic policy with parameters under Gaussian distribution. For each iteration, Gaussian random variables are generated, and are then used to generate non-deterministic policies. For each non-deterministic policies, we sample again using its parameter distribution to estimate its expected return with the maximum entropy regularizer, and then update the policy parameter using the ES gradient estimator. The overall framework is shown in Fig. 1. We will describe the optimization of non-deterministic policy, and derive the entropy calculation in the following part.

1) Optimizing a Non-deterministic Policy Using Evolution Strategies: We seek to optimize a non-deterministic policy whose parameters are under Gaussian distribution:

$$\hat{\pi}(\theta) : \theta \sim \mathcal{N}(\mu, v^2 I) \tag{9}$$

Here μ is the mean value matrix of parameters, v is the standard deviation that describes the degree of determinacy

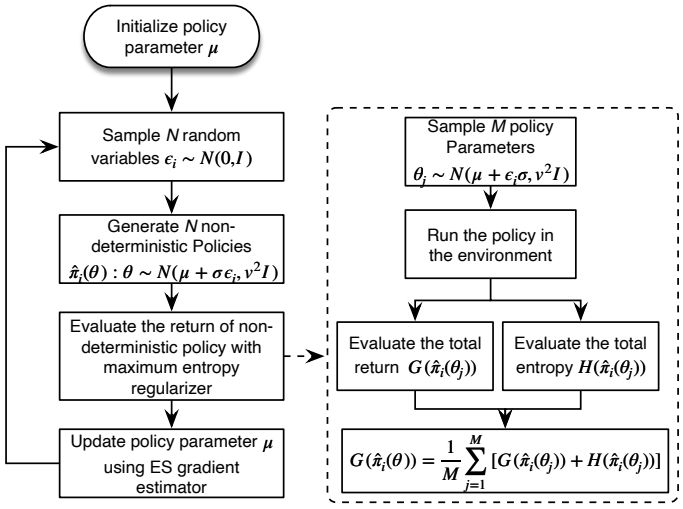


Fig. 1. Overall framework. Comparing to the conventional ES methods, our framework optimize a non-deterministic policies with Gaussian distribution, together with the maximum entropy regularizer. For each time step of evaluation, we also evaluate the entropy of policies when encountering an observation.

of policy π and I is the identity matrix. If v is close to 0, then the policy is become more deterministic, and vice versa. Similarly, the goal of learning a non-deterministic policy is to maximize the expected discounted return, as described in equation 2.

Based on equation 1, the return of policy $\hat{\pi}$ can be estimated using Monte Carlo sampling: we can sample M deterministic policies based on the parameter distribution of policy $\hat{\pi}$, and then evaluate them into the environments:

$$G(\hat{\pi}) = \frac{1}{M}\sum_{j=1}^{M}\sum_{t=1}^{T}\gamma^i R(s_t, a_t|\pi_j) \tag{10}$$

If we treat the policy variance v as a hyperparameter, then based on equation 6, the gradient of parameter μ can be estimate using the ES gradient estimator as:

$$\nabla J(\mu) = \frac{1}{N\sigma}\sum_{i=1}^{N} G(\hat{\pi}(\mu + \sigma\epsilon_i))\epsilon_i \tag{11}$$

Note that $\hat{\pi}(\mu+\sigma\epsilon_i)$ indicates that the parameters in policy $\hat{\pi}$ is under Gaussian distribution:

$$\hat{\pi}(\theta) : \theta \sim \mathcal{N}(\mu + \delta\epsilon_i, v^2 I) \tag{12}$$

The antithetic sampling technique can also be used using the following equation:

$$\nabla J(\mu) = \frac{1}{2N\sigma}\sum_{i=1}^{N}[G(\hat{\pi}(\mu + \sigma\epsilon_i)) - G(\hat{\pi}(\mu - \sigma\epsilon_i))]\epsilon_i \tag{13}$$

Hence, we have derived a ES optimization method for non-deterministic policies. The involving of non-deterministic policies can encourage the policy to explore more, as the sampled policies are different at each gradient update step.

2) Efficient Entropy Calculation for Linear Policies: The involving of non-deterministic policies make it possible for evaluation of entropy during learning. To calculate the entropy of policies, we need first analyze the probability distribution of actions from a non-deterministic policy $\hat{\pi}$. As the parameters in policy $\hat{\pi}$ is under Gaussian distribution and we use linear policies, the output action is a linear transformation of $\hat{\pi}$. Based on the property of Gaussian distribution, the probability distribution of output action of policy $\hat{\pi}$ under some state s is also a multivariate Gaussian distribution [32]:

$$\hat{\pi}(a|s) \sim \mathcal{N}((\mu + \delta\epsilon_i)^T \cdot s, v^2 \cdot I \cdot s^T \cdot s) \tag{14}$$

The entropy of the policy can be calculated using the following theorem:

Theorem: Given the multivariate Gaussian policy $\hat{\pi}$, where the parameters in $\hat{\pi}(\theta) \sim \mathcal{N}(\mu, v^2 \cdot I)$, the entropy of policy $\hat{\pi}$ under state s $H(\hat{\pi}(\cdot|s))$ is:

$$H(\hat{\pi}(\cdot|s)) = \frac{n}{2} + \frac{n}{2}\log(2\Pi) + \frac{1}{2}\log|v^2 \cdot I \cdot s^T \cdot s| \tag{15}$$

where Π is the Archimedes' constant.

Proof: Denoting $f(a)$ as the probability density function of output action under policy $\hat{\pi}(\theta) : \theta \sim \mathcal{N}(\mu + \delta\epsilon_i, v^2 \cdot I)$, the entropy of stochastic policy $\hat{\pi}$ under state s can be calculated through a multiple-integral over all the state dimensions:

$$H(\hat{\pi}(\cdot|s)) = -\underbrace{\int_{-\infty}^{+\infty}\dots\int_{-\infty}^{+\infty}}_{n} f(a)\log f(a)\mathrm{d}a|_s \tag{16}$$

According to the theory of differential entropy of multivariate Gaussian distribution [31], the result of the above multiple-integral only depends on the covariance matrix of the output action distribution:

$$H(\hat{\pi}(\cdot|s)) = \frac{n}{2} + \frac{n}{2}\log(2\Pi) + \frac{1}{2}\log|v^2 \cdot I \cdot s^T \cdot s| \tag{17}$$

■

The above equation states that the entropy of a Gaussian policy can be calculated directly. For each rollout of ES, the total entropy of policy $H(\hat{\pi}(\theta))$ can be calculated as:

$$H(\hat{\pi}(\theta)) = \sum_{i=0}^{T} H(\hat{\pi}(\cdot|s_t, \theta)) \tag{18}$$

which is determined by the states the agent encountered. Hence, we thus obtain an efficient entropy calculation method of the policies generated by ES.

Ignoring the constant terms, we can rewrite the return in Equation 10 with a maximum entropy regularizer:

$$G_e(\hat{\pi}) = \frac{1}{M}\sum_{j=1}^{M}\sum_{t=1}^{T}\gamma^i[R(s_t, a_t|\pi_j) + \alpha\log|v^2 \cdot I \cdot s_t^T \cdot s_t|] \tag{19}$$

We can then use the sampling technique to estimate the gradient of parameters in policy $\hat{\pi}$ using the ES gradient estimator

in Equation 6 and 7. So far we have derived the maximum entropy reinforcement learning method using ES, we name it *Maximum Entropy Evolution Policy Search* (MEPS).

B. Practical Algorithm

Since the ES gradient estimator is always high variance, in practice we need to use some techniques to reduce the variance in learning. Similar to some existing ES methods for policy search [1], [2], [7], We use several techniques to reduce the variance and increase the stability in learning:

1) State Normalization. State normalization is important to treat each state features with equal influence and can also increase the stability during learning. We here use the z-score normalization for the state encountered.
2) Antithetic sampling and using the top directions.. Antithetic sampling can improve the robustness in ES gradient estimation [7]. In addition, using the top directions of antithetic sampling can reduce noise in gradient update as discussed in [2].
3) Reward Scaling. Reward scaling can reduce the variance in antithetic sampling and make the step size less sensitive to the gradient update of the policy parameters [2]. In this paper we scale the total reward received by dividing its standard deviation.

Finally, the overall algorithm of MEPS is illustrated in algorithm 1. The MEPS is as efficient as the the current ES methods such as [1] and [2], as we just need to calculate entropy based on the states during each rollouts. Our method is simple and efficient comparing to the orthogonal sampling methods [11], and can also researve the high scalability of ES.

Algorithm 1 MEPS: Maximum Entropy Evolution Policy Search

Input: learning rate lr, number of directions sampled per iteration N, exploration range σ, number of top-performing directions to use $b(b \leq N)$, number of sampling M in evaluate stochastic policies, policy varince v and temperature parameter α.
Initialize: $\mu_0 = 0 \in \mathbb{R}^{m \times n}, j = 0$
while ending condition not satisfied **do**
  Sample $\epsilon_1, \epsilon_2, ..., \epsilon_N$ with i.i.d. standard normal entries.
  Generate stochastic policies using the sampled variables:

$$\begin{cases} \hat{\pi}_{i,+}(\theta)|_j \sim \mathcal{N}(\mu_j + \sigma\epsilon_i, v^2 I) \\ \hat{\pi}_{i,-}(\theta)|_j \sim \mathcal{N}(\mu_j - \sigma\epsilon_i, v^2 I) \end{cases}$$

  for $\hat{\pi} \in \{\hat{\pi}_{i,+}|_j, \hat{\pi}_{i,-}|_j\}$ **do**
    Sample M parameters according to the policy distribution of $\hat{\pi}$.
    Generate M rollouts with sampled policy parameters. Note that the state need to be normalized before calculating action.
    Evaluate $G_e(\hat{\pi})$ based on Equation 10.
  end for
  Sort the directions by $\max\{G_e(\hat{\pi}_{i,+}|_j), G_e(\hat{\pi}_{i,-}|_j)\}$ and using the top b corresponding ϵ for update. Denoting $\epsilon_{(l)}$ as the l-th largest direction.
  Update the policy via:

$$\mu_{j+1} \leftarrow \mu_j + \frac{lr}{b\sigma\delta_R} \sum_{l=1}^{b} [G_e(\hat{\pi}_{l,+}|_j) - G_e(\hat{\pi}_{l,-}|_j)]\epsilon_{(l)}$$

  where δ_R is the standard deviation of the rewards collected of the rollouts.
  $j \leftarrow j + 1$
end while

V. Experimental Results

In this section, we conduct experiments to evaluate the performance of maximum entropy evolution policy search to answer the following questions:

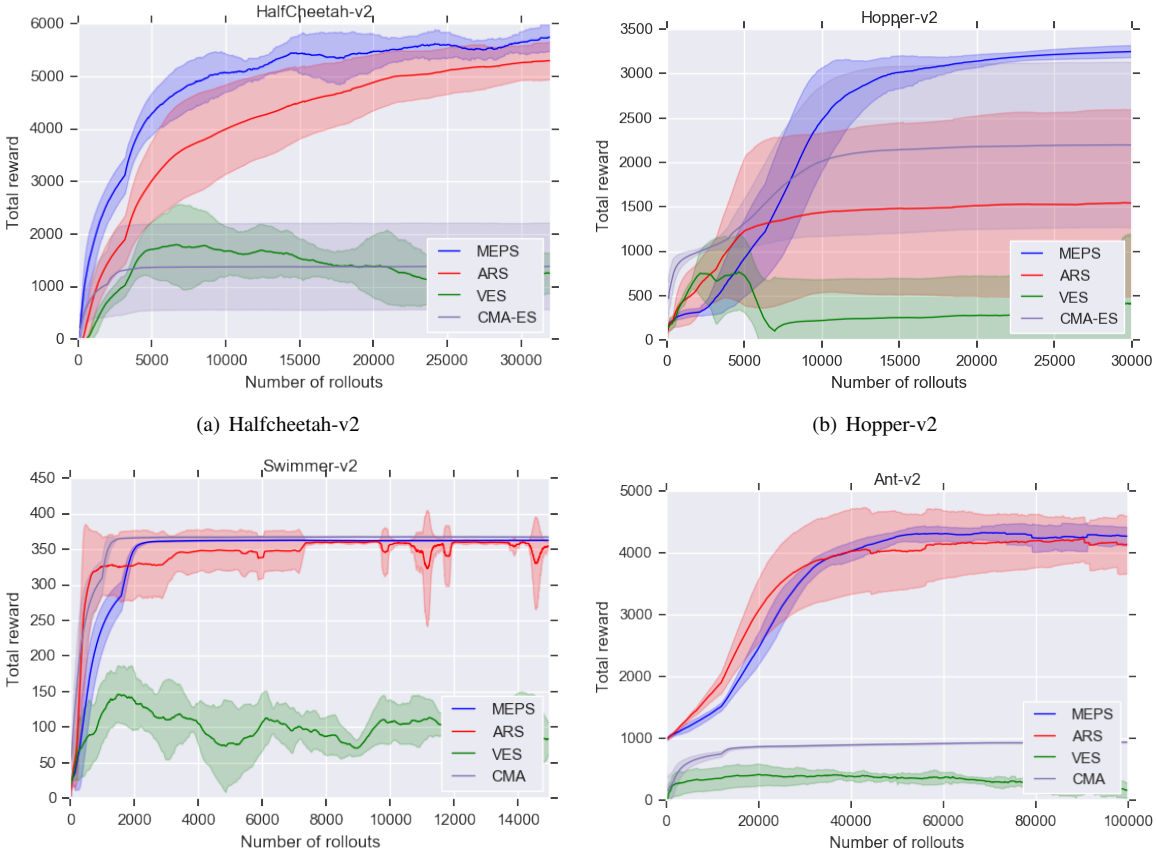
1) Can maximum entropy evolution policy search improve the performance and stability against the existing methods on the typical RL benchmark environments?
2) How sensitive is MEPS to the new involved hyperparameters?

To answer the above questions, we evaluate our method against some existing methods on the well-known benchmark MuJoCo robotic control tasks [16]. Specifically, we use four tasks: HalfCheetah-v2, Hopper-v2, Swimmer-v2 and Ant-v2. The details of the experiments can be found in Appendices part. The evaluation is performed on the OpenAI Gym [33]. The detail of experiment results is described below.

A. Performance Evaluation

We evaluate the performance of MEPS against Augment Random Search (ARS), Vanilla ES (VES) and CMA-ES on the MuJoCo continuous control tasks. We run each method for 5 times with fixed random seeds and measure score of total average reward achieved during learning. The score is estimated by running the policy without exploration noise after a gradient update step. To make a fair comparison, for each running the 3 methods share the sample random seed. In addition, the results are also smoothed with a right-centered moving window of 50 consecutive epochs. The hyperparameters of our method, vanilla ES and CMA-ES is described in the Appendices, while the hyperparameters of ARS we used is the same as ARS-vt in [2].

The learning curves is shown in Figure 2. The solid lines represent the mean total reward obtained under 5 independent runs and the shaded regions represents the variance. The experiment results show that our method is stable over different random seeds than the baseline methods ARS, VES. Specifically, in HalfCheetah-v2, Hopper-v2 and Ant-v2 the MEPS outperforms the baseline methods by obtaining more rewards per episode. The CMA-ES method outperforms the other methods on the Swimmer-v2. However, it performs poorly on high-dimension tasks such as Ant-v2 (888 control dimension), HalfCheetah-v2 (106 control dimension), as is reported in [24]. In addition, in Swimmer-v2, the original ARS is not stable after convergence, in contrast our method is very stable after convergence. We can empirically conclude that



(a) Halfcheetah-v2

(b) Hopper-v2

(c) Swimmer-v2

(d) Ant-v2

Fig. 2. Learning curves on the MuJoCo [16] tasks of maximum entropy evolution policy search (MEPS), augment random search (ARS), vanilla evolution stategies (VES) and Covariance Matrix Adaption (CMA-ES). The x-axis denotes the number of rollouts during the learning and y-axis represents the total reward achieved. The solid lines represent the mean total reward obtained under 5 independent runs and the shaded regions represents the variance. Our method achieves a stable and comparable results across different random seeds than the baseline methods.

our method can improve the performance and robustness of evolution strategies.

B. Ablation Study

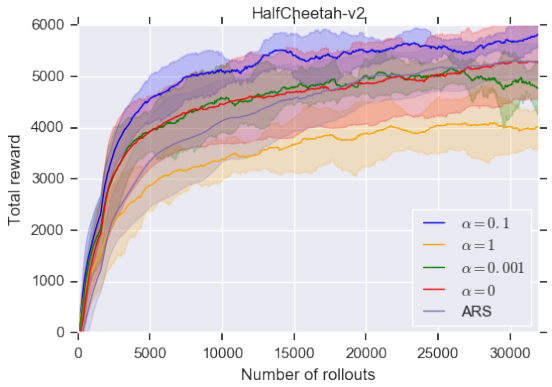
In this part, we study the sensitivity of two hyperparameters: temperature parameter α and policy variance of stochastic policies v. To show the sensitivity of the new involved hyperparameters, we evaluate the effect of the new involved hyperparameters on HalfCheetah-v2 environment with five fixed random seeds. We will explain the effect of each parameters below:

Temperature parameter α. Theoretically, the optimization objective with large α makes the learned policy more stochastic, while the small α makes the learned policy more deterministic [14]. In addition, The mixture learning objective in 8 also shows that the percentage of the entropy influence the optimization of total return. Our experiment results confirm it in Figure 3(a). We found that when $\alpha = 1$, the total return decrease and also unstable to the random seeds, while the performance of $\alpha = 0.001$ is similar to ARS. When $\alpha = 0$, the algorithm runs without the maximum entropy regularizer, and the performance is also similar to ARS. In practice a suitable α need to be tuned to obtain both performance and stability.

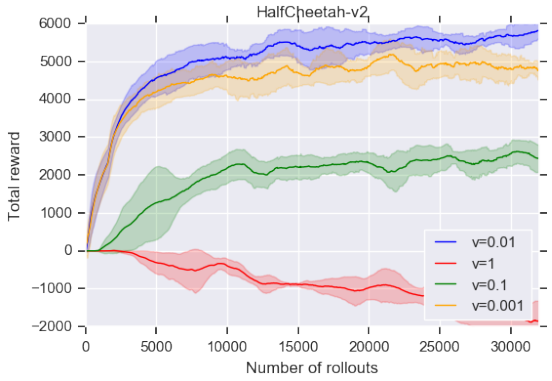
Policy variance v **of stochastic policies**. The variance of output action is directly related to the policy variance v as described in Equation 12. Therefore the policy variance parameter v controls the exploration of the action for a stochastic policy. The larger policy variance may cause the policy unstable, as the variance of output action is larger, and may need more rollouts to evaluate its return in Equation 10. As is shown in Figure 3(b), the performance is quite sensitive to the parameter v, as larger v leads the policy hard to optimize, even diverged at $v = 1$. Therefore, in practice we suggest to fixed a smaller v and tune α subsequently.

VI. Conclusion

We have proposed a maximum entropy learning framework for evolution strategies. In addition, we also derive an efficient algorithm called maximum entropy evolution policy search for continuous control. The MEPS algorithm maintains the property of ES with high efficiency, low computation cost and easy to parallelize. Our experimental results show it can achieve stable and comparable performance over evolution



(a) Temperature parameter α



(b) Policy variance v

Fig. 3. Ablation Studies on HalfCheetah-v2 environment. The upper figure shows the effect of temperature parameter α and the lower figure shows the effect of policy variance v.

strategies on the benchmark MuJoCo continuous control tasks, and is capable for high-dimensional tasks. In addition, we also provide ablation studies of the new involved hyperparameters. By selecting a suitable temperature parameter the learning of evolution strategies for continuous control can be improved. One of the drawbacks of our work is the policies used for learning are constrained to linear policies. Although our method is only capable for continuous control tasks, in the future we will study how to incorporate ES with maximum entropy reinforcement learning for discrete tasks.

ACKNOWLEDGEMENT

The authors would like to thank the anonymous reviewers for their valuable comments. This work is partly supported by National Key Research and Development Program of China (2017YFB1002503) and Natural Science Foundation of China (No. 61925603).

APPENDIX

A. Environment Details

Table I shows the state dimensions and action dimensions of the tasks used in the experiments, here is a brief description for each task:

- HalfCheetah-v2: Agent controls a cheetah-like body to run forward as quickly as possible.
- Ant-v2: Agent controls a 4-leg ant to move forward as quickly as possible.
- Hopper-v2: Agent controls a monoped to keep it from falling.
- Swimmer-v2: Agent controls a snake-like robot to swim forward as fast as possible.

We use the OpenAI Gym platform [1] [33] for implementation.

Task	State Dimensions	Action Dimensions
HalfCheetah-v2	17	6
Hopper-v2	11	3
Ant-v2	111	8
Swimmer-v2	8	2

TABLE I
THE DETAILS OF THE MUJOCO TASKS USED IN THE EXPERIMENTS.

B. Hyperparameters

The hyperparameters of MEPS used in Figure 2 are illustrated in Table II. The discount factor γ is set to 1 for all the methods. For the hyperparameters in ARS, we use the same as in [2].

The hyperparameters for vanilla ES is shown in table III .

The hyperparameters for CMA-ES[2] is shown in table IV.

For the parameters in ablation study, we keep the same parameters in II while only vary the selected parameters.

To further improve the efficiency of online learning, the normalization of states in the experiment we use is implemented in a online approach, which can be also found in [2].

REFERENCES

[1] T. Salimans, J. Ho, X. Chen, S. Sidor, and I. Sutskever, "Evolution strategies as a scalable alternative to reinforcement learning," *arXiv preprint arXiv:1703.03864*, 2017.
[2] H. Mania, A. Guy, and B. Recht, "Simple random search of static linear policies is competitive for reinforcement learning," in *Advances in Neural Information Processing Systems*, 2018, pp. 1800–1809.
[3] F. P. Such, V. Madhavan, E. Conti, J. Lehman, K. O. Stanley, and J. Clune, "Deep neuroevolution: Genetic algorithms are a competitive alternative for training deep neural networks for reinforcement learning," *arXiv preprint arXiv:1712.06567*, 2017.
[4] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.
[5] K. Choromanski, A. Pacchiano, J. Parker-Holder, Y. Tang, D. Jain, Y. Yang, A. Iscen, J. Hsu, and V. Sindhwani, "Provably robust blackbox optimization for reinforcement learning," *CoRR, abs/1903.02993*, 2019.
[6] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Machine learning*, vol. 8, no. 3-4, pp. 229–256, 1992.
[7] F. Sehnke, C. Osendorfer, T. Rückstieß, A. Graves, J. Peters, and J. Schmidhuber, "Parameter-exploring policy gradients," *Neural Networks*, vol. 23, no. 4, pp. 551–559, 2010.
[8] W. Grathwohl, D. Choi, Y. Wu, G. Roeder, and D. Duvenaud, "Backpropagation through the void: Optimizing control variates for black-box gradient estimation," *arXiv preprint arXiv:1711.00123*, 2017.
[9] Y. Nesterov and V. Spokoiny, "Random gradient-free minimization of convex functions," *Foundations of Computational Mathematics*, vol. 17, no. 2, pp. 527–566, 2017.

[1] http: gym.openai.com
[2] We use this version for implementation: https://github.com/CMA-ES/pycma

Env	HalfCheetah-v2	Swimmer-v2	Hopper-v2	Ant-v2
Learning rate (lr)	0.02	0.02	0.02	0.015
Number of individuals per iteration (N)	16	4	16	60
Number of top-performing individuals (b)	8	4	4	20
Exploration range (σ)	0.03	0.035	0.03	0.015
Temperature (α)	0.1	0.001	0.1	0.1
Number of sampling policis (M)	1	2	2	1
Policy variance (v)	0.01	0.01	0.01	0.01
Random seeds	1,2,3,4,5			

TABLE II
GRIDS OF HYPERPARAMETERS OF MAXIMUM ENTROPY EVOLUTION POLICY SEARCH USED IN FIGURE 2

Env	HalfCheetah-v2	Swimmer-v2	Hopper-v2	Ant-v2
Learning rate (lr)	0.02	0.02	0.02	0.015
Number of individuals per iteration (N)	32	8	32	120
Exploration range (σ)	0.03	0.035	0.03	0.015
Number of sampling policis (M)	1	2	2	1

TABLE III
GRIDS OF HYPERPARAMETERS OF VANILLA ES USED IN FIGURE 2

Env	HalfCheetah-v2	Swimmer-v2	Hopper-v2	Ant-v2
Learning rate	0.02	0.02	0.02	0.1
Number of individuals per iteration	40	10	10	120
Initial variance	1	0.5	0.5	0.1
Weight decay	0.01	0.01	0.01	0.01

TABLE IV
GRIDS OF HYPERPARAMETERS OF CMA-ES USED IN FIGURE 2

[10] A. R. Conn, K. Scheinberg, and L. N. Vicente, *Introduction to derivative-free optimization*. Siam, 2009, vol. 8.

[11] K. Choromanski, M. Rowland, V. Sindhwani, R. E. Turner, and A. Weller, "Structured evolution with compact architectures for scalable policy optimization," in *International Conference on Machine Learning*, 2018, pp. 969–977.

[12] S. Khadka and K. Tumer, "Evolution-guided policy gradient in reinforcement learning," in *Advances in Neural Information Processing Systems*, 2018, pp. 1188–1200.

[13] A. Pourchot and O. Sigaud, "Cem-rl: Combining evolutionary and gradient-based methods for policy search," *arXiv preprint arXiv:1810.01222*, 2018.

[14] T. Haarnoja, H. Tang, P. Abbeel, and S. Levine, "Reinforcement learning with deep energy-based policies," in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, 2017, pp. 1352–1361.

[15] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *International Conference on Machine Learning*, 2018, pp. 1856–1865.

[16] 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*. IEEE, 2012, pp. 5026–5033.

[17] Y. Tang, K. Choromanski, and A. Kucukelbir, "Variance reduction for evolution strategies via structured control variates," *arXiv preprint arXiv:1906.08868*, 2019.

[18] M. Rowland, K. M. Choromanski, F. Chalus, A. Pacchiano, T. Sarlós, R. E. Turner, and A. Weller, "Geometrically coupled monte carlo sampling," in *Advances in Neural Information Processing Systems*, 2018, pp. 195–206.

[19] J. Kober and J. R. Peters, "Policy search for motor primitives in robotics," in *Advances in neural information processing systems*, 2009, pp. 849–856.

[20] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.

[21] C. Colas, O. Sigaud, and P.-Y. Oudeyer, "Gep-pg: Decoupling exploration and exploitation in deep reinforcement learning algorithms," in *International Conference on Machine Learning*, 2018, pp. 1038–1047.

[22] S. Fujimoto, H. van Hoof, and D. Meger, "Addressing function approximation error in actor-critic methods," *arXiv preprint arXiv:1802.09477*, 2018.

[23] G. Liu, L. Zhao, F. Yang, J. Bian, T. Qin, N. Yu, and T.-Y. Liu, "Trust region evolution strategies," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, 2019, pp. 4352–4359.

[24] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, "Trust region policy optimization," in *International conference on machine learning*, 2015, pp. 1889–1897.

[25] B. D. Ziebart, "Modeling purposeful adaptive behavior with the principle of maximum causal entropy," Ph.D. dissertation, Carnegie Mellon University, 2010.

[26] B. D. Ziebart, A. Maas, J. A. Bagnell, and A. K. Dey, "Maximum entropy inverse reinforcement learning," in *Proceedings of the 23rd national conference on Artificial intelligence-Volume 3*. AAAI Press, 2008, pp. 1433–1438.

[27] E. Todorov, "General duality between optimal control and estimation," in *2008 47th IEEE Conference on Decision and Control*. IEEE, 2008, pp. 4286–4292.

[28] M. Toussaint, "Robot trajectory optimization using approximate inference," in *Proceedings of the 26th annual international conference on machine learning*. ACM, 2009, pp. 1049–1056.

[29] S. Levine and V. Koltun, "Guided policy search," in *International Conference on Machine Learning*, 2013, pp. 1–9.

[30] S. Levine, C. Finn, T. Darrell, and P. Abbeel, "End-to-end training of deep visuomotor policies," *The Journal of Machine Learning Research*, vol. 17, no. 1, pp. 1334–1373, 2016.

[31] O. Nachum, M. Norouzi, K. Xu, and D. Schuurmans, "Trust-pcl: An off-policy trust region method for continuous control," *arXiv preprint arXiv:1707.01891*, 2018.

[32] M. P. Deisenroth, G. Neumann, J. Peters *et al.*, "A survey on policy search for robotics," *Foundations and Trends® in Robotics*, vol. 2, no. 1–2, pp. 1–142, 2013.

[33] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "Openai gym," *arXiv preprint arXiv:1606.01540*, 2016.