arXiv:1901.11271v1 [cs.NE] 31 Jan 2019

Improving Evolutionary Strategies with Generative Neural Networks

Louis Faury [1 2] **Clément Calauzènes** [1] **Olivier Fercoq** [2] **Syrine Krichen** [1]

Abstract

Evolutionary Strategies (ES) are a popular family of black-box zeroth-order optimization algorithms which rely on search distributions to efficiently optimize a large variety of objective functions. This paper investigates the potential benefits of using highly flexible search distributions in classical ES algorithms, in contrast to standard ones (typically Gaussians). We model such distributions with Generative Neural Networks (GNNs) and introduce a new training algorithm that leverages their expressiveness to accelerate the ES procedure. We show that this tailored algorithm can readily incorporate existing ES algorithms, and outperforms the state-of-the-art on diverse objective functions.

1. Introduction

We are interested in the global minimization of a black-box objective function, only accessible through a *zeroth-order* oracle. In many instances of this problem the objective is expensive to evaluate, rendering brute force optimization undesirable. Moreover it is possibly non-convex and potentially highly multi-modal, hence its global optimization cannot be done greedily but requires a careful balance between exploitation and exploration of the *optimization landscape* (the surface defined by the objective).

The family of algorithms used to tackle such a problem is usually dictated by the *cost* of one evaluation of the objective function - or equivalently by the maximum number of function evaluations that are reasonable to make, and by the precision requirement. For instance, Bayesian Optimization (Jones et al., 1998; Shahriari et al., 2016) targets problems of very high evaluation cost, where the global minimum must be approximately discovered after a few hundreds of function evaluations. When aiming for a better precision and hence having a larger budget (typically several thousands of function evaluations), a popular class of algorithms is the one of Evolutionary Strategies (ES) (Rechenberg, 1978; Schwefel, 1977), a family of heuristic search procedures.

ES algorithms rely on a *search distribution*, which role is to propose queries of potentially small value of the objective function. This search distribution is almost always chosen to be a multivariate Gaussian. It is namely the case of the Covariance Matrix Adaptation Evolution Strategies (CMA-ES) (Hansen & Ostermeier, 2001), a state-of-the-art ES algorithm made popular in the machine learning community by its good results on hyper-parameter tuning (Friedrichs & Igel, 2005; Loshchilov & Hutter, 2016). It is also the case for Natural Evolution Strategies (NES) (Wierstra et al., 2008) algorithms, which were recently used for direct policy search in Reinforcement Learning (RL) and shown to compete with state-of-the-art MDP-based RL techniques (Salimans et al., 2017). Occasionally, other distributions have been used; e.g. fat-tails distributions like the Cauchy were shown to over-perform the Gaussian for highly multi-modal objectives (Schaul et al., 2011).

We argue in this paper that in ES algorithms, the choice of a given parametric distribution (Gaussian, Cauchy, ..) constitutes a *potentially harmful implicit constraint* for the stochastic search of a global minimum. To overcome the limitations of classical parametric search distributions, we propose using *flexible* distributions generated by bijective Generative Neural Networks (GNNs), with computable and differentiable log-probabilities. We discuss why common existing optimization methods used by ES algorithms cannot be directly used to train such models and design a tailored algorithm that efficiently train GNNs for an ES objective. We show how this new algorithm can readily incorporate existing ES algorithms that operates on simple search distributions, like the Gaussian. Finally, we show that this algorithm outperforms state-of-the-art ES algorithms on a variety of objective functions.

We introduce the problem and provide background on Evolutionary Strategies in Section 2. We discuss the role of GNN in generating flexible search distributions in Section 3. We explain why usual algorithms fail to train GNNs for an ES objective and introduce a new algorithm in Section 4. Finally we report experimental results in Section 5.

[1]Criteo AI Labs, 32 Rue Blanche, Paris, France [2]LTCI, Télécom ParisTech, Université Paris Saclay, France. Correspondence to: Louis Faury <l.faury@criteo.com>.

Algorithm 1 Generic ES procedure

Input: objective f, distribution π_0, population size n
repeat
(Sampling) Sample $x_1, \dots, x_n \overset{\text{i.i.d}}{\sim} \pi_t$
(Evaluation) Evaluate $f(x_1), \dots, f(x_n)$.
(Update) Update π_t to produce x of potentially smaller objective values.
until convergence

2. Preliminaries

In what follows, the objective function will be noted f and we will consider its global optimization over a compact set $\mathcal{X} \subset \mathbb{R}^d$:

$$x^* \in \underset{x \in \mathcal{X}}{\operatorname{argmin}} f(x) \tag{1}$$

The symbol π will generically denote a probability density function over $\mathcal{X}$.

2.1. Evolutionary Strategies

The generic procedure followed by ES algorithms is presented in Algorithm 1. To make the update step tractable, the search distribution is tied to a family of distributions and parametrized by a real-valued parameter vector θ (e.g. the mean and covariance matrix of a Gaussian), and would therefore now be referred to as π_θ. This update step constitute the main difference between ES algorithms.

Natural Evolution Strategies One principled way to perform that update is to minimize the expected objective value over samples x drawn from π_θ:

$$J(\theta) \triangleq \mathbb{E}_{\pi_\theta}[f(x)] \tag{2}$$

When the search distribution is parametric and tied to a parameter θ, this objective can be differentiated with respect to θ thanks to the log-trick:

$$\frac{\partial J(\theta)}{\partial \theta} = \mathbb{E}_{\pi_\theta}\left[f(x)\frac{\partial \log \pi_\theta(x)}{\partial \theta}\right] \tag{3}$$

This quantity can be approximated from samples - it is known as the score-function or REINFORCE (Williams, 1992) estimator, and provides a direction of update for θ. Unfortunately, naively following a stochastic version of the gradient (3) – a procedure called Plain Gradient Evolutionary Strategies (PGES) – is known to be highly ineffective. PGES main limitation resides in its instability when the search distribution is concentrating, making it unable to *precisely* locate any local minimum. To improve over the PGES algorithm the authors of (Wierstra et al., 2008) proposed to descend $J(\theta)$ along its *natural gradient* (Amari, 1998). More precisely, they introduce a trust-region optimization scheme to limit the instability of PGES, and minimize a linear approximation of $J(\theta)$ under a Kullback-Leibler (KL) divergence constraint:

$$\begin{aligned} \underset{\delta\theta}{\operatorname{argmin}} \quad & J(\theta+\delta\theta) \simeq J(\theta) + \delta\theta^T \nabla_\theta J(\theta) \\ \text{s.t} \quad & \mathrm{KL}(\pi_{\theta+\delta\theta} || \pi_\theta) \leq \epsilon \end{aligned} \tag{4}$$

To avoid solving analytically the trust region problem (4), (Wierstra et al., 2008) shows that its solution can be approximated by:

$$\delta\theta^* \propto -F_\theta^{-1} \nabla_\theta J(\theta) \tag{5}$$

where

$$F_\theta = \mathbb{E}_{\pi_\theta}\left[\nabla_\theta \log \pi_\theta(x) \nabla_\theta \log \pi_\theta(x)^T\right] \tag{6}$$

is the Fischer Information Matrix (FIM) of π_θ. The parameter θ is therefore not updated along the negative gradient of J but rather along $F_\theta^{-1}\nabla_\theta J(\theta)$, a quantity known as the natural gradient. The FIM F_θ is known analytically when π_θ is a multivariate Gaussian and the resulting algorithm, Efficient Natural Evolutionary Strategies (xNES) (Sun et al., 2009) has been shown to reach state-of-the-art performances on a large ES benchmark.

CMA-ES Naturally, there exist other strategies to update the search distribution π_θ. For instance, CMA-ES relies on a variety of heuristic mechanisms like covariance matrix adaptation and evolution paths, but is only defined when π_θ is a multivariate Gaussian. Explaining such mechanisms would be out of the scope of this paper, but the interested reader is referred to the work of (Hansen, 2016) for a detailed tutorial on CMA-ES.

2.2. Limitations of classical search distributions

ES implicitly balances the need for exploration and exploitation of the optimization landscape. The exploitation phase consists in updating the search distribution, and exploration happens when samples are drawn from the search distribution's tails. The key role of the search distribution is therefore to produce a support adapted to the landscape's structure, so that new points are likely to improve over previous samples.

We argue here that the choice of a given parametric distribution (the multivariate Gaussian distribution being overwhelmingly represented in state-of-the-art ES algorithms) constitutes a *potentially harmful implicit constraint* for the stochastic search of a global minimum. For instance, any symmetric distribution will be slowed down in curved valleys because of its inability to continuously curve its density. This lack of flexibility will lead it to drastically reduce its entropy, until the curved valley looks *locally* straight. At

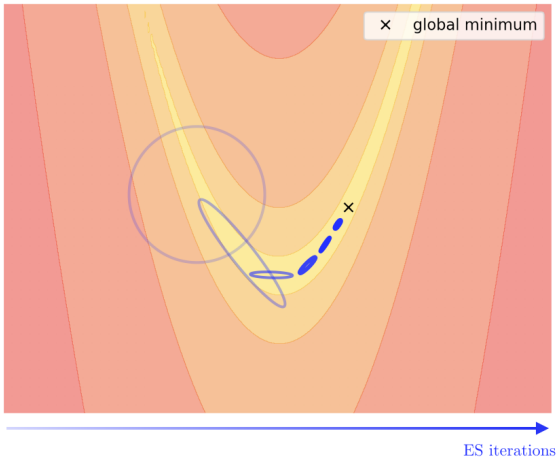


Figure 1. Example of an undesirable behavior of a Gaussian search distribution trained by xNES on the Rosenbrock function. The solid lines represent the one standard-deviation isoline of the distribution, from the beginning of the algorithm (shaded) until it reaches the straight part of the valley (full color).

this point, the ES algorithm resembles a hill-climber and barely takes advantage of the exploration abilities of the search distribution. An illustration of this phenomenon is presented in Figure 1 on the Rosenbrock function, a curved valley with high-conditioning. Another limitation of classical search distribution is their inability to follow *multiple hypothesis*, that is to explore at the same time different local minima. Even if mixture models can achieve that flexibility, hyper-parameters like the number of mixtures have optimal values that are impossible to guess *a priori*.

We want to introduce *flexible* search distributions to overcome these limitations. Such distributions should, despite their expressiveness, be easily trainable. We should also be concerned when designing them with their role in the exploration / exploitation trade off: a search distribution with too much capacity could over-fit some seemingly good samples, leading to premature convergence. To sum-up, we want to design search-distributions that are:

- more flexible than classical distributions
- yet easily trainable
- while keeping control over exploration / exploitation

In the following section, we carefully investigate the class of Generative Neural Networks (GNNs) to find a parametric class of distributions satisfying such properties.

3. Flexible search distributions with GNNs

Generative Neural Networks (MacKay, 1995) have been studied in the context of *density estimation* and shown to be able to model complex and highly multimodal distributions (Srivastava et al., 2017). We propose here to leverage their expressiveness for ES, and train them in a principled way thanks to the ES objective:

$$J(\pi) = \mathbb{E}_\pi \left[f(x) \right]$$

As discussed in Section 2, optimizing $J(\pi)$ with gradient-based methods is possible through the score-function estimator, which requires to be able to compute and efficiently differentiate the log-probabilities of π.

3.1. GNN backgound

The core idea behind a GNN is to map a *latent* variable $z \in \mathcal{Z}$ drawn from a known distribution ν_μ to an output variable $x = g_\eta(z)$ where g_η is the forward-pass of a neural network. The parameter η represents the weights of this neural network while μ describe the degrees of freedom of the latent space distribution ν_μ. We denote $\theta = (\mu, \eta)$ and $\pi_\theta(x)$ the density of the resulting output variable x.

For general neural network architectures, it is impossible to compute the likelihood $\pi_\theta(x)$ of samples x drawn from the GNN. This is namely why their are often trained with adversarial methods (Goodfellow et al., 2014) for sample generation purposes, bypassing the need of computing densities, but at the expense of a good density estimation (mode-dropping). An alternative to adversarial methods was proposed with Variational Auto-Encoders (Kingma & Welling, 2013) however at the cost of learning two neural networks (an encoder and a decoder). A less computationally expensive method consists in restricting the possible architectures to build *bijective* GNNs, which allows the exact computation of the distribution's density.

Indeed, if g_η is a bijection from $\mathcal{Z}$ to $\mathcal{X}$ with inverse $h_\eta \triangleq g_\eta^{-1}$, the change of variable formula provides a way to compute $\pi_\theta(x)$:

$$\pi_\theta(x) = \nu_\mu(h_\eta(x)) \cdot \left| \frac{\partial h_\eta(x)}{\partial x} \right| \tag{7}$$

To have a tractable density one therefore needs to ensure that the determinant of the Jacobian $|\partial h_\eta(x)/\partial x|$ is easily computable. Several models satisfying these two properties (*i.e* bijectivity and computable Jacobian) have been proposed for density estimation (Rippel & Adams, 2013; Dinh et al., 2014; 2016), and proved their high expressiveness despite their relatively simple structure.

Bijective GNNs therefore answer two of our needs when building our new search distribution: flexibility and easiness to train. In this work, we will focus on one bijective GNN, the Non-Linear Independent Component Estimation (Dinh et al., 2014) (NICE) model, for its numerical stability and *volume preserving* properties.

3.2. NICE model

The authors of NICE proposed to build complex yet invertible transformations through the use of *additive coupling layers*. An additive coupling layer leaves half of its input unchanged, and adds a non-linear transformation of the first half to the second half. More formally, by noting $v = [v_1, v_2]$ the output of a coupling layer and $u = [u_1, u_2]$ its input, one has:

$$\begin{aligned} v_1 &= u_1 \\ v_2 &= u_2 + t(u_1) \end{aligned} \tag{8}$$

where t is an arbitrarily complex transformation - modelled by a Multi-Layer Perceptron (MLP) with learnable weights and biases. This transformation has unit Jacobian and is easily invertible:

$$\begin{aligned} u_1 &= v_1 \\ u_2 &= v_2 - t(v_1) \end{aligned} \tag{9}$$

and only requires a feed-forward pass on the MLP t. The choice of the decomposition $u = [u_1, u_2]$ can be arbitrary, and is performed by applying a binary filter to the input. By stacking additive coupling layers, one can create complex and possibly multimodal distributions. The inversion of the resulting network is independent of the complexity of t. Also, as noted by the authors of (Dinh et al., 2014), it is enough to stack only three coupling layers with complementary binary masks in order to make sure that every dimension influences all the others. The density of the resulting distribution is then readily computable thanks to the inverse transform theorem (7).

3.3. Volume preserving properties

The transformation induced by NICE is *volume preserving* (it has a unitary Jacobian determinant). This is quite desirable in a ES context, as the role of concentrating the distribution on a minimum can be left to the latent space distribution ν_μ. The role of the additive coupling layers is therefore only to introduce non-linearities in the inverse transform h_η so that the distribution is better adapted to the optimization landscape. The fact that this fit can be done while preserving the global volume of the distribution forces the model to align its tails with the optimization landscape, which is likely to improve the quality of future exploration steps. The NICE model therefore fits perfectly our needs for a flexible search distribution that is easy to train, and that provides enough control on the exploration / exploitation trade-off. Indeed, other bijective GNN models like the Real-NVP (Dinh et al., 2016) introduce non-volume preserving transformations, which can easily overfit on good samples and lead to premature convergence.

4. An efficient training algorithm

We are now equipped with enough tools to use GNNs for ES: an adapted model (NICE) for our search distribution π_θ, and an objective to train it with:

$$J(\theta) = \mathbb{E}_{\pi_\theta}\left[f(x)\right] \tag{10}$$

Here, θ describes *jointly* the free parameters of the latent distribution ν_μ and η, the weights and biases of the MLPs forming the additive coupling layers.

We start this section by explaining why existing training strategies based on the objective (10) are not sufficient to truly leverage the flexibility of GNNs for ES, before introducing a new algorithm tailored for this task.

4.1. Limitations of existing training strategies

We found that the PGES algorithm (naive stochastic gradient descent of (10) with the score-function estimator (3)) applied to the NICE distribution suffers from the same limitations as when applied to the Gaussian - that is inability to precisely locate any local minimum. As for the Gaussian, training the NICE distribution for ES requires employing more sophisticated algorithms - such as NES.

Using the natural gradient for the GNNs distributions is not trivial. First the Fischer Information Matrix F_θ is not known analytically and must be estimated via Monte-Carlo sampling, thereby introducing approximation errors. Also, we found that the approximations justifying to follow the descent direction provided by the natural gradient are not adapted to the NICE distribution. Indeed, the assumption behind the NES update (5) is that the loss $J(\theta)$ can be (locally) well approximated by the quadratic objective:

$$J(\theta + \delta\theta) = J(\theta) + \delta\theta^T \nabla_\theta J(\theta) + \frac{\lambda}{2}\delta\theta^T F_\theta \delta\theta \tag{11}$$

where λ is a given non-negative Lagrange multiplier. For NICE, given the highly non-linear nature of π_θ this approximation is bound to fail even close to the current parameter θ and will lead to spurious updates. A classical technique (Nocedal & Wright, 2006; Martens, 2010) to avoid such updates is to artificially increase the curvature of the quadratic term, and is known as *damping*. Practically, this implies using $F_\theta + \beta I$ instead of F_θ as the local curvature metric, with β a non-negative damping parameter.

We found in our experiments that to ensure continuous decrease of $J(\theta)$, and because of its highly non-linear nature when using the GNNs, the damping parameter β has to be set to such high values that the modifications in the search distributions are too small to quickly make progress and by no means reaches state-of-the-art performances. We observed that even if the training of the additive coupling layers is performed correctly (i.e the distribution has the cor-

rect *shape*), it is high damping of the latent space parameters that prevents the distribution to quickly concentrate when a minimum is found.

It is unclear how the damping parameter should be adapted to avoid spurious update, while still allowing the distribution to make large step in the latent space and ensure fast concentration when needed. Hence, in the following, we rather present an *alternated minimization* scheme to bypass the issues raised by natural gradient training for GNN distributions in a ES context.

4.2. Alternated minimization

Latent space parameters optimization So far, we used the parameter θ to describe both the free parameters μ of the latent space distribution ν_μ and the ones of the non-linear transformations of the additive coupling layers η, and the optimization of all these parameters was done jointly. Separating the roles of μ and η, the initial objective (2) can be rewritten as follows:

$$J(\theta) = J(\mu, \eta) \tag{12}$$
$$= \mathbb{E}_{z\sim\nu_\mu}\left[f(g_\eta(z))\right] \tag{13}$$

This rewriting of the initial objective can give us a new view of the role of GNNs in ES algorithms. Namely, that this role can be to learn an efficient transformation g_η so that the representation of $f \circ g_\eta$ in the latent space is *easy* to minimize for the latent distribution. If ν_μ is a standard distribution (i.e efficiently trainable with the natural gradient) and that $f \circ g_\eta$ is a *well structured* function (i.e one for which ν_μ is an efficient search distribution), then the single optimization of μ by classical methods should avoid the limitations discussed earlier. This new representation motivates us to design a new training algorithm that now optimizes the parameters μ and η *separately*.

Alternated Minimization In the following, we will replace the notation π_θ with $\pi_{\mu,\eta}$ to refer to the NICE distribution with parameter $\theta = (\mu, \eta)$. We want to optimize μ and η in an alternate fashion, which means performing the following updates at every step of the ES procedure presented in Algorithm 1:

$$\mu_{t+1} = \operatorname*{argmin}_\mu J(\mu, \eta_t) \tag{14a}$$
$$\eta_{t+1} = \operatorname*{argmin}_\eta J(\mu_{t+1}, \eta) \tag{14b}$$

This means that at iteration t, samples will be drawn from π_{μ_t,η_t} and will serve to first optimize the latent space distribution parameters μ, and then the additive coupling layers parameters η. The next population will therefore be sampled with $\pi_{\mu_{t+1},\eta_{t+1}}$.

The update (14a) of the latent space parameters is naturally derived from the new representation (13) of the initial objective. Indeed, μ can be updated via natural gradient ascent of $J(\mu, \eta_t)$ - that is with keeping $\eta = \eta_t$ fixed. Practically, this therefore reduces to applying a NES algorithm to the latent distribution ν_μ on the modified objective function $f \circ g_{\eta_t}$.

Once the latent space parameters updated, the coupling layers parameters should be optimized with respect to:

$$J(\mu_{t+1}, \eta) = \mathbb{E}_{\pi_{\mu_{t+1},\eta}}\left[f(x)\right] \tag{15}$$

Because the available samples are drawn from π_{μ_t,η_t}, the objective (15) must be unbiased with *importance sampling* if we wish to approximate it from samples:

$$J(\mu_{t+1}, \eta) = \mathbb{E}_{\pi_{\mu_t,\eta_t}}\left[f(x)\frac{\pi_{\mu_{t+1},\eta}(x)}{\pi_{\mu_t,\eta_t}(x)}\right] \tag{16}$$

The straightforward minimization of the *off-line* objective (16) is known to lead to degeneracies (Swaminathan & Joachims, 2015), and must therefore be regularized. For our application, it is also desirable to make sure that the update η does not undo the progress made in the latent space - or in other words, we want to regularize the change in $f \circ g_\eta$. To that extent, we add a trust-region to the minimization of (16) to ensure that the KL divergence between $\pi_{\theta_{t+1}}$ and π_{μ_{t+1},η_t} remains bounded by a small quantity ε:

$$\begin{aligned}\eta_{t+1} = \operatorname*{argmin}_\eta \quad & \mathbb{E}_{\pi_{\theta_t}}\left[f(x)\frac{\pi_{\mu_{t+1},\eta}(x)}{\pi_{\mu_t,\eta_t}(x)}\right] \\ \text{s.t} \quad & \mathrm{KL}\left(\pi_{\mu_{t+1},\eta_t}||\pi_{\mu_{t+1},\eta}\right) \leq \varepsilon\end{aligned} \tag{17}$$

This problem can be efficiently approximated by a simple penalization scheme:

$$\begin{aligned}&\mathbb{E}_{\pi_{\theta_t}}\left[f(x)\frac{\pi_{\mu_{t+1},\eta}(x)}{\pi_{\mu_t,\eta_t}(x)}\right] \\ &+ \lambda\mathrm{KL}\left(\pi_{\mu_{t+1},\eta_t}||\pi_{\mu_{t+1},\eta}\right)\end{aligned} \tag{18}$$

where λ is a non-negative penalization coefficient. Its value can be adapted through a simple scheme presented in (Schulman et al., 2017) to ensure that the constraint arising in (17) is satisfied. The KL divergence can readily be approximated by Monte-Carlo sampling of π_{μ_{t+1},η_t} (which only requires sampling in the distribution) and a minimum of (18) found by a gradient-based algorithm. We found the initial value of λ has little practical impact, as it is quickly adapted through the optimization procedure.

To sum up, we propose optimizing the latent distribution and the coupling layers separately. The latent space is optimized by natural gradient descent, and the coupling layers via an off-policy objective with a Kullback-Leibler divergence penalty. We call this algorithm GNN-ES for Generative Neural Networks Evolutionary Strategies.

Latent space optimization It turns out the GNN-ES can be readily modified to incorporate virtually *any* existing ES algorithms that operates on the simple distribution ν_μ. For instance, if ν_μ is set to be a multivariate Gaussian with learnable mean and covariance matrix, the latent space optimization (14a) can be performed by either xNES or CMA-ES. This holds for any standard distribution ν_μ and any ES algorithm operating on that distribution.

This remark allows us to place GNN-ES in a more general framework and to understand it as a way to improve currently existing ES algorithm, by providing a principled way to learn complex, non-linear transformations on top of rather standard search distributions (like the Gaussian). In what follows, we will use the GNN prefix in front of existing ES algorithm to describe their augmented version with our algorithm.

This new way of understanding GNN-ES highlights the need for a volume-preserving transformation, like the one provided by NICE. Indeed, the role of non-linearities is now only to provide a better representation of the objective function in the latent space *without* changing the volume, leaving the exploration / exploitation balance to the ES algorithm dedicated to training the latent distribution.

Algorithm overview We provide in Algorithm 2 the pseudo-code for the generic algorithm GNN-$\mathcal{A}$-ES. Its principal inputs are the nature of the latent distribution ν_μ and the ES algorithm $\mathcal{A}$ used to optimize it.

Using historic data Because the objective (16) uses importance sampling and therefore is off-line by nature, any historic data can be used to estimate its left-hand side. Therefore, all previously sampled points can be stored in a buffer and used to optimize the coupling layers parameters. On the optimization landscapes we considered, we didn't find this technique to bring a significant uplift to GNN-ES performances. In-depth exploration of this augmentation, popular in similar settings like MDP-based Reinforcement Learning and counterfactual reasoning (Nedelec et al., 2017; Agarwal et al., 2017), was therefore left to future work.

Algorithm 2 GNN-$\mathcal{A}$-ES (ex: GNN-xNES, GNN-CMA-ES)

Input: objective f, distribution ν and its related ES algorithm $\mathcal{A}$, initial parameter $\theta_0 = (\mu_0, \eta_0)$, initial λ_0, radius ε, population size N, KL sample size M.
repeat
Sampling
Sample $Z = z_1, \dots z_N \overset{\text{i.i.d}}{\sim} \nu_{\mu_t}$.
Feed-forward on g_{η_t} to obtain $x_1, \dots x_N \overset{\text{i.i.d}}{\sim} \pi_{\theta_t}$.
Evaluation Evaluate $F = f(x_1), \dots, f(x_N)$.
Latent space optimization

$$\mu_{t+1} \leftarrow \mathcal{A}\left(\mu_t, (Z, F)\right)$$

▷ apply $\mathcal{A}$ to the latent distribution
Bijective network optimization
Sample $\tilde{x}_1, \dots, \tilde{x}_M \overset{\text{i.i.d}}{\sim} \pi_{\mu_{t+1}, \eta_t}$.
Compute

$$\tilde{KL}(\mu_t, \eta) \leftarrow \frac{1}{M} \sum_{i=1}^{M} \log \left(\frac{\pi_{\mu_{t+1}, \eta_t}(\tilde{x}_i)}{\pi_{\mu_{t+1}, \eta}(\tilde{x}_i)} \right)$$

Minimize w.r.t η:

$$\frac{1}{N} \sum_{i=1}^{N} f(x_i) \frac{\pi_{\mu_{t+1}, \eta}(x_i)}{\pi_{\mu_{t+1}, \eta_t}(x_i)} + \lambda_t \tilde{KL}(\mu_t, \eta)$$

to obtain η_{t+1}.
Adapting λ
if $\tilde{KL}(\mu_t, \eta_{t+1}) > 2\varepsilon$ **then**
$\lambda_{t+1} \leftarrow 1.5\lambda_t$
else if $\tilde{KL}(\mu_t, \eta_{t+1}) < 0.5\varepsilon$ **then**
$\lambda_{t+1} \leftarrow \lambda_t / 1.5$
end if
until convergence

5. Experimental results

5.1. Visualization

We present here two-dimensional visualizations of the behavior of a GNN distribution trained with GNN-xNES versus a Gaussian distribution trained by xNES.

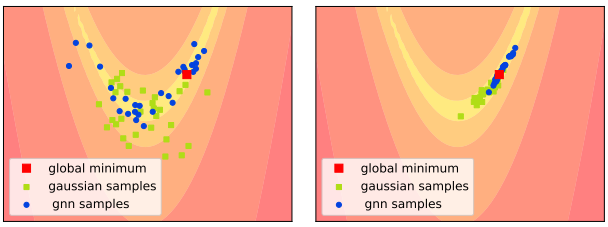
Figure 2 shows two iterations of the different algorithms on the Rosenbrock function, that typically highlights the Gaussian distribution lack of flexibility. On this function, GNN-xNES efficiently detects the curved valley and reaches the global minimum faster than xNES, while keeping a higher entropy.

Figure 3 provide a similar visualization on the Rastrigin function, a highly multimodal but symmetric objective. While both distributions discover the global minimum, GNN-xNES naturally creates a multi-model distribution, simultaneously exploring several local minima.

5.2. Synthetic objectives

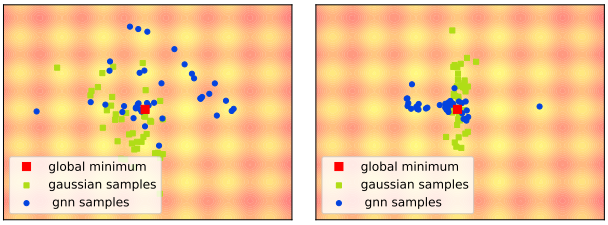
Experimental set-up We present experiments on both unimodal and multimodal optimization landscapes. We use official implementations[1] of xNES and CMA-ES as base-

[1]Implementation of xNES and CMA-ES were taken respectively from the PyBrain (Schaul et al.,



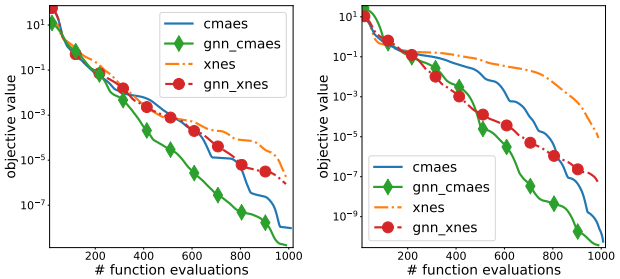
(a) Iteration 5 (b) Iteration 15

Figure 2. Evolution of distribution trained by GNN-xNES and xNES on the Rosenbrock function.

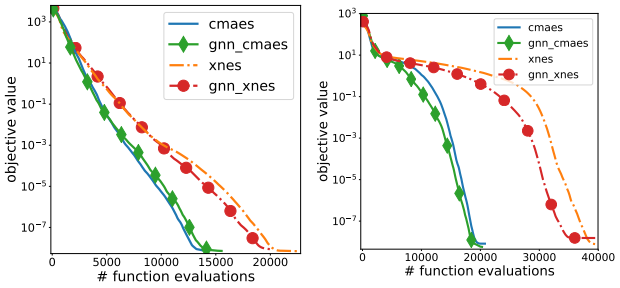


(a) Iteration 10 (b) Iteration 20

Figure 3. Evolution of distribution trained by GNN-xNES and xNES on the Rastrigin function.



(a) Bent Cigar, d=2 (b) Rosenbrock, d=2



(c) Bent Cigar, d=10 (d) Rosenbrock, d=10

Figure 4. Unimodal experiments

lines and as inner optimization methods of GNN-xNES and GNN-CMA-ES. Both xNES and CMA-ES are used with their default (adaptive) hyper-parameters.

For all synthetic objectives, the optimization landscape is randomly translated in the compact $[-2, 2]^d$ at the beginning of every ES trajectory to evaluate multiple configurations with different global minimum positions. All algorithms are benchmarked on the same translated version of the objective, with similar initialization of their respective search distributions.

We build the NICE model with three coupling layers. Each coupling layer's non-linear mapping t is built with a one hidden layer MLP, with 16 neurons and hyperbolic tangent activation. This architecture is kept constant in all our experiments. Other parameters, like the Kullback-Leibler radius ε is also kept constant in all experiments, with value $\varepsilon = 0.01$.

Unimodal landscapes We run the different algorithms on asymmetric landscapes, where we expect GNN search distributions to bring a significant improvement compared to the Gaussian - as discussed in 2.2. These objectives functions are the Rosenbrock function (a curved valley with high conditioning) and the Bent Cigar function (a slightly better conditioned curved valley). Those two landscapes are often used in ES benchmarks (Hansen et al., 2010) to evaluate a search distribution abilities to continuously adapt its search direction.

We report performance measured in terms of the best current found value of the objective as a function of the number of function evaluations. Results are averaged over 10 random seeds. The population size is set to $N = 10d$ for all algorithms.

Results for $d = 2$ are presented in Figure 4a and 4b. Systematically, the GNN models (GNN-CMA-ES and GNN-xNES) significantly outperform their respective baselines CMA-ES and xNES by achieving a faster convergence to the minimum. The mean improvement is quite significant, as at a given number of function evaluations the performance uplift can be of several orders of magnitude.

Results in higher dimensions ($d = 10$) are presented in Figure 4c and 4d. On the Bent Cigar function, we observe that GNN-CMA-ES does not improve over CMA-ES. We attribute this behavior to faster concentration of CMA-ES near the global optimum (the performances slightly diverge only close to the minimum).On the other hand, GNN-xNES strictly improved over xNES on this landscape. On the Rosenbrock function, both the GNN-based algorithm improve over their respective baselines. On this landscape, the speed-up is quite significant, as it reaches orders of magnitude at the beginning of the ES procedure. For instance, at 10^4 function evaluations, GNN-CMA-ES discovers objective values approximately 10 time better than its counterpart CMA-ES.

2010) library, and the PyCMA package available at `https://github.com/CMA-ES/pycma`.

Table 1. Multimodal results

Objective	Average Minimum Value			
	CMA-ES	GNN-CMA-ES	XNES	GNN-XNES
Styblinski d=2	11.3	**5.65**	9.89	9.89
Styblinski d=4	25.44	**14.1**	29.68	18.37
Rastrigin d=2	1.03	1.14	**0.29**	0.49
Rastrigin d=4	1.60	2.82	**0.65**	3.56
Griewank d=2	0.005	**0.003**	0.005	0.025
Griewank d=4	0.002	**0.001**	0.003	0.003
Beale, d=2	0.14	**0.05**	0.09	0.09
Beale, d=4	0.10	**0.06**	0.34	0.09

Multimodal landscapes We report the performances of the different algorithms on multimodal objectives in Table 1. We measure performance as the value of the minimum (possibly local) of the objective function discovered by the ES algorithms at convergence, averaged over 20 random seeds. We consider four multimodal landscapes with different structures: the Rastrigin, the Styblinski, the Griewank and the Beale functions. Their expressions are reported in Appendix A.

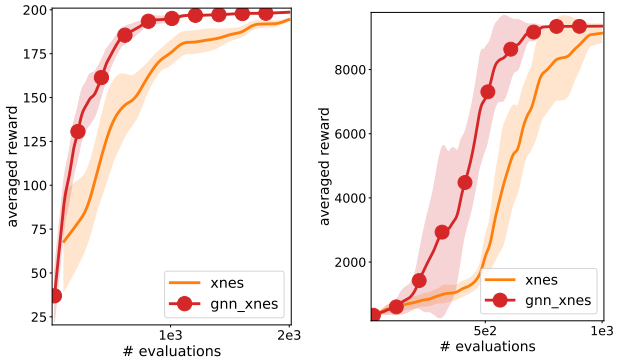
GNN-CMA-ES constantly discovers better global minimum than other algorithms, except for one test function, the Rastrigin. This objective is highly multimodal and with poor global structure (i.e relatively *flat*). Apart from leveraging its symmetry, the Gaussian search distribution has less capacity and is less likely to prematurely converge to a local minima in such a highly multimodal landscape - which explains its better results in this case. The benefits of using GNN are more highlighted in the other functions which have less local minima and better global structure, which allows the GNN search distribution to efficiently explore several of them at the same time and to discover more often the global minimum.

5.3. Reinforcement Learning experiments

ES algorithms have recently been used for direct policy search in Reinforcement Learning (RL) and shown to reach performances comparable with state-of-the-art MDP-based techniques (Liu et al., 2019; Salimans et al., 2017). Direct Policy Search forgets the MDP structure of the RL environment and rather considers it as a *black-box*. The search for the optimal policy is performed directly in parameter space to maximize the average reward per trajectory:

$$f(x) = \mathbb{E}_{\tau \sim p_x} \left[\sum_{j \in \tau} r_j \right] \tag{19}$$

where p_x is the distribution of trajectories induced by the policy (the state-conditional distribution over actions) parametrized by x, and r the rewards generated by the environment.



(a) Swimmer-v1 (b) InvertedDoublePendulum

Figure 5. Direct Policy Search experiments

The objective (19) can readily be approximated from samples by simply rolling out M trajectories, and optimized using ES. In our experiments, we set $M = 10$ and optimize deterministic linear policies[2].

In Figures 5a and 5b we report results of the GNN-xNES algorithm compared to xNES, when run on the Mujoco locomotion tasks Swimmer And InvertedDoublePendulum, both from the OpenAI Gym (Brockman et al., 2016). Performance is measured by the average reward per trajectory and per ES populations as a function of the number of evaluations of the objective f. Results are averaged over 5 random seeds (ruling the initialization of the environment and the initial distribution over the policy parameters x). In both environments, GNN-xNES discovers behaviors of high rewards faster than xNES.

6. Conclusion

In this work, we motivate the use of GNNs for improving Evolutionary Strategies by pinpointing the limitations of classical search distributions, commonly used by standard ES algorithms. We propose a new algorithm that leverages the high flexibility of distributions generated by bijective GNNs with an ES objective. We highlight that this algorithm can be seen as a plug-in extension to existing ES algorithms, and therefore can virtually incorporate *any* of them. Finally, we show its empirical advantages across a diversity of synthetic objective functions, as well as from objectives coming from Reinforcement Learning.

As possible extensions to our method, we could investigate *noisy* zeroth-order oracles as well as first-order oracles, a problem already explored in (Grathwohl et al., 2017; Faury et al., 2018).

[2]We used the `rllab` library (Duan et al., 2016) for experiments.

References

Agarwal, A., Basu, S., Schnabel, T., and Joachims, T. Effective evaluation using logged bandit feedback from multiple loggers. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 687–696. ACM, 2017.

Amari, S.-I. Natural gradient works efficiently in learning. *Neural Computation*, 10(2):251–276, 1998.

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., and Zaremba, W. OpenAi Gym. *arXiv preprint arXiv:1606.01540*, 2016.

Dinh, L., Krueger, D., and Bengio, Y. NICE: Non-Linear Independent Components Estimation. *arXiv preprint arXiv:1410.8516*, 2014.

Dinh, L., Sohl-Dickstein, J., and Bengio, S. Density Estimation using Real NVP. *arXiv preprint arXiv:1605.08803*, 2016.

Duan, Y., Chen, X., Houthooft, R., Schulman, J., and Abbeel, P. Benchmarking Deep Reinforcement Learning for Continuous Control. In *International Conference on Machine Learning*, pp. 1329–1338, 2016.

Faury, L., Vasile, F., Calauzènes, C., and Fercoq, O. Neural generative models for global optimization with gradients. *arXiv preprint arXiv:1805.08594*, 2018.

Friedrichs, F. and Igel, C. Evolutionary tuning of multiple svm parameters. *Neurocomputing*, 64:107–117, 2005.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. Generative Adversarial Nets. In *Advances in Neural Information Processing Systems*, pp. 2672–2680, 2014.

Grathwohl, W., Choi, D., Wu, Y., Roeder, G., and Duvenaud, D. Backpropagation through the void: Optimizing control variates for black-box gradient estimation. *arXiv preprint arXiv:1711.00123*, 2017.

Hansen, N. The CMA Evolution Strategy: a tutorial. *arXiv preprint arXiv:1604.00772*, 2016.

Hansen, N. and Ostermeier, A. Completely derandomized self-adaptation in Evolution Strategies. *Evolutionary Computation*, 9(2):159–195, 2001.

Hansen, N., Auger, A., Finck, S., and Ros, R. *Real-parameter black-box optimization benchmarking 2010: Experimental setup*. PhD thesis, INRIA, 2010.

Jones, D. R., Schonlau, M., and Welch, W. J. Efficient global optimization of expensive black-box functions. *Journal of Global optimization*, 13(4):455–492, 1998.

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

Liu, G., Zhao, L., Yang, F., Bian, J., Qin, T., Yu, N., and Liu, T.-Y. Trust Region Evolution Strategies. 2019.

Loshchilov, I. and Hutter, F. CMA-ES for hyperparameter optimization of deep neural networks. *arXiv preprint arXiv:1604.07269*, 2016.

MacKay, D. J. Bayesian neural networks and density networks. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 354(1):73–80, 1995.

Martens, J. Deep Learning via Hessian-free Optimization. In *International Conference of Machine Learning*, volume 27, pp. 735–742, 2010.

Nedelec, T., Roux, N. L., and Perchet, V. A comparative study of counterfactual estimators. *arXiv preprint arXiv:1704.00773*, 2017.

Nocedal, J. and Wright, S. J. *Numerical Optimization*. Springer, New York, NY, USA, second edition, 2006.

Rechenberg, I. Evolutionsstrategien. In *Simulationsmethoden in der Medizin und Biologie*, pp. 83–114. Springer, 1978.

Rippel, O. and Adams, R. P. High-dimensional Probability Estimation with Deep Density Models. *arXiv preprint arXiv:1302.5125*, 2013.

Salimans, T., Ho, J., Chen, X., Sidor, S., and Sutskever, I. Evolution Strategies as a scalable alternative to Reinforcement Learning. *arXiv preprint arXiv:1703.03864*, 2017.

Schaul, T., Bayer, J., Wierstra, D., Sun, Y., Felder, M., Sehnke, F., Rückstieß, T., and Schmidhuber, J. PyBrain. *Journal of Machine Learning Research*, 11:743–746, 2010.

Schaul, T., Glasmachers, T., and Schmidhuber, J. High dimensions and heavy tails for Natural Evolution Strategies. In *Proceedings of the 13th annual conference on Genetic and Evolutionary Computation*, pp. 845–852. ACM, 2011.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Schwefel, H.-P. *Numerische Optimierung von Computer-Modellen mittels der Evolutionsstrategie: mit einer vergleichenden Einführung in die Hill-Climbing-und Zufallsstrategie*. Birkhäuser, 1977.

Shahriari, B., Swersky, K., Wang, Z., Adams, R. P., and De Freitas, N. Taking the human out of the loop: A

review of Bayesian Optimization. *Proceedings of the IEEE*, 104(1):148–175, 2016.

Srivastava, A., Valkoz, L., Russell, C., Gutmann, M. U., and Sutton, C. Veegan: Reducing mode collapse in GANs using Implicit Variational Learning. In *Advances in Neural Information Processing Systems*, pp. 3308–3318, 2017.

Sun, Y., Wierstra, D., Schaul, T., and Schmidhuber, J. Efficient Natural Evolution Strategies. In *Proceedings of the 11th Annual Conference on Genetic and Evolutionary Computation*, pp. 539–546. ACM, 2009.

Swaminathan, A. and Joachims, T. Counterfactual Risk minimization: Learning from Logged Bandit Feedback. In *International Conference on Machine Learning*, pp. 814–823, 2015.

Wierstra, D., Schaul, T., Peters, J., and Schmidhuber, J. Natural Evolution Strategies. In *Evolutionary omputation, 2008. CEC 2008.(IEEE World Congress on Computational Intelligence)*, pp. 3381–3387. IEEE, 2008.

Williams, R. J. Simple statistical gradient-following algorithms for connectionist Reinforcement Learning. *Machine Learning*, 8(3-4):229–256, 1992.

A. Experimental details

A.1. Synthetic objectives

We provide in Table 2 the expressions of the synthetic objective functions used in this paper. We use the asymmetric operator T^{β}_{asy} for the Bent Cigar function introduced in (Hansen et al., 2010). R denotes a (random) rotation matrix in $\mathbb{R}^d$. We use $\beta = 0.5$ for $d = 2$ and $\beta = 2$ for $d = 10$.

Table 2. Synthetic objectives

Objective function	Expression
Rosenbrock	$f_1(x) = \sum_i (1 - x_i)^2 + 100(x_{i+1} - x_i)$
Cigar	$f_2(x) = x_1^2 + 10^4 \sum_{i=2}^{d} x_i^2$
Bent Cigar	$f_3(x) = f_2(RT^{\beta}_{\text{asy}}(x)R)$
Rastrigin	$f_4(x) = 10d + \sum_{i=1}^{d} \left(x_i^2 - A\cos(2\pi x_i)\right)$
Griewank	$f_5(x) = \sum_{i=1}^{d} \frac{x_i^2}{4000} - \prod_{i=1}^{2} \cos\left(\frac{\sqrt{x_i}}{\sqrt{i}}\right) + 1$
Beale	$f_6(x) = (1.5 - x_1 + x_1x_2)^2 + \left(2.25 - x_1 + x_1x_2^2\right)^2 + \left(2.625 - x_1 + x_1x_2^3\right)^2 + \sum_{i=3}^{d} x_i^2$
Styblinski	$f_7(x) = \frac{1}{2}\sum_{i=1}^{d} \left(x_i^4 - 16x_i^2 + 5x_i\right)$

A.2. RL environments

Table 3 provides details on the RL environment used to compare GNN-xNES and xNES, like the dimensions of the state space $\mathcal{S}$ and action space $\mathcal{A}$, and the maximum number of steps per trajectory.

Table 3. RL environment details

Name	$\lvert\mathcal{S}\rvert$	$\lvert\mathcal{A}\rvert$	Steps
Swimmer-v1	13	2	1000
InvertedDoublePendulum-v1	11	1	1000