# CoNES: Convex Natural Evolutionary Strategies

**Anonymous authors**
Paper under double-blind review

## Abstract

We present a novel algorithm – *convex natural evolutionary strategies (CoNES)* – for optimizing high-dimensional blackbox functions by leveraging tools from convex optimization and information geometry. CoNES is formulated as an efficiently-solvable convex program that adapts the evolutionary strategies (ES) gradient estimate to promote rapid convergence. The resulting algorithm is *invariant* to the parameterization of the belief distribution. Our numerical results demonstrate that CoNES vastly outperforms conventional blackbox optimization methods on a suite of functions used for benchmarking blackbox optimizers. Furthermore, CoNES demonstrates the ability to converge faster than conventional blackbox methods on a selection of OpenAI's MuJoCo reinforcement learning tasks for locomotion.

## 1 Introduction

Policy optimization in reinforcement learning (RL) can be posed as a blackbox optimization problem: given access to a "blackbox" in the form of a simulator or robot hardware, find a setting of policy parameters that maximizes rewards. This perspective has led to significant recent interest from the RL community towards scaling blackbox optimization methods and has catapulted the use of blackbox optimizers from low-dimensional hyperparameter tuning (Golovin et al., 2017; Hutter et al., 2019) to training deep neural networks (DNNs) with thousands of parameters (Salimans et al., 2017; Choromanski et al., 2019a;b; Liu et al., 2019; Conti et al., 2018; Mania et al., 2018). Despite these promising advances, the sample complexity of blackbox methods remains high and is the subject of ongoing research.

In this paper we study a class of blackbox optimization methods called evolutionary strategies (ES) (Rechenberg & Eigen, 1973; Salimans et al., 2017). ES methods maintain a belief distribution on the domain of candidates. At each iteration, a batch of candidates is sampled from this distribution and their fitness is evaluated. These fitness scores are used to obtain a Monte-Carlo (MC) estimate of the loss function's gradient with respect to the parameters of the belief distribution. In the domain of ES for RL, approaches that adapt the sampling rate from the belief distribution and reuse samples from previous iterations have been proposed to improve the sample complexity (Choromanski et al., 2019a;b). However, standard ES methods are not invariant to re-parameterizations of the belief distribution. Hence, the choice of belief parameterization (e.g., encoding the covariance as a symmetric positive definite matrix vs. a Cholesky decomposition) can affect the rate of convergence and cause undesirable behavior (e.g., oscillations) (Wierstra et al., 2014). In contrast, ES techniques based on the *natural gradient* (Amari, 1998; Sun et al., 2009; Wierstra et al., 2014) are *parameterization invariant* and can demonstrate improved sample efficiency. However, these methods have not been thoroughly exploited in RL due to the difficulties in computing the natural gradient for high-dimensional problems; in particular, the challenging estimation of the Fisher information matrix is necessary for computing the natural gradient.

In this paper, we present a novel algorithm – *convex natural evolutionary strategies (CoNES)* – that leverages results on the natural gradient (Amari, 1998; Sun et al., 2009; Wierstra et al., 2014) from information geometry (Amari, 2016) and couples them with powerful tools from convex optimization (e.g., second-order cone programming (Boyd & Vandenberghe, 2004) and geometric programming (Boyd et al., 2007)) to promote rapid convergence. In particular, CoNES refines a crude gradient estimate by transforming it through a convex program that searches for the direction of steepest ascent in a KL-divergence ball around the current belief distribution. The relationship to

natural evolutionary strategies (NES) (Wierstra et al., 2014) comes from the fact that the limiting solution of the KL-constrained optimization problem (as the "radius" of the KL-divergence ball shrinks to zero) corresponds to the natural gradient. However, in contrast to NES (Wierstra et al., 2014), CoNES circumvents the estimation of the Fisher information matrix by directly solving the convex KL-constrained optimization problem. Furthermore, tuning the radius of the KL-divergence ball facilitates better alignment of the update direction with the update step size, yielding faster convergence than NES (which provides the steepest ascent direction for infinitesimal steps lengths).

Our theoretical results establish that CoNES is *invariant* to the parameterization of the belief distribution (e.g., encoding the covariance as a symmetric positive definite matrix or a Cholesky decomposition does not affect the solution of the CoNES optimization problem). Parameterization invariance ensures that we are working with the intrinsic mathematical object (i.e., probability distribution) and the specific encoding of these objects do not affect the outcome. Moreover, CoNES is agnostic to the method that generates the crude gradient estimate and can thus be potentially combined with various existing ES methods, such as (Salimans et al., 2017; Choromanski et al., 2019a;b). Through our numerical results we demonstrate that CoNES vastly outperforms various conventional blackbox optimizers on a suite of 5000-dimensional benchmark functions for blackbox optimizers: `Sphere`, `Rosenbrock`, `Rastrigin`, and `Lunacek`. We also demonstrate the improved sample complexity achieved by CoNES on the following OpenAI MuJoCo RL tasks: `HalfCheetah-v2`, `Walker2D-v2`, `Hopper-v2`, and `Swimmer-v2`.

## 2 Related Work

**Blackbox optimization.** Various engineering problems require optimizing systems for which the governing mechanisms are not explicitly known; e.g., system identification of complex physical systems (Amaran et al., 2016) and mechanism design (Audet & Kokkolaras, 2016). Blackbox optimization techniques such as Nelder-Mead (Nelder & Mead, 1965), evolutionary strategies (ES) (Rechenberg & Eigen, 1973), simulated annealing (Kirkpatrick et al., 1983), genetic algorithms (Holland, 1992), the cross-entropy method (De Boer et al., 2005), and covariance matrix adaptation (CMA) (Hansen, 2016) were developed to address such problems. Recently, the growing potential of these methods for training control policies with reinforcement learning has reignited interest in blackbox optimizers (Salimans et al., 2017; Mania et al., 2018; Choromanski et al., 2019a;b; Liu et al., 2019; Conti et al., 2018; Chatzilygeroudis et al., 2017; Ha, 2019). In this paper, we will primarily consider the class of blackbox optimizers that fall under the purview of ES.

**Evolutionary strategies for reinforcement learning.** In RL tasks, the advantages of ES – high parallelizability, better robustness, and richer exploration – were first demonstrated in (Salimans et al., 2017). Spurred by these findings, a plethora of recent developments aimed at improving ES for RL have emerged, some of which include: explicit novelty search regularization to avoid local minima (Conti et al., 2018), robustification of ES and efficient re-use of prior rollouts (Choromanski et al., 2019a), and adaptive sampling for the ES gradient estimate (Choromanski et al., 2019b). We remark that all the above papers focus on improving the ES MC gradient estimator. In contrast, this paper presents a method that *refines* the ES gradient estimate – regardless of where that estimate comes from – by solving a convex program.

**Natural gradient.** Our method is directly motivated by the concept of the natural gradient (Amari, 2016). The application of natural gradient in learning was initially pioneered in (Amari, 1998) and was later demonstrated to be effective for RL (Kakade, 2002), deep learning with backpropagation (Pascanu & Bengio, 2013), and blackbox optimization with ES (Sun et al., 2009; Wierstra et al., 2014). However, the latent potential of the natural gradient has not been completely realized due to the difficulty in estimation of the Fisher information matrix. Much of the prior work employing natural gradient has focused on efficient estimation or computation of the Fisher information matrix (Wu et al., 2017; Sun et al., 2009; Pascanu & Bengio, 2013). In contrast, CoNES does not work directly with the Fisher information matrix. Instead, we approximate the update direction by solving a convex program that maximizes the loss while being constrained to a KL-divergence ball around the current belief distribution; as the radius of the KL-divergence ball goes to zero, the limiting solution of this convex program corresponds to the natural gradient (see Proposition 1).

**Trust-regions for blackbox optimization.** Recent work on trust region methods for blackbox optimizers (Liu et al., 2019; Miyashita et al., 2018; Abdolmaleki et al., 2017) performs updates on the belief distribution by optimizing the loss on a KL-divergence ball. However, (Abdolmaleki et al.,

2017; Miyashita et al., 2018) perform the constrained optimization on a *discretization* of the belief distribution. The approach in (Liu et al., 2019) computes the KL-divergence for each dimension individually and bounds their maximum; the resulting optimization problem is approximated via a clipped surrogate objective similar to proximal policy optimization (PPO) (Schulman et al., 2017). In contrast, we *exactly* solve a KL-constrained problem whose solution approximates the natural gradient (as outlined above and formally discussed in Section 4.1) using powerful tools from convex optimization (e.g., second-order cone programming and geometric programming).

# 3 NOTATION

We denote a blackbox loss function by $\hat{l} : \mathcal{X} \to \mathbb{R}$ with $\mathcal{X} \subseteq \mathbb{R}^m$ as its domain. Let $P$ be a distribution on the domain $\mathcal{X}$ that signifies our *belief* of where the optimal candidate for $\hat{l}$ resides. We assume that $P$ belongs to the statistical manifold $\mathcal{P}$ (Suzuki, 2014) which is a Riemannian manifold (Petersen et al., 2006) of probability distributions. Any point $P \in \mathcal{P}$ is expressed in the coordinates $\theta \in \mathbb{R}^n$. Rather than optimizing $\hat{l}$ directly, we will work with the loss function $l : \mathcal{P} \to \mathbb{R}$ which provides the expected loss $P \mapsto \mathbb{E}_{x\sim P}[\hat{l}(x)]$ under the belief distribution $P$. When referring to the manifold in a coordinate-free setting, we express the loss as $l : \mathcal{P} \to \mathbb{R}$, whereas, when we work with a particular coordinate system on $\mathcal{P}$, we express the loss as $l : \mathbb{R}^n \to \mathbb{R}$; the abuse of notation creates no confusion as it will always be clear from context.

The (Euclidean) gradient operator is denoted by $\nabla$; the natural gradient operator is denoted by $\tilde{\nabla}$; and the solution of CoNES is denoted by $\hat{\nabla}$. The KL-divergence between two distributions is denoted by $\mathbb{D}(\cdot||\cdot)$ and the Euclidean inner product between two vectors is denoted by $\langle\cdot,\cdot\rangle$.

# 4 BACKGROUND

## 4.1 NATURAL GRADIENT

It is a commonly-held belief that the steepest ascent direction for a loss function $l : \mathcal{P} \to \mathbb{R}$ is given by its gradient $\nabla l$. However, this is only true if the domain $\mathcal{P}$ is expressed in an orthonormal coordinate system in a Euclidean space. If the space $\mathcal{P}$ admits a Riemannian manifold structure (Petersen et al., 2006), the steepest ascent direction is then given by the *natural gradient* $\tilde{\nabla} l$ instead (Amari, 2016, Section 12.1.2). Besides providing the steepest ascent direction on $\mathcal{P}$, the natural gradient possesses various attractive properties: (a) natural gradient is independent of the choice of coordinates $\theta$ on the statistical manifold $\mathcal{P}$; (b) natural gradient avoids saturation due to sigmoidal activation functions (Amari, 2016, Theorem 12.2); (c) online natural-gradient learning is asymptotically Fisher efficient, i.e., it asymptotically approaches equality of the Cramér-Rao bound (Amari, 1998). These qualities lay the foundation of our interest in leveraging the natural gradient in learning applications. In the rest of this section we will present two explicit characterizations of the natural gradient relevant to this paper.

Let $F(\theta)$ be the Fisher information matrix for the Reimannian manifold of distributions $\mathcal{P}$ described in the coordinates $\theta$; e.g., Gaussian distributions can be expressed in the coordinates $\theta = (\mu, \texttt{vec} \circ \texttt{upper-triangle}(\Sigma))$ where $\mu$, $\Sigma$ denote the mean and the covariance, respectively. The natural gradient then satisfies the following relation with the Euclidean gradient:

$$\tilde{\nabla} l(\theta) = F(\theta)^{-1} \nabla l(\theta) \ . \tag{1}$$

For the second characterization of the natural gradient we will need the Fisher-Rao norm $\|\cdot\|_F : \mathcal{P} \to [0, \infty)$ defined as $\|\theta\|_F := \sqrt{\langle\theta, F(\theta)\theta\rangle}$ (Liang et al., 2017, Definition 2). Using this norm we can express the natural gradient as follows:

**Proposition 1. [Adapted from (Ollivier et al., 2017, Proposition 1)]** *Let $\mathcal{P}$ be a statistical manifold, each point of which is a probability distribution $P_\theta$ parameterized by $\theta$. Let $l : \mathcal{P} \to \mathbb{R}$ be a loss function which maps a probability distribution $P_\theta$ to a scalar. Then, the natural gradient $\tilde{\nabla} l(\theta)$ of the loss function computed at any $\theta$ satisfies:*

$$\frac{\tilde{\nabla} l(\theta)}{\|\tilde{\nabla} l(\theta)\|_F} = \lim_{\epsilon \to 0} \ \underset{v \in \mathbb{R}^n}{\arg\max} \ l(\theta + \epsilon v) \quad \textit{s.t.} \ \ \mathbb{D}(P_{\theta+\epsilon v} || P_\theta) \leq \epsilon^2/2 \ . \tag{2}$$

Proposition 1 states that the natural gradient is aligned with the direction $v$ which maximizes the loss function in an infinitesimal KL-divergence ball around the current distribution $P_\theta$. To avoid confusion, it is worth clarfiying that the maximization in Proposition 1 computes the natural gradient which can then be passed to a gradient-based optimizer to *minimze the loss*.

**Remark 1.** *Proposition 1 also holds true for the linear approximation of the loss function $l(\theta + \epsilon v)$ at $\theta$. Intuitively, the reason for this is that the linear approximation locally converges to the loss function for arbitrarily small $\epsilon > 0$.*

### 4.2 Natural Evolutionary Strategies

The evolutionary strategies (ES) framework performs a Monte-Carlo estimate of the gradient of the loss with respect to the belief distribution (Wierstra et al., 2014, Section 2):

$$\nabla l(\theta) = \nabla \mathop{\mathbb{E}}_{x \sim P_\theta}[\hat{l}(x)] = \mathop{\mathbb{E}}_{x \sim P_\theta}[\hat{l}(x) \nabla \ln P_\theta(x)] \ . \tag{3}$$

This gradient estimate is then supplied to a gradient-based optimizer to update the belief distribution. Note that equation 3 provides an estimate of the Euclidean gradient. Instead of using the Euclidean gradient equation 3, Natural Evolutionary Strategies (NES) (Wierstra et al., 2014; Sun et al., 2009) estimates the natural gradient by transforming the Euclidean gradient estimate equation 3 through equation 1.

## 5 Convex Natural Evolutionary Strategies

Despite the various advantages offered by the natural gradient, the computationally expensive estimation of the Fisher information matrix $F(\theta)$ and its inverse makes it difficult to scale to very high-dimensional problems. Proposition 1 offers an alternative to compute the natural gradient while obviating the need to estimate $F(\theta)$; however, equation 2 is a challenging non-convex optimization problem. To develop CoNES we "massage" the optimization problem in equation 2 into an efficiently-solvable convex program.

We begin by relaxing the requirement $\lim \epsilon \to 0$ and instead choosing a fixed $\epsilon > 0$, resulting in the following optimization problem:[1]

$$v^*(\theta) \in \arg\max_v \{l(\theta + \epsilon v) \mid \mathbb{D}(P_{\theta+\epsilon v} || P_\theta) \leq \epsilon^2, v \in \mathbb{R}^n\}, \tag{4}$$

where $\epsilon$ is now a hyperparameter which can be as large as necessary. Using $v^*(\theta)$ as the update direction could yield faster convergence than $\tilde{\nabla} l(\theta)$. This may seem counter-intuitive because the natural gradient is the steepest ascent direction, as discussed in Section 4.1; however, it is worth noting that this holds true only for an infinitesimal step length. The flexibility of choosing an $\epsilon$ permits us to align the search for the steepest ascent direction with the desired step-length of the update, yielding rapid convergence.

We are interested in settings where the landscape of the loss function $l$ is unknown and querying loss values of individual candidates is expensive. Even if the analytical form of $l$ was available to us, equation 4 may be a non-convex problem and hence challenging to solve. To make this problem more tractable, we perform a Taylor expansion of the loss function $l(\theta + \epsilon v) \approx l(\theta) + \langle \nabla l(\theta), \epsilon v \rangle$ and work with the following optimization problem:

$$v^*(\theta) \in \arg\max_v \{l(\theta) + \langle \nabla l(\theta), \epsilon v \rangle \mid \mathbb{D}(P_{\theta+\epsilon v} || P_\theta) \leq \epsilon^2, v \in \mathbb{R}^n\}. \tag{5}$$

In equation 5, $l(\theta)$ is a constant offset which does not affect the choice of $v$ and can hence be ignored. Further, we denote $\delta\theta := \epsilon v$ and restate equation 5 as:

$$\hat{\nabla} l(\theta; \epsilon) \in \arg\max_{\delta\theta} \{\langle \nabla l(\theta), \delta\theta \rangle \mid \mathbb{D}(P_{\theta+\delta\theta} || P_\theta) \leq \epsilon^2, \delta\theta \in \mathbb{R}^n\}. \tag{6}$$

Despite these relaxations, the optimization problem equation 6 may still be intractable due to the lack of convexity of the feasible set. However, in the following theorem we establish for the Gaussian family of probability distributions that equation 6 is convex and can be solved in *polynomial time*.

[1] Without loss of generality, we are replacing $\epsilon^2/2$ with $\epsilon^2$.

**Theorem 1.** *The optimization problem in equation 6 is:*

- *a semidefinite program (SDP) with an additional exponential cone constraint if $\mathcal{P}$ is the space of Gaussian distributions;*
- *a second-order cone program (SOCP) with an additional exponential cone constraint if $\mathcal{P}$ is the space of Gaussian distributions with diagonal covariance.*

*Proof.* As the objective function of equation 6 is linear, we only need to verify the convexity of the feasible set. We will first consider the case when $\mathcal{P}$ is the space of Gaussian distributions. Let $P_{\theta+\delta\theta} = \mathcal{N}(\mu, \Sigma)$ and $P_\theta = \mathcal{N}(\mu_0, \Sigma_0)$. Then:

$$\mathbb{D}(P_{\theta+\delta\theta}||P_\theta) = \frac{1}{2}\left(\mathrm{Tr}(\Sigma_0^{-1}\Sigma) + (\mu-\mu_0)^{\mathrm{T}}\Sigma_0^{-1}(\mu-\mu_0) - \log\det(\Sigma) + \log\det(\Sigma_0) - n\right) \quad (7)$$

which is convex because $\mathrm{Tr}(\Sigma_0^{-1}\Sigma)$ is linear, $(\mu-\mu_0)^{\mathrm{T}}\Sigma_0^{-1}(\mu-\mu_0)$ is positive-definite quadratic, and $-\log\det(\Sigma)$ is convex. Finally, noting that $\log\det$ constraints can be formulated as an SDP with an additional exponential cone constraint (log) completes the proof of this part.

Now we consider the family of Gaussian distributions $P_{\theta+\delta\theta} = \mathcal{N}(\mu, \Sigma)$ and $P_\theta = \mathcal{N}(\mu_0, \Sigma_0)$ with diagonal covariance. We denote the mean as $\mu = (\mu_1, \cdots, \mu_n)$ and $\mu_0 = (\mu_{0,1}, \cdots, \mu_{0,n})$. The diagonal elements of the covariance $\Sigma$ and $\Sigma_0$ are expressed as $(\sigma_1, \cdots, \sigma_n)$ and $(\sigma_{0,1}, \cdots, \sigma_{0,n})$, respectively. Then, the KL-divergence between two distributions in this family is:

$$\mathbb{D}(P_{\theta+\delta\theta}||P_\theta) = -\frac{1}{2}\sum_{i=1}^{n}\left(1 + \log\sigma_i^2 - \log\sigma_{0,i}^2 - \frac{(\mu_i-\mu_{0,i})^2}{\sigma_{0,i}^2} - \frac{\sigma_i^2}{\sigma_{0,i}^2}\right) . \quad (8)$$

From equation 8, it follows that equation 6 for this family of distributions is an SOCP with an additional exponential cone constraint (that arises from the $\log$ terms), completing the proof. □

Restricting the class of belief distributions to those in Theorem 1 gives rise to CoNES: a family of convex programs that draws motivation from the concept of the natural gradient to transform the Euclidean gradient. To geometrically visualize CoNES, consider the illustration in Fig. 1. The orange surface is the loss landscape and the gray surface is the linearization of the loss at the point denoted by $\theta$; in differential geometric terms, the orange surface is more accurately characterized as the manifold given by the graph of the loss $l(\theta)$ while the gray surface is the manifold's tangent space at $(\theta, l(\theta))$. The green arrow represents the solution of CoNES for a KL-divergence ball (light green region) with a very small $\epsilon$ which can also be regarded as the natural gradient (modulo the norm) at $\theta$ by Remark 1. The red arrow is the solution of CoNES for a KL-divergence ball (light red region) with a larger $\epsilon$. Note that this figure is an illustration; the KL-divergence balls may not necessarily manifest in the depicted shapes. The NES gradient is the sharpest ascent direction for an infinitesimal step size, but, it may not be ideal for a larger step size. With CoNES, we can tune the scalar parameter $\epsilon$ to better align the update direction with the gradient-based optimizer's step size (learning rate), yielding faster updates. Indeed, the choice of $\epsilon$ is important to the performance of CoNES as demonstrated in our numerical results in Section 7.2. The mechanism for selecting (or adapting) the hyperparameter $\epsilon$ is beyond the scope of this paper and will be explored in our future work.

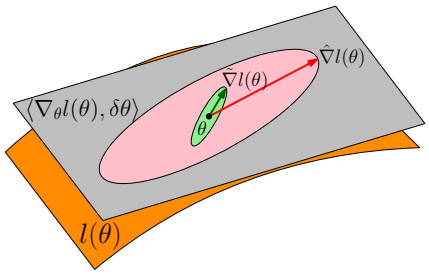


Figure 1: Geometric illustration of CoNES.

The psuedo-code for our implementation of CoNES as a blackbox optimizer is detailed in Algorithm 1. We use the ES gradient estimate (presented in Section 4.2) as the GRADIENT-ESTIMATOR in Line 5 of Algorithm 1; any estimator of the Euclidean gradient, such as (Choromanski et al., 2019a;b), can be used here. We use Adam (Kingma & Ba, 2014) as our gradient-based optimizer in Line 7; any gradient-based optimizer can be used.

## 6 PARAMETERIZATION INVARIANCE OF CONES

An important property of the natural gradient is its independence to the parameterization of the belief distribution; e.g., for Gaussian distributions it does not matter whether we use the covariance matrix

**Algorithm 1** CoNES

1: Hyperparameters: radius $\epsilon$ of KL-divergence ball, number of candidates $N$ drawn at each iteration
2: Initialize: $\theta \leftarrow \theta_0$, OPTIMIZER
3: **repeat**
4: $\quad \{\hat{x}_i\}_{i=1}^N \leftarrow$ Draw $N$ samples from the belief distribution $P_\theta$
5: $\quad \nabla_\theta l(\theta) \leftarrow$ GRADIENT-ESTIMATOR$(\{x_i\}_{i=1}^N, \{\hat{l}(x_i)\}_{i=1}^N)$
6: $\quad \hat{\nabla}_\theta l(\theta) \leftarrow$ CONES$(\nabla_\theta l(\theta), \epsilon)$ ▷ solve equation 6
7: $\quad \theta \leftarrow$ OPTIMIZER$(\theta, \hat{\nabla}_\theta l(\theta))$
8: **until** Termination conditions satisfied
9: **return** $\theta$

or its Cholesky decomposition. The natural gradient inherits this property by construction as the covariant gradient on the statistical manifold (Amari, 2016). Parameterization invariance ensures that we are working with the intrinsic mathematical objects (probability distributions here) and the specific encoding of these objects will not affect the outcome. From a practical perspective, we derive the benefit of fewer properties to "engineer".

A natural question to ask is whether CoNES (equation 6) exhibits the same property. Proposition 1 ensures that the CoNES optimization exhibits this property in the limit of $\epsilon$ tending to zero, as the update direction then coincides with the natural gradient. However, establishing this property for an arbitrary $\epsilon > 0$ is not immediately obvious.

We will work with the loss function $l$ rather than its linearization with the understanding that if the parameterization invariance holds for an arbitrary function $l$, it will automatically hold for the linear function in equation 6. With a slight abuse of notation, we will express the loss function $l : \mathbb{R}^n \to \mathbb{R}$ in the coordinates $\theta$ on the statistical manifold instead of the coordinate-free notation of $l : \mathcal{P} \to \mathbb{R}$. Now we are ready to present the main result of this section:

**Theorem 2.** *Consider the optimization problem:*

$$OPT_\theta : l_\theta^* = \max\{l(\theta + \epsilon v_\theta) \mid \mathbb{D}(P_{\theta+\epsilon v_\theta} || P_\theta) \leq \epsilon^2, v_\theta \in \mathbb{R}^n\}. \tag{9}$$

*Let $\Phi : \mathbb{R}^n \to \mathbb{R}^n$ be a smooth invertible mapping which performs a coordinate change from $\theta \mapsto \phi := \Phi(\theta)$. Consider the following optimization problem $OPT_\phi$ in the new coordinates:*[2]

$$OPT_\phi : l_\phi^* = \max\{l \circ \Phi^{-1}(\phi + \epsilon v_\phi) \mid \mathbb{D}(P_{\phi+\epsilon v_\phi} || P_\phi) \leq \epsilon^2, v_\phi \in \mathbb{R}^n\}. \tag{10}$$

*Then, there exists an invertible mapping $\Phi_v : \mathbb{R}^n \to \mathbb{R}^n$ such that $v_\theta^* \in \arg\max_v OPT_\theta \iff \Phi_v(v_\theta^*) \in \arg\max_v OPT_\phi$, ensuring that $l_\theta^* = l_\phi^*$.*

Theorem 2 shows that expressing the belief distribution $P \in \mathcal{P}$ in different coordinates $\theta$ or $\phi$ provides the same optimal loss and the same set of possible solutions (upto a bijective mapping). Of course, we cannot ensure that the solution, i.e., the $\arg\max$ of the CoNES optimization is the same due to the potential lack of uniqueness of the optima; e.g., consider the maximization of $x_1^2 + x_2^2$ in $x_1^2 + x_2^2 \leq 1$ initialized at $(x_1, x_2) = (0, 0)$ – all directions $v$ from the initial point are equally good.

Intuitively, Theorem 2 holds because the KL-divergence is independent of the parameterization of the distribution (Kullback & Leibler, 1951, Corollary 4.1), i.e., for $\theta$, $\phi$, and $\Phi$ as defined in Theorem 2, we have:

$$\mathbb{D}(P_{\theta+\epsilon v_\theta} || P_\theta) = \mathbb{D}(P_{\Phi(\theta+\epsilon v_\theta)} || P_{\Phi(\theta)}) \ . \tag{11}$$

The proof of Theorem 2 is detailed in Appendix A.

## 7 RESULTS

In this section, we use CoNES on two classes of problems: (a) a standard suite of high-dimensional loss functions used to benchmark blackbox optimizers, and (b) a selection of OpenAI's MuJoCo suite of RL tasks. We compare CoNES against existing methods including ES, natural evolutionary strategies (NES), and covariance matrix adaptation (CMA). We custom-implemented ES, NES, and

[2]From a geometric perspective, $\theta$ and $\phi$ are coordinates on the statistical manifold $\mathcal{P}$, either of which can be used to express a distribution $P \in \mathcal{P}$. The directions $v_\theta$ and $v_\phi$ lie in the tangent space $T_P\mathcal{P}$ of $\mathcal{P}$ at $P$.

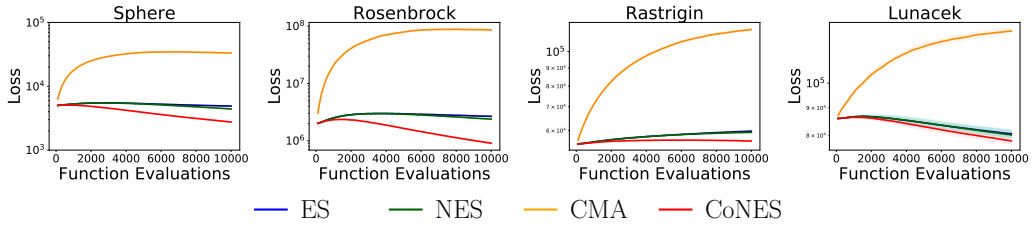


Figure 2: Average loss (solid curve) with standard deviation (shaded region) across 10 seeds for ES, NES, CMA, and CoNES on `Sphere`, `Rosenbrock`, `Rastrigin`, and `Lunacek`.

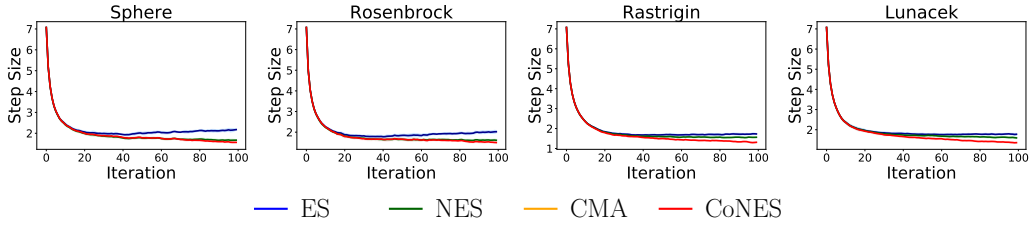


Figure 3: Average step size (solid) with standard deviation (shaded region) of the belief distribution's mean across 10 seeds for ES, NES, and CoNES on `Sphere`, `Rosenbrock`, `Rastrigin`, and `Lunacek`.

CoNES, while CMA is adapted directly from the open-source PyCMA package (Hansen et al., 2019); our code is available in the supplementary material.

The family of Gaussian belief distributions with diagonal covariance is used for ES, NES, and CoNES. This family of belief distributions permits the implementation of NES *exactly* (i.e., without having to numerically estimate the Fisher information matrix (Sun et al., 2009)) for high-dimensional problems, serving as a strong baseline to compare CoNES against. For CMA, PyCMA's default family of belief distributions – Gaussian distributions with non-diagonal covariance – is used. For ES, NES, and CoNES we compute an estimate of the gradient direction and pass it to the Adam optimizer (Kingma & Ba, 2014) to update the belief distribution. For each of these methods we perform antithetic sampling and rank-based fitness transformation (Salimans et al., 2017). Unlike (Salimans et al., 2017), we also update the variance of the belief distribution; we circumvent the non-negativeness constraint of the variance by updating the log of variance with the Adam optimizer instead. The resulting convex optimization problems for CoNES are solved using the CVXPY package (Diamond & Boyd, 2016) and the MOSEK solver (MOSEK ApS, 2019).

### 7.1 Benchmark Functions

We first test our approach on four 5000-dimensional functions: `Sphere`, `Rosenbrock`, `Rastrigin`, and `Lunacek` (Hansen et al., 2009) which are provided in Appendix C. These functions are commonly-used benchmarks for blackbox optimization methods (Hansen et al., 2016; Teytaud & Rapin, 2018). Hyperparameters for ES, NES, and CoNES are shared across all problems (see Appendix B) while the hyperparameters of CMA are the default values chosen by PyCMA. Training for these benchmark functions was performed on a desktop with a 3.30 GHz Intel i9-7900X CPU with 10 cores and 32 GB RAM. Fig. 2 plots the average and standard deviation (shaded region) of the loss curves across 10 seeds. The rapid drop of the loss for CoNES demonstrates significant benefits in terms of the sample complexity over other methods. Fig. 3 shows that the step size for CoNES is smaller than ES and NES, which coupled with its lower loss implies that the update direction for CoNES is more accurate than ES and NES. The run-time for a single seed is $\sim$1 minute for ES and NES, $\sim$5 minutes for CoNES, and $\sim$35 minutes for CMA.

### 7.2 Reinforcement Learning Tasks

Next, we benchmark our approach on the following environments from the OpenAI Gym suite of RL problems: `HalfCheetah-v2`, `Walker2D-v2`, `Hopper-v2`, and `Swimmer-v2`. We employ a fully-connected neural network policy with `tanh` activations possessing one hidden layer with 16 neurons for `Swimmer-v2` and 50 neurons for all other environments. The input to the policies are the agent's state – which are normalized using a method similar to the one adopted by (Mania et al., 2018) – and the output is a vector in the agent's action space. The training for these tasks was performed on a `c5.24xlarge` instance on Amazon Web Services (AWS). Fig. 4

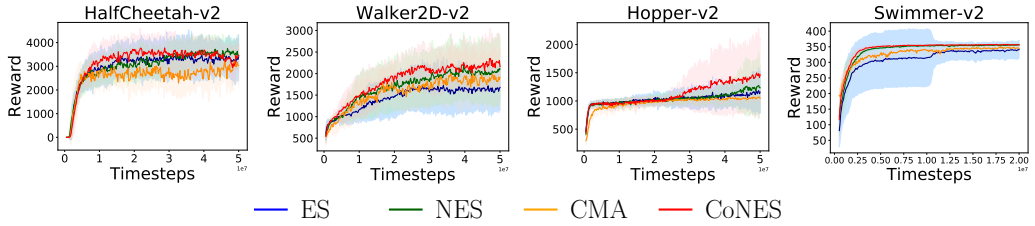


Figure 4: Average reward (solid curve) with standard deviation (shaded region) across 10 seeds for ES, NES, CMA, and CoNES on `HalfCheetah-v2`, `Walker2D-v2`, `Hopper-v2`, and `Swimmer-v2`.

| Environments | Target Avg. Reward | # Timesteps to attain target average reward | | | |
|---|---|---|---|---|---|
| | | ES | NES | CMA | CoNES |
| `HalfCheetah-v2` | 3500 | $3.23 \times 10^7$ | $3.01 \times 10^7$ | – | $\mathbf{1.40 \times 10^7}$ |
| `Walker2D-v2` | 2000 | – | $3.07 \times 10^7$ | – | $\mathbf{2.10 \times 10^7}$ |
| `Hopper-v2` | 1400 | – | – | – | $\mathbf{4.15 \times 10^7}$ |
| `Swimmer-v2` | 340 | $1.99 \times 10^7$ | $3.60 \times 10^6$ | $8.33 \times 10^6$ | $\mathbf{3.01 \times 10^6}$ |

Table 1: Timesteps to attain a target average reward (over 10 seeds) for RL tasks. For each environment the timestep for the best performing blackbox method is displayed in bold. Hyphen ( – ) is used for the method that failed to achieve the target average reward in $2 \times 10^7$ timesteps for `Swimmer-v2` and $5 \times 10^7$ timesteps for all other environments.

presents the average and standard deviation of the rewards for each RL task across 10 seeds against the number of time-steps interacted with the environment. Fig. 4 as well as Table 1 illustrate that CoNES performs well on all these tasks. For each environment we share the same hyperparameters (excluding $\epsilon$) between ES, NES, and CoNES; for CMA we use the default hyperparameters as chosen by PyCMA. It is worth pointing out that for RL tasks, CoNES demonstrates high sensitivity to the choice of $\epsilon$. The results for CoNES reported in Fig. 4 and Table 1 are for the best choice of $\epsilon$ from $[\sqrt{0.1}, \sqrt{1}, \sqrt{10}, \sqrt{100}, \sqrt{1000}]$. Exact hyperparameters for the problems are provided in Appendix B. Each seed of `HalfCheetah-v2`, `Walker2D-v2` and `Hopper-v2`, takes $\sim$4-5 hours with ES, NES, CoNES and $\sim$10 hours with CMA. Each seed of `Swimmer-v2` takes $\sim$2 hours with ES, NES, CoNES and $\sim$4 hours with CMA.

## 8 Conclusions and Future Work

We presented convex natural evolutionary strategies (CoNES) for optimizing high-dimensional blackbox functions. CoNES combines the notion of the natural gradient from information geometry with powerful techniques from convex optimization (e.g., second-order cone programming and geometric programming). In particular, CoNES refines a gradient estimate by solving a convex program that searches for the direction of steepest ascent in a KL-divergence ball around the current belief distribution. We formally established that CoNES is invariant under transformations of the belief parameterization. Our numerical results on benchmark functions and RL examples demonstrate the ability of CoNES to converge faster than conventional blackbox methods such as ES, NES, and CMA.

**Future Work.** This paper raises numerous exciting future directions to explore. The performance of CoNES is dependent on the choice of the radius $\epsilon^2$ of the KL-divergence ball. Furthermore, a suitable choice of $\epsilon$ in one region of the loss landscape may not be suitable for another. Hence, an adaptive scheme for choosing the radius of the KL-divergence ball could substantially enhance the performance of CoNES. Another potentially fruitful future direction arises from the observation that Proposition 1 — which serves as the cornerstone of CoNES — holds for any[3] $f$-divergence (Csiszár & Shields, 2004). Hence, we can generalize CoNES to arbitrary $f$-divergences; this may afford greater flexibility in tuning it for the specific loss landscape and further improving performance. We can increase the flexibility afforded by CoNES even more by expanding beyond the family of Gaussian belief distributions. Finally, we are also exploring the empirical benefits of adaptively restricting the covariance matrix model (Akimoto & Hansen, 2016; Choromanski et al., 2019b) in order to further enhance sample complexity.

[3]This is an outcome of the fact that the Hessian of all $f$-divergences is the Fisher information (Makur, 2015).

## REFERENCES


Mosek modeling cook-book: Log-determinant. https://docs.mosek.com/modeling-cookbook/sdo.html#log-determinant.

Abbas Abdolmaleki, Bob Price, Nuno Lau, Luis Paulo Reis, and Gerhard Neumann. Deriving and improving CMA-ES with information geometric trust regions. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pp. 657–664, 2017.

Youhei Akimoto and Nikolaus Hansen. Projection-based restricted covariance matrix adaptation for high dimension. In *Proceedings of the Genetic and Evolutionary Computation Conference 2016*, pp. 197–204, 2016.

Satyajith Amaran, Nikolaos V Sahinidis, Bikram Sharda, and Scott J Bury. Simulation optimization: a review of algorithms and applications. *Annals of Operations Research*, 240(1):351–380, 2016.

Shun-Ichi Amari. Natural gradient works efficiently in learning. *Neural computation*, 10(2):251–276, 1998.

Shun-ichi Amari. *Information Geometry And Its Applications*, volume 194. Springer, 2016.

Charles Audet and Michael Kokkolaras. Blackbox and derivative-free optimization: theory, algorithms and applications, 2016.

Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

Stephen Boyd, Seung-Jean Kim, Lieven Vandenberghe, and Arash Hassibi. A tutorial on geometric programming. *Optimization and Engineering*, 8(1):67, 2007.

Konstantinos Chatzilygeroudis, Roberto Rama, Rituraj Kaushik, Dorian Goepp, Vassilis Vassiliades, and Jean-Baptiste Mouret. Black-box data-efficient policy search for robotics. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 51–58. IEEE, 2017.

Krzysztof Choromanski, Aldo Pacchiano, Jack Parker-Holder, Yunhao Tang, Deepali Jain, Yuxiang Yang, Atil Iscen, Jasmine Hsu, and Vikas Sindhwani. Provably robust blackbox optimization for reinforcement learning. *arXiv:1903.02993*, 2019a.

Krzysztof M Choromanski, Aldo Pacchiano, Jack Parker-Holder, Yunhao Tang, and Vikas Sindhwani. From complexity to simplicity: Adaptive ES-active subspaces for blackbox optimization. In *Advances in Neural Information Processing Systems*, pp. 10299–10309, 2019b.

Edoardo Conti, Vashisht Madhavan, Felipe Petroski Such, Joel Lehman, Kenneth Stanley, and Jeff Clune. Improving exploration in evolution strategies for deep reinforcement learning via a population of novelty-seeking agents. In *Advances in Neural Information Processing Systems*, pp. 5027–5038, 2018.

Imre Csiszár and Paul C Shields. *Information Theory and Statistics: A Tutorial*. Now Publishers Inc, 2004.

Pieter-Tjerk De Boer, Dirk P Kroese, Shie Mannor, and Reuven Y Rubinstein. A tutorial on the cross-entropy method. *Annals of Operations Research*, 134(1):19–67, 2005.

Steven Diamond and Stephen Boyd. CVXPY: A Python-embedded modeling language for convex optimization. *Journal of Machine Learning Research*, 17(83):1–5, 2016.

Daniel Golovin, Benjamin Solnik, Subhodeep Moitra, Greg Kochanski, John Karro, and D Sculley. Google vizier: A service for black-box optimization. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 1487–1495, 2017.

David Ha. Reinforcement learning for improving agent design. *Artificial Life*, 25(4):352–365, 2019.

Nikolaus Hansen. The CMA evolution strategy: A tutorial. *arXiv preprint arXiv:1604.00772*, 2016.

Nikolaus Hansen, Steffen Finck, Raymond Ros, and Anne Auger. Real-parameter black-box optimization benchmarking 2009: Noiseless functions definitions. *[Research Report] RR-6829, INRIA*, 2009. inria00362633v2.

Nikolaus Hansen, Anne Auger, Olaf Mersmann, Tea Tusar, and Dimo Brockhoff. COCO: A platform for comparing continuous optimizers in a black-box setting. *arXiv preprint arXiv:1603.08785*, 2016.

Nikolaus Hansen, Youhei Akimoto, and Petr Baudis. CMA-ES/pycma on Github. Zenodo, DOI:10.5281/zenodo.2559634, February 2019. URL `https://doi.org/10.5281/zenodo.2559634`.

John Henry Holland. *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence*. MIT Press, 1992.

Frank Hutter, Lars Kotthoff, and Joaquin Vanschoren. *Automated Machine Learning*. Springer, 2019.

Sham M Kakade. A natural policy gradient. In *Advances in Neural Information Processing Systems*, pp. 1531–1538, 2002.

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

Scott Kirkpatrick, C Daniel Gelatt, and Mario P Vecchi. Optimization by simulated annealing. *Science*, 220(4598):671–680, 1983.

Solomon Kullback and Richard A Leibler. On information and sufficiency. *The Annals of Mathematical Statistics*, 22(1):79–86, 1951.

Tengyuan Liang, Tomaso Poggio, Alexander Rakhlin, and James Stokes. Fisher-Rao metric, geometry, and complexity of neural networks. *arXiv preprint arXiv:1711.01530*, 2017.

Guoqing Liu, Li Zhao, Feidiao Yang, Jiang Bian, Tao Qin, Nenghai Yu, and Tie-Yan Liu. Trust region evolution strategies. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pp. 4352–4359, 2019.

Anuran Makur. *A study of local approximations in information theory*. PhD thesis, Massachusetts Institute of Technology, 2015.

Horia Mania, Aurelia Guy, and Benjamin Recht. Simple random search provides a competitive approach to reinforcement learning. *arXiv preprint arXiv:1803.07055*, 2018.

Megumi Miyashita, Shiro Yano, and Toshiyuki Kondo. Mirror descent search and its acceleration. *Robotics and Autonomous Systems*, 106:107–116, 2018.

MOSEK ApS. Mosek fusion api for python 9.0.84(beta), 2019. URL `https://docs.mosek.com/9.0/pythonfusion/index.html`.

John A Nelder and Roger Mead. A simplex method for function minimization. *The Computer Journal*, 7(4):308–313, 1965.

Yann Ollivier, Ludovic Arnold, Anne Auger, and Nikolaus Hansen. Information-geometric optimization algorithms: A unifying picture via invariance principles. *The Journal of Machine Learning Research*, 18(1):564–628, 2017.

Paolo Pagliuca, Nicola Milano, and Stefano Nolfi. Efficacy of modern neuro-evolutionary strategies for continuous control optimization. *arXiv preprint arXiv:1912.05239*, 2019.

Razvan Pascanu and Yoshua Bengio. Revisiting natural gradient for deep networks. *arXiv preprint arXiv:1301.3584*, 2013.

Peter Petersen, S Axler, and KA Ribet. *Riemannian Geometry*, volume 171. Springer, 2006.

I. Rechenberg and M. Eigen. Evolutionsstrategie: Optimierung technischer systeme nach prinzipien der biologischen evolution. *Frommann-Holzboog Stuttgart*, 1973.

Tim Salimans, Jonathan Ho, Xi Chen, Szymon Sidor, and Ilya Sutskever. Evolution strategies as a scalable alternative to reinforcement learning. *arXiv preprint arXiv:1703.03864*, 2017.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Yi Sun, Daan Wierstra, Tom Schaul, and Juergen Schmidhuber. Efficient natural evolution strategies. In *Proceedings of the Annual Conference on Genetic and evolutionary computation*, pp. 539–546, 2009.

Mashbat Suzuki. Information geometry and statistical manifold. *arXiv preprint arXiv:1410.3369*, 2014.

O Teytaud and J Rapin. Nevergrad: An open source tool for derivative-free optimization, 2018.

Daan Wierstra, Tom Schaul, Tobias Glasmachers, Yi Sun, Jan Peters, and Jürgen Schmidhuber. Natural evolution strategies. *The Journal of Machine Learning Research*, 15(1):949–980, 2014.

Yuhuai Wu, Elman Mansimov, Roger B Grosse, Shun Liao, and Jimmy Ba. Scalable trust-region method for deep reinforcement learning using kronecker-factored approximation. In *Advances in Neural Information Processing Systems*, pp. 5279–5288, 2017.

# APPENDIX

## A PROOF OF THEOREM 2

To formally prove Theorem 2, we will first establish two lemmas. The first lemma shows the existence of a bijective mapping between $v_\theta$ and $v_\phi$.

**Lemma 1.** *Let $\theta$, $\phi$, and $\Phi$ be as defined in Theorem 2. Then, there exists a bijective mapping $\Phi_v : \mathbb{R}^n \to \mathbb{R}^n$, defined as*

$$v_\theta \mapsto \frac{\Phi(\theta + \epsilon v_\theta) - \Phi(\theta)}{\epsilon}. \tag{12}$$

*Proof.* First we will check the injectivity of $\Phi_v$:

$$\Phi_v(v_\theta) = \Phi_v(v'_\theta) \iff \Phi(\theta + \epsilon v_\theta) = \Phi(\theta + \epsilon v'_\theta) \iff v_\theta = v'_\theta \text{ (since } \Phi \text{ is injective).} \tag{13}$$

Next, to check the surjectivity of $\Phi_v$, let $v_\phi \in \mathbb{R}^n$ be arbitrary. Then there exists $v_\theta := (\Phi^{-1}(\Phi(\theta) + \epsilon v_\phi) - \theta)/\epsilon$ which satisfies $\Phi_v(v_\theta) = v_\phi$. □

In the following remark, we express the result of Lemma 1 in a form that is more conducive to our forthcoming proof.

**Remark 2.** *Lemma 1 ensures that the following relation holds for any $v_\theta \in \mathbb{R}^n$:*

$$v_\phi = \Phi_v(v_\theta) \iff \phi + \epsilon v_\phi = \Phi(\theta + \epsilon v_\theta) \iff \theta + \epsilon v_\theta = \Phi^{-1}(\phi + \epsilon v_\phi)$$

*where the first equivalence relation holds by using the expression of $\Phi_v$ equation 12 and the second equivalence relations hold from the bijectivity of $\Phi$.*

**Lemma 2.** *Let $B_\theta := \{v \in \mathbb{R}^n \mid \mathbb{D}(P_{\theta+\epsilon v}||P_\theta) \leq \epsilon^2\}$ and $B_\phi := \{v \in \mathbb{R}^n \mid \mathbb{D}(P_{\phi+\epsilon v}||P_\phi) \leq \epsilon^2\}$ be the feasible sets of $OPT_\theta$ and $OPT_\phi$, respectively. Let $\Phi_v$ be defined as in Lemma 1. Then, $B_\phi = \{\Phi_v(v) \mid v \in B_\theta\}$.*

*Proof.* Let $v_\phi \in \{\Phi_v(v) \mid v \in B_\theta\}$, then there exists a $v_\theta \in B_\theta$ such that $v_\phi = \Phi_v(v_\theta)$. Therefore, Remark 2 ensures that $\phi + \epsilon v_\phi = \Phi(\theta + \epsilon v_\theta)$, which further gives us:

$$\mathbb{D}(P_{\phi+\epsilon v_\phi}||P_\phi) = \mathbb{D}(P_{\Phi(\theta+\epsilon v_\theta)}||P_{\Phi(\theta)}) = \mathbb{D}(P_{\theta+\epsilon v_\theta}||P_\theta) \leq \epsilon^2, \tag{14}$$

where the last equality follows from equation 11 and the inequality follows from the fact that $v_\theta \in B_\theta$. From equation 14 we have that $v_\phi \in B_\phi$ implying that $\{\Phi_v(v) \mid v \in B_\theta\} \subseteq B_\phi$.

Now, let $v_\phi \in B_\phi$. By the surjectivity of $\Phi_v$ from Lemma 1, there exists a $v_\theta \in \mathbb{R}^n$ such that $v_\phi = \Phi_v(v_\theta)$. With this, Remark 2 ensures that $\phi + \epsilon v_\phi = \Phi(\theta + \epsilon v_\theta)$. Hence, using equation 11, followed by $\phi + \epsilon v_\phi = \Phi(\theta + \epsilon v_\theta)$ gives:

$$\mathbb{D}(P_{\theta+\epsilon v_\theta}||P_\theta) = \mathbb{D}(P_{\Phi(\theta+\epsilon v_\theta)}||P_{\Phi(\theta)}) = \mathbb{D}(P_{\phi+\epsilon v_\phi}||P_\phi) \leq \epsilon^2 \tag{15}$$

where the last inequality follows from the fact that $v_\phi \in B_\phi$. Therefore, by equation 15, we have that $v_\theta \in B_\theta$, which, on combining with the earlier assertion that $v_\phi = \Phi_v(v_\theta)$ implies that $v_\phi \in \{\Phi_v(v) \mid v \in B_\theta\}$. Thereby, ensuring that $B_\phi \subseteq \{\Phi_v(v) \mid v \in B_\theta\}$ and completing the proof. □

**Proof of Theorem 2.** The proof follows from the following chain of arguments:

$$\begin{aligned}
v^*_\theta \in \arg\max_v OPT_\theta &\iff l(\theta + \epsilon v^*_\theta) \geq l(\theta + \epsilon v_\theta),\ \forall v_\theta \in B_\theta && (16)\\
&\iff l \circ \Phi^{-1}(\phi + \epsilon \Phi_v(v^*_\theta)) \geq l \circ \Phi^{-1}(\phi + \epsilon \Phi_v(v_\theta)),\ \forall v_\theta \in B_\theta && (17)\\
&\iff l \circ \Phi^{-1}(\phi + \epsilon \Phi_v(v^*_\theta)) \geq l \circ \Phi^{-1}(\phi + \epsilon v_\phi),\ \forall v_\phi \in B_\phi && (18)\\
&\iff \Phi_v(v^*_\theta) \in \arg\max_v OPT_\phi\ , && (19)
\end{aligned}$$

where equation 17 follows from Remark 2 (Lemma 1) and equation 18 follows from Lemma 2. Further, because $l(\theta + \epsilon v^*_\theta) = l \circ \Phi^{-1}(\phi + \epsilon \Phi_v(v^*_\theta))$ from Remark 2, we get $l^*_\theta = l^*_\phi$. □

## B HYPERPARAMETERS

The parameters for the Adam optimizer were chosen according to (Kingma & Ba, 2014, Algorithm 1) for all results in Section 7.

**Benchmark Functions.** For all the results in Section 7.1 the initial belief distribution is chosen to be the normal distribution $\mathcal{N}(0, I)$. The hyperparameters for ES, NES and CoNES were chosen as follows: the number of function evaluations performed per iteration is 100 and the learning rate for the mean and log of the variance is 0.1. Additionally, $\epsilon$ is set to 100 for CoNES.

**RL Tasks.** The hyperparameters for ES, NES, and CoNES for the results in Section 7.2 are detailed in Table 2 below; some of these hyperparameters were borrowed from (Pagliuca et al., 2019).

| Environments | Initial Distribution mean ($\mu$) | Initial Distribution std ($\sigma$) | Learning Rate $\mu$ | Learning Rate $\log(\sigma^2)$ | # policies evaluated per itr (N) | # envs interacted per policy (m) | $\epsilon$ |
|---|---|---|---|---|---|---|---|
| `HalfCheetah-v2` | 0 | 0.02 | 0.01 | 0.01 | 40 | 1 | $\sqrt{1000}$ |
| `Walker2D-v2` | 0 | 0.02 | 0.01 | 0.01 | 40 | 1 | $\sqrt{1000}$ |
| `Hopper-v2` | 0 | 0.02 | 0.01 | 0.01 | 40 | 1 | 1 |
| `Swimmer-v2` | 0 | 1 | 0.5 | 0.1 | 40 | 1 | 10 |

Table 2: Hyperparameters for RL tasks.

## C BENCHMARK FUNCTIONS

Let $x \in \mathbb{R}^n$ be expressed in its coordinates as $x = (x_1, \cdots, x_n)$.

- `Sphere`: $x \mapsto x^{\mathrm{T}}x$
- `Rosenbrock`: $x \mapsto \sum_{i=1}^{n-1}(100(x_i^2 - x_{i+1})^2 + (1 - x_i)^2)$
- `Rastrigin`: $x \mapsto 10n + \sum_{i=1}^{n}(x_i^2 - 10\cos(2\pi x_i))$
- `Lunacek`: First define the constants

$$\mu_1 = 2.5, \quad s = 1 - \frac{1}{2\sqrt{n+20} - 8.2}, \quad d = 1, \quad \mu_2 = -\sqrt{\frac{\mu_1^2 - d}{s}}.$$

  Using these constants the function can be expressed as $x \mapsto \min\{\sum_{i=1}^{n}(x_i - \mu_1)^2, dn + s\sum_{i=1}^{n}(x_i - \mu_2)^2\} + 10\sum_{i=1}^{n}(1 - \cos(2\pi(x_i - \mu_1)))$.