Hard-Thresholding Meets Evolution Strategies in Reinforcement Learning

Chengqian Gao^{*1} , **William de Vazelhes**^{*1} , **Hualin Zhang**^{1} , **Bin Gu**^{1,2†} and **Zhiqiang Xu**^{1†}
^{1}Mohamed bin Zayed University of Artificial Intelligence, UAE
^{2}School of Artificial Intelligence, Jilin University, China
jsgubin@gmail.com, zhiqiang.xu@mbzuai.ac.ae

Abstract

Evolution Strategies (ES) have emerged as a competitive alternative for model-free reinforcement learning, showcasing exemplary performance in tasks like Mujoco and Atari. Notably, they shine in scenarios with imperfect reward functions, making them invaluable for real-world applications where dense reward signals may be elusive. Yet, an inherent assumption in ES—that all input features are task-relevant—poses challenges, especially when confronted with irrelevant features common in real-world problems. This work scrutinizes this limitation, particularly focusing on the Natural Evolution Strategies (NES) variant. We propose NESHT, a novel approach that integrates Hard-Thresholding (HT) with NES to champion sparsity, ensuring only pertinent features are employed. Backed by rigorous analysis and empirical tests, NESHT demonstrates its promise in mitigating the pitfalls of irrelevant features and shines in complex decision-making problems like noisy Mujoco and Atari tasks. Our code is available at https://github.com/cangcn/NES-HT.

1 Introduction

Evolution Strategies (ES) offer a compelling alternative for model-free reinforcement learning. Many studies proved the effectiveness of the ES algorithm in addressing complex decision-making problems, such as Mujoco and Atari tasks [Salimans *et al.*, 2017; Such *et al.*, 2017; Mania *et al.*, 2018], and, particularly, its remarkable proficiency in dealing with problems where imperfect reward functions are demanded such as sparse reward signals or delayed feedback [Salimans *et al.*, 2017; Majid *et al.*, 2021; François-Lavet *et al.*, 2015; Qian and Yu, 2021]. This capability is particularly appealing in real-world applications where acquiring dense reward signals may be expensive or unachievable.

However, the ES works under a potentially oversimplifying assumption, i.e., every input feature is inherently relevant to the task at hand, which could lead to poor performance in its application to real-world decision-making systems. For example, for an autonomous driving system, which receives pixel inputs from onboard cameras, while its main objective is to ensure safe navigation, the data stream might unintentionally include features irrelevant to driving decisions, such as the vehicle color. The inclusion of such irrelevant features in the learning process can not only result in unnecessarily large model sizes, but also lead to sub-optimal decisions. Although the detrimental effects of task-irrelevant features have been noticed across deep learning [Hubara *et al.*, 2017; Blalock *et al.*, 2020; Chen *et al.*, 2023], reinforcement learning [Sokar *et al.*, 2022; Grooten *et al.*, 2023a], and unsupervised learning [Li and Tang, 2015] research, it remains unclear how the noise features affect the performance of the ES algorithm.

This work aims to regularize the ES algorithm with sparsity, expecting that the obtained sparse policies can automatically select and utilize a small but necessary portion of available features. Specifically, we focus on the Natural Evolution Strategies (NES) [Wierstra *et al.*, 2014], a prevalent variant of the ES algorithm. NES estimates gradients, by only evaluating the objective function, for the objective optimization where irrelevant features can potentially drive the training process to end up with poor policies. To mitigate the impact of task-irrelevant features, we introduce the Hard-Thresholding (HT) operator [Blumensath and Davies, 2009] to the NES framework, given its popularity and simplicity for performing L_0 constrained optimization at a desired sparsity level. However, the HT was originally developed for optimization problems where the gradient comes with a closed-form expression [Garg and Khandekar, 2009; Nguyen *et al.*, 2017]. The compatibility and effectiveness of the HT operator with the NES estimate of the gradient remains unclear. To the best of our knowledge, this is the first time to propose sparsity-induced natural evolution strategies.

We begin by examining the negative effects of task-irrelevant observations on the NES. We find that the inclusion of such irrelevant features increases the randomness of the reward function, resulting in a higher variance of the estimated gradient. Consequently, it hinders convergence to the optimal policy. To address the issue of irrelevant features, we present the NESHT, which seamlessly integrates the HT operator into the NES algorithm. The *modus operandi* of the NESHT is straightforward: the parameters are truncated to retain only a specified proportion, upon each gradient descent/ascent update. In addition, we provide a comprehensive analysis of the convergence and complexity of the NESHT, underpinning it with the canonical assumptions of sparse learning. This analytical deep dive effectively resolves the lingering uncertainty regarding the compatibility of natural gradients with the hard-thresholding operator. Further, an extensive empirical study verifies the effectiveness of the NESHT, particularly in challenging noisy Mujoco environments with sparse rewards and Atari environments with pixel inputs.

2 Preliminaries

Markov decision process. We are concerned with the reinforcement learning problem, where our objective is to optimize a policy, denoted by $\pi_{\boldsymbol{\theta}}$, parameterized by $\boldsymbol{\theta}$. This policy is defined over a Markov decision process represented by $M = \langle \mathcal{S}, \mathcal{A}, \mathcal{T}, d_0, \mathcal{R}, \gamma \rangle$. For each episode, the initial state $\boldsymbol{s}_0$ is sampled from the distribution d_0. At each time step t, given an observation $\boldsymbol{s_t} \in \mathcal{S}$, the policy determines an action $\boldsymbol{a_t} \in \mathcal{A}$, which then results in an immediate reward $r(\boldsymbol{s_t}, \boldsymbol{a_t}) \in \mathcal{R}$. Subsequently, the system transitions to a new observation $\boldsymbol{s_{t+1}}$ in accordance with the dynamics $\mathcal{T}$. The resulting trajectory can be presented as $\boldsymbol{\tau} = \{(\boldsymbol{s}_t, \boldsymbol{a}_t, r, \boldsymbol{s}_{t+1})\}$. In order to balance the trade-off between immediate rewards and long-term rewards, the discount factor γ is introduced.

3 Decision-Making with Irrelevant Features

We propose NESHT, equipping the Natural Evolution Strategies with the Hard-Thresholding operator, for handling task-irrelevant observations in decision-making problem.

3.1 The Objective Function

Our objective is to maximize the *fitness score* achieved by the policy while minimizing the *impact* induced from the task-irrelevant features. We hypothesize that employing a sparse policy can effectively manage these redundant observations.

Fitness score. The performance of the policy can be quantified by the *fitness function*, which is defined as the expected sum of rewards over its rollout trajectories:

$$F(\boldsymbol{\theta}) := \mathbb{E}_{\tau \sim d_0, \pi_{\boldsymbol{\theta}}, \mathcal{T}} f_\tau(\boldsymbol{\theta}), \text{ with } f_\tau(\boldsymbol{\theta}) := \sum_{t=0}^{|\tau|} r(\boldsymbol{s_t}, \boldsymbol{a_t}) \tag{1}$$

It's important to note that, in this context, the discount factor γ is set to 1. This is in contrast to traditional RL settings where it often assumes values such as 0.99 or 0.9. Another characteristic is that the fitness function can be discontinuous *w.r.t.* the policy parameters due to the randomness in environments and the complex reward function.

L_0-constraint optimization. We propose mitigating the impact of task-irrelevant features through a sparse policy, under the premise that sparsity can effectively filter out irrelevant information present in inputs. Formally, our objective is to improve a policy while also constraining its complexity, i.e., the L_0 constrained optimization, with $\|\cdot\|_0$ denotes the L_0 (pseudo-)norm (number of non-zero components of a vector):

$$\max_{\boldsymbol{\theta}} F(\boldsymbol{\theta}) \quad s.t. \quad \|\boldsymbol{\theta}\|_0 < k \tag{2}$$

Why L_0 constraint? In our context, where only a small subset of observations is task-relevant, irrelevant features can significantly degrade performance. L_0-constrained optimization directly enforces a constraint on the L_0 norm of the learned parameter vector, ensuring the sparsity of the resulting model, alluring for feature selection tasks. Unlike L_1-constrained optimization, which promotes sparsity but does not guarantee exact zero values, L_0-constrained optimization offers precise control over sparsity by allowing certain model parameters to be set exactly to zero. This capability not only enhances model interpretability but also makes it well-suited for our setting, i.e., decision-making with irrelevant observations.

3.2 Our Proposal: NESHT

We introduce NESHT, a solution for decision-making problems involving both task-relevant and irrelevant features. While NES and the Hard-Thresholding operator are not novel concepts individually, their compatibility when used together may raise questions. To be self-contained, we now provide brief descriptions of each.

NES. We employ the competitive NES algorithm, to optimize the policy, with the following gradient estimator:

$$\nabla_{\boldsymbol{\theta}} \mathbb{E}_{\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, I)} F(\boldsymbol{\theta} + \sigma \boldsymbol{\epsilon}) = \frac{1}{\sigma} \mathbb{E}_{\boldsymbol{\epsilon} \sim \mathcal{N}(0, I)} F(\boldsymbol{\theta} + \sigma \boldsymbol{\epsilon}) \boldsymbol{\epsilon} \tag{3}$$

In NES, the gradient is approximated through sampling and serves as an approximation, bypassing challenges with non-differentiable functions or exploding gradients. For the derivation about Equation (3), please refer to Appendix.

Hard-thresholding operator. To achieve the L_0-constrained optimization described as Equation (2), we introduce the hard-thresholding operator into NES. It truncates the parameter vector, retaining only k components with the most significant absolute magnitudes, represented as $\mathrm{trunc}(\boldsymbol{\theta}, k)$, or, more succinctly, as $\mathrm{trunc}(\boldsymbol{\theta})$. While incorporating HT into NES is straightforward, the compatibility between HT and NES remains an open question.

Compatibility concerns. To establish the convergence of NESHT, it is essential to demonstrate the convergence of the hard-thresholding algorithm for non-convex and discontinuous F, with a gradient estimated as in (3) via the NES algorithm. In the literature, [Xu *et al.*, 2019] proved the convergence of stochastic algorithms in the case of non-convex objective functions F, for a non-convex proximal term which can be taken as the indicator function of the set of all k-sparse vectors (i.e. the L_0 pseudo-ball). This proof of convergence applies to stochastic hard-thresholding algorithms. However, their analysis assumes Lipschitz-smoothness of F and considers a general stochastic estimator of the gradient. Therefore, it does not account for the specific errors introduced by the gradient estimator from (3). More recently, the work of [Metel, 2023], analyzes the convergence of zeroth-order methods (similar to evolutionary strategies) for a Lipschitz-continuous and non-convex function F. However, in our case, F is discontinuous in general. Thus, to the best of our knowledge, the convergence of evolutionary strategies in such setting remains an open question. In the next section, we address this question by demonstrating that, under mild assumptions, proper convergence of Algorithm 1 is guaranteed.

Algorithm 1 NES with Hard-Thresholding

```
Input:
$\alpha$ - Learning rate,
$\boldsymbol{\theta_0}$- Initial policy parameters in $\mathbb{R}^d$,
$n$ - Population size,
$N$ - Number of rollouts collected for each agent,
$\sigma$ - Noise standard deviation,
$k$ - Number of parameters to be kept.
for $t = 0, 1, 2, ...T - 1$ do
    for $i = 1, ..., n$ do
        Sample a Gaussian perturbation $\boldsymbol{\epsilon}_i \sim \mathcal{N}(0, I)$ .
        for $j = 1, ..., N$ do
            Sample a rollout $\tau_j^{\boldsymbol{\epsilon}_i}$
            Compute returns $f_{\tau_j^{\boldsymbol{\epsilon}_i}}(\boldsymbol{\theta}_t + \sigma\boldsymbol{\epsilon_i})$
        end for
    end for
    Set $\boldsymbol{\theta}_{t+\frac{1}{2}} \leftarrow \boldsymbol{\theta}_t + \frac{\alpha}{nN\sigma}\sum_{i=1}^n\sum_{j=1}^N f_{\tau_j^{\boldsymbol{\epsilon}_i}}(\boldsymbol{\theta}_t + \sigma\boldsymbol{\epsilon_i})\boldsymbol{\epsilon}_i$
    Truncate the parameters: $\boldsymbol{\theta}_{t+1} \leftarrow \mathrm{trunc}(\boldsymbol{\theta}_{t+\frac{1}{2}}, k)$
end for
```

4 Convergence Analysis

The integration of NES with HT is detailed in Algorithm 1, where the hard-thresholding operator is applied to the learned parameters after each update. In this section, we provide a proof of convergence for NES combined with Hard-Thresholding, i.e., our NESHT, addressing the compatibility concern. Additionally, we would like to highlight that our analysis can also cover the case where no hard-thresholding operator is used (it only suffices to take the proximal term r in our proof of Theorem 1 in Appendix to be the constant zero): to our knowledge, such a proof of convergence for NES for general discontinuous functions F (which correspond to a realistic reinforcement learning setting) is the first in the literature, and we hope that such a result, as well as the subsequent remarks and discussions on the influence of each parameter on the convergence rate (bound on the expected reward B, dimension d, etc.) can be of interest to the NES community.

4.1 Assumptions

To proceed with the proof of convergence of NESHT, we will need the following assumptions below.

Assumption 1 (Boundedness of F)**.** *The fitness function F is bounded on its domain, that is, there exists a universal constant $B > 0$ such that:*

$$\forall \boldsymbol{\theta} \in \mathbb{R}^d : |F(\boldsymbol{\theta})| \leq B$$

Remark 1. *$F(\boldsymbol{\theta})$ represents the expected rewards obtained by executing policy $\pi_{\boldsymbol{\theta}}$. The boundedness assumption is typically reasonable since immediate rewards do not tend to infinity, and evaluation trajectories always have finite lengths. Importantly, this assumption remains valid even when dealing with task-irrelevant features.*

Additionally, we will need the following assumption on the variance of the cumulative reward, for a given parameter vector $\boldsymbol{\theta}$.

Assumption 2 (Bounded variance of f_τ)**.** *We posit the existence of a universal constant $C > 0$ such that the variance of the cumulative reward for any $\boldsymbol{\theta} \in \{\boldsymbol{\theta}_0, \boldsymbol{\theta}_{\frac{1}{2}}, ..., \boldsymbol{\theta}_{T-\frac{1}{2}}, \boldsymbol{\theta}_T\}$ is bounded by C, i.e.:*

$$\mathbb{E}_\tau\left[|f_\tau(\boldsymbol{\theta}) - F(\boldsymbol{\theta})|^2\right] \leq C.$$

Remark 2. *Assumption 2 reflects the inherent randomness from both the policy, whether it is deterministic or stochastic, and the environment, which introduces randomness through factors such as the dynamics $\mathcal{T}$, the reward function $r(s, a)$, and the initial distribution of states d_0. Also, please note that if the reward and the episode length are limited, as is usually the case in RL, then Assumptions 1 and 2 are satisfied. An observant reader may notice that the inclusion of task-irrelevant features unavoidably leads to an increase in the constant C due to the introduction of randomness. As we will see later, this increase hampers the convergence of NES algorithms.*

4.2 Smoothness

Since F can be discontinuous in general, maximizing F directly is impossible with evolutionary strategies. For instance if F is Dirac-like, such as $F(\boldsymbol{\theta}) = \begin{cases} 1 \text{ if } \boldsymbol{\theta} = \boldsymbol{0} \\ 0 \text{ otherwise} \end{cases}$, the probability (for a given $\boldsymbol{\theta}$), to successfully sample an $\boldsymbol{\epsilon}$ such that $F(\boldsymbol{\theta} + \sigma\boldsymbol{\epsilon}) = 1$ is actually zero, which means the parameters will be updated with probability zero. However, we can instead analyze the convergence of a smoothed version of F, F_σ, defined below:

$$F_\sigma(\boldsymbol{\theta}) := \mathbb{E}_{\boldsymbol{\epsilon}\sim\mathcal{N}(0,I)}F(\boldsymbol{\theta} + \sigma\boldsymbol{\epsilon})$$

Note that F_σ converges towards F for small σ in terms of *eh-convergence*, as described in Theorem 3.2 from [Yu *et al.*, 1992]. The first step, to derive the convergence rate of our algorithm with F_σ, is to prove that F_σ is smooth, and to derive its smoothness constant, which we then use in a proof framework similar to [Xu *et al.*, 2019].

Lemma 1. *Under Assumption 1, F_σ is Lipschitz-smooth (i.e. its gradient is Lipschitz-continuous), with a smoothness constant $L = \frac{(d+1)B}{\sigma^2}$, that is, such L verifies:*

$$\forall \boldsymbol{\theta}_1, \boldsymbol{\theta}_2 \in (\mathbb{R}^d)^2 : \|\nabla F_\sigma(\boldsymbol{\theta}_1) - \nabla F_\sigma(\boldsymbol{\theta}_2)\| \leq L\|\boldsymbol{\theta}_1 - \boldsymbol{\theta}_2\|$$

Proof. Proof in Appendix. □

For discontinuous functions F, the fact that F_σ is smooth was already known before in the literature (see e.g. [Ermoliev and Norkin, 1995]). However, such works did not provide an explicit formula for the smoothness constant L. Here, for the first time in the literature (to the best of our knowledge), using the boundedness assumption on F, we could derive an explicit formula for the smoothness constant L.

One can therefore see that L is proportional to both the bound of the fitness function, B, and the dimension of the policy parameters, d, while being inversely proportional to the variance σ^2. In Section 4.4, we will observe the role of such smoothness constant L: the smaller it is, the faster the NES algorithm will converge.

4.3 Error of the Gradient Estimator

We now consider the gradient estimator with a general population of n random perturbations, and a number of rollouts of N for each perturbation. More precisely, assume that we sample n random directions $\{\boldsymbol{\epsilon}_i\}_{i=1}^n := \{\boldsymbol{\epsilon}_1, ..., \boldsymbol{\epsilon}_n\}$ independently and identically distributed, and that for each of these random directions $\boldsymbol{\epsilon}_i$, we sample N rollouts $\{\tau_j^{\boldsymbol{\epsilon}_i}\}_{j=1}^N := \{\tau_1^{\boldsymbol{\epsilon}_i}, .., \tau_N^{\boldsymbol{\epsilon}_i}\}$ independently and identically distributed, to obtain a final collection of rollouts $\{\{\tau_j^{\boldsymbol{\epsilon}_i}\}_{j=1}^N\}_{i=1}^n$, and to get $N \times n$ gradient estimators $\hat{g}_{\sigma,\boldsymbol{\epsilon}_i,\tau_j^{\boldsymbol{\epsilon}_i}}, (i, j) \in [n] \times [N]$ defined below:

$$\hat{g}_{\sigma,\boldsymbol{\epsilon}_i,\tau_j^{\boldsymbol{\epsilon}_i}}(\boldsymbol{\theta}) := \frac{1}{\sigma} f_{\tau_j^{\boldsymbol{\epsilon}_i}}(\boldsymbol{\theta} + \sigma\boldsymbol{\epsilon}_i)\boldsymbol{\epsilon}_i$$

which we aggregate in the following estimator:

$$\bar{g}_{\sigma,\{\boldsymbol{\epsilon}_i\}_{i=1}^n,\{\{\tau_j^{\boldsymbol{\epsilon}_i}\}_{j=1}^N\}_{i=1}^n}(\boldsymbol{\theta}) := \frac{1}{nN}\sum_{i=1}^{n}\sum_{j=1}^{N}\hat{g}_{\sigma,\boldsymbol{\epsilon}_i,\tau_j^{\boldsymbol{\epsilon}_i}}(\boldsymbol{\theta})$$

Lemma 2. *Under Assumptions 1 and 2, the estimator above is an unbiased estimate of the gradient of the smoothed function F, and its variance is bounded, more precisely, for any $\boldsymbol{\theta} \in \{\boldsymbol{\theta}_0, \boldsymbol{\theta}_{\frac{1}{2}}, ..., \boldsymbol{\theta}_{T-\frac{1}{2}}, \boldsymbol{\theta}_T\}$:*

$$\mathbb{E}\bar{g}_{\sigma,\{\boldsymbol{\epsilon}_i\}_{i=1}^n,\{\{\tau_j^{\boldsymbol{\epsilon}_i}\}_{j=1}^N\}_{i=1}^n}(\boldsymbol{\theta}) = \nabla_{\boldsymbol{\theta}}F_\sigma(\boldsymbol{\theta})$$

$$\mathbb{E}\|\bar{g}_{\sigma,\{\boldsymbol{\epsilon}_i\}_{i=1}^n,\{\{\tau_j^{\boldsymbol{\epsilon}_i}\}_{j=1}^N\}_{i=1}^n}(\boldsymbol{\theta}) - \nabla_{\boldsymbol{\theta}}F_\sigma(\boldsymbol{\theta})\|^2 \leq \frac{Cd}{N\sigma^2} + \frac{dB^2}{n\sigma^2}$$

Proof. See Appendix. We begin by examining the unbiasedness (using a standard proof) and variance (using a novel proof up to our knowledge) of the gradient estimator for a single perturbation, i.e., $\hat{g}_{\sigma,\boldsymbol{\epsilon}_i,\tau_j^{\boldsymbol{\epsilon}_i}}$. We then generalize our results to account for multiple perturbations (n) and rollouts (N). □

Advantages of NESHT: reduction in constant C. We present here a formal explanation for the superiority of NESHT over NES in the lens of constant C. Thanks to hard-thresholding, along training, $\boldsymbol{\theta}_t$ and $\boldsymbol{\theta}_{t+\frac{1}{2}}$ remain in the space of k-sparse vectors (up to small perturbations $\sigma\boldsymbol{\epsilon}$), whereas they could live anywhere in $\mathbb{R}^d$ in the case of NES. Based on the hypothesis that the hard-thresholding operation effectively selects relevant features (which we have verified experimentally in Section 5.2), NESHT can successfully mitigate the impact of irrelevant features and reduces the value of C. To illustrate this, one can consider the following scenario.

Example 1. *Consider a one-step decision-making experiment, with linear policy, and fitness score given as: $f_\tau(\boldsymbol{\theta}) := \boldsymbol{x}^\top(\boldsymbol{\theta} - \boldsymbol{\theta}^*)$, where $\boldsymbol{\theta}^*$ is a k-sparse vector, with $S \subseteq [d]$ being the set of coordinates of its non-zero components, i.e., the relevant features. In addition, $\boldsymbol{x}$ is the input state, which we assume follows a normal distribution $\mathcal{N}(\mathbf{0}, \sigma\boldsymbol{I}_{d\times d})$ for $\sigma > 0$ ($\boldsymbol{I}_{d\times d}$ denoting the identity). We then have, for any bounded policy $\boldsymbol{\theta} \in [-1, 1]^d$:*

$$\begin{aligned}\mathbb{E}_{\boldsymbol{x}}|f_\tau(\boldsymbol{\theta}) - F(\boldsymbol{\theta})|^2 &= \mathbb{E}_{\boldsymbol{x}}(\boldsymbol{\theta} - \boldsymbol{\theta}^*)^\top\boldsymbol{x}\boldsymbol{x}^\top(\boldsymbol{\theta} - \boldsymbol{\theta}^*) \\ &= (\boldsymbol{\theta} - \boldsymbol{\theta}^*)^\top\sigma^2\boldsymbol{I}_{d\times d}(\boldsymbol{\theta} - \boldsymbol{\theta}^*) = \sigma^2\|\boldsymbol{\theta} - \boldsymbol{\theta}^*\|^2\end{aligned}$$

Therefore, if there are many irrelevant components present (i.e. $|[d] \setminus S|$ is large), the episode-wise variance of f_τ (and its bound C) will be higher when $\boldsymbol{\theta}$ is dense (proportionally to σ^2). As established in Lemma 2, the proper convergence of NESHT depends on this variance. The application of a hard-thresholding operator explicitly filters out some of the noisy features, introducing a bias that steers the policy towards making decisions exclusively based on sparse observations. This reduces the variance and ensures better convergence to the optimal policy.

In practical terms, given a fixed interaction budget for n and N, the variance of the gradient estimator may be too high for vanilla NES, causing it to fail to converge to the optimal policy. However, with the reduced variance of the gradient estimator in NESHT, as described above, convergence of the parameters $\boldsymbol{\theta}$ to a stationary point of the fitness function F can be successfully ensured, as stated in Theorem 1. Section 5 provides illustrations of cases where NES fails to converge to a successful policy, but NESHT can learn a successful policy in several RL tasks. This validates our hypothesis that learning a sparse policy with NESHT can properly handle irrelevant noise in the observations.

4.4 Convergence Rate

Equipped with Lemmas 1 and 2, we can now prove the convergence of Algorithm 1, following for the most part the framework of [Xu *et al.*, 2019] for stochastic gradient descent with a non-convex function and a non-convex non-smooth proximal term, but plugging into it our novel bounds for (i) the smoothness constant of F_σ and (ii) the variance of the gradient estimator $\bar{g}_{\sigma,\{\boldsymbol{\epsilon}_i\}_{i=1}^n,\{\{\tau_j^{\boldsymbol{\epsilon}_i}\}_{j=1}^N\}_{i=1}^n}(\boldsymbol{\theta})$, under our specific assumption of boundedness of F. Because of such non-convex and non-smooth optimization problem, convergence is proven in terms of the expected distance of the Fréchet sub-differential $\hat{\partial}(-F_\sigma(\boldsymbol{\theta}) + \mathbb{1}_{L_0(k)}(\boldsymbol{\theta}_T))$ to zero [Rockafellar, 1976], where $\mathbb{1}_{L_0(k)}$ denotes the indicator function of the L_0 constraint, i.e. $\mathbb{1}_{L_0(k)}(\boldsymbol{\theta}) = \begin{cases} 0 \text{ if } \boldsymbol{\theta} \text{ is } k\text{-sparse} \\ +\infty \text{ otherwise}\end{cases}$. Note that this is the standard way to define stationary points for non-smooth regularizers (such as sparsity constraints) (see e.g. Thm. 2 in [Xu *et al.*, 2019] or Thm. 3 in [Deleu and Bengio, 2021]).

Theorem 1. *Under Assumption 1 and 2, run Algorithm 1, with $\alpha = \frac{c}{L}\left(0 < c < \frac{1}{2}\right)$, a number of iterations $T = 2c_2B/\left(\alpha\varepsilon^2\right)$ and $N \geq \frac{4c_1dC}{\sigma^2\varepsilon^2}$ and $n \geq \frac{4c_1dB^2}{\sigma^2\varepsilon^2}$ for $t = 0, \ldots, T-1$, then the output $\boldsymbol{\theta}_T$ of Algorithm 1 satisfies*

$$\mathbb{E}\left[\operatorname{dist}\left(\mathbf{0}, \hat{\partial}\left(-F_\sigma\left(\boldsymbol{\theta}_T\right) + \mathbb{1}_{L_0(k)}(\boldsymbol{\theta}_T)\right)\right)\right] \leq \varepsilon,$$

where $c_1 = \frac{2c(1-2c)+2}{c(1-2c)}$, and $c_2 = \frac{12-8c}{1-2c}$, and where $\operatorname{dist}(\boldsymbol{z}, S)$ is the distance of a set S to a point $\boldsymbol{z}$, defined as the minimal Euclidean distance of any point in S to $\boldsymbol{z}$. In particular in order to have $\mathbb{E}\left[\operatorname{dist}\left(0, \hat{\partial}(-F_\sigma(\boldsymbol{\theta}) + \mathbb{1}_{L_0(k)}(\boldsymbol{\theta}_T))\right)\right] \leq \varepsilon$, that is, in order to ensure convergence to a stationary point, it suffices to set $T = O\left(1/\varepsilon^2\right)$.

Proof. Proof in Appendix. □

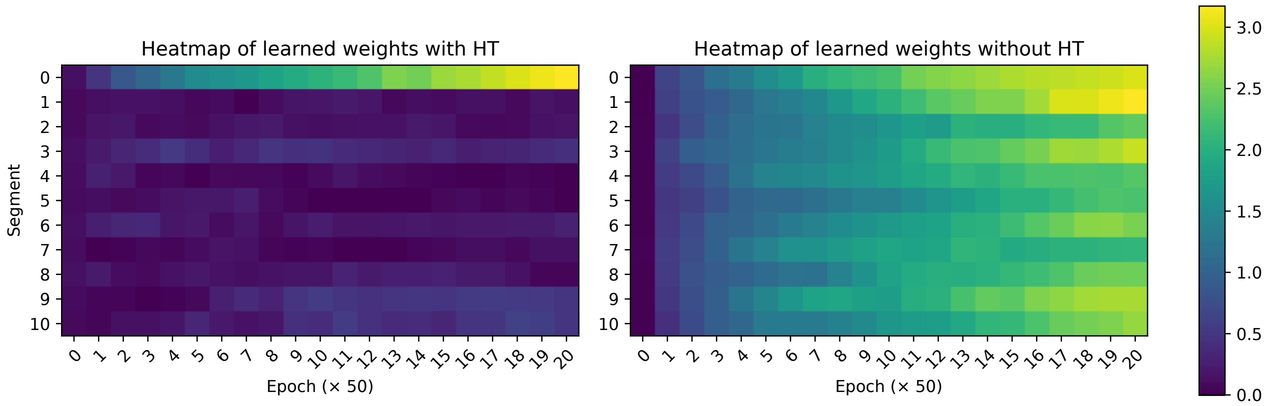

Figure 1: Heatmap illustrating the evolution of learned weights from a NESHT policy (left) and a NES policy (right) over epochs. The environment studied is Hopper-V3, with tenfold Gaussian noise. Among the 11 distinct observation segments (Y-axis), only the first (0-th) segment corresponds to the environment-provided features, while all subsequent 10 segments represent Gaussian noise (task-irrelevant features). The heatmap color indicates the norm of the learned weights. With the HT operator, only the portion of the neuron corresponding to task-relevant features (the 0-th segment) is activated. Without HT, NES struggles with task-irrelevant features, leading to poor performance.

Remark 3. *As per Theorem 1, we can see that a large smoothing radius σ will ease convergence, as it allows one to evaluate fewer random perturbations and rollouts. However, the counterpart is that the function optimized F_σ may be further away from the true function F.*

Remark 4 (Overall complexity)**.** *From Theorem 1, to ensure convergence to a stationary point up to tolerance ε, we need to take $N = O(\frac{dC}{\sigma^2\varepsilon^2})$, $n = O(\frac{dB^2}{\sigma^2\varepsilon^2})$, and $T = O(\frac{B}{\alpha\varepsilon^2}) \overset{(a)}{=} O(\frac{BL}{\varepsilon^2}) \overset{(b)}{=} O(\frac{B^2 d}{\varepsilon^2\sigma^2})$, where (a) follows from the definition of α from Theorem 1, and (b) follows from Lemma 1. Therefore, the overall number of episodes needed to ensure convergence is $TNn = O(\frac{d^3B^4C}{\sigma^6\varepsilon^6})$. Note however that if one has access to a massively parallel device able to run in parallel Nn simulations, which is very common in RL settings (e.g. as in [Salimans* et al.*, 2017]), the time complexity of the whole optimization process is simply $T = O(\frac{B^2 d}{\varepsilon^2\sigma^2})$.*

5 Experiment

The design of NESHT is based on two central premises: (1) weights and biases corresponding to task-irrelevant features can be set to zero by the hard-thresholding operator, and (2) the hard-thresholding operator is compatible with the NES algorithm. In this section, we design experiments to address the following questions:

1) Does HT truly capture task-irrelevant observations?
2) Can the NESHT policy outperform other solutions?
3) Can the effect of HT be extended to visual tasks?
4) How does the HT ratio affect performance?

5.1 Experimental Setups

We perform evaluations on two popular RL protocols, Mujoco [Todorov *et al.*, 2012] and Atari [Bellemare *et al.*, 2013] environments.

Mujoco setups. Observations in the Mujoco continuous environment are represented as floating-point values. Detecting redundancy features in these observations is challenging due to the complexity of the environment dynamics. To simulate decision-making in the presence of task-irrelevant features, we *concatenate* Gaussian noise with the environment-provided observations. Additionally, we set 90% of the immediate rewards to zero, replicating a more challenging real-world scenario characterized by an imperfect reward function. In the analysis section, we use the notation k to represent the number of parameters to be retained. In the experiment section and in our implementation, we prefer β to denote the hard-thresholding ratio, which refers to the ratio of activated neurons.

Protocol for Mujoco tasks. We primarily base our implementation on the framework outlined in [Salimans *et al.*, 2017]. In this context, we use i.i.d. Gaussian perturbations in the parameter space to estimate the gradient. As a result, the natural gradient simplifies to the plain gradient, as shown in Equation 3. It is worth noting that we choose a linear policy for NESHT since it is easier to train and has been shown to be expressive enough for such tasks [Mania *et al.*, 2018].

Atari. Beside the Mujoco benchmarks, we include the more challenging Atari games with pixel inputs to answer the question of whether the hard-thresholding algorithm can handle irrelevant features in visual observations. We use the full screen of the Atari game as input (110x84 pixels). This includes not only the playing area, but also other task-irrelevant features, such as the scoreboard and backgrounds. Notably, we employ a CNN module for extracting latent features from the pixel inputs for Atari Games only, alone with a linear layer mapping them to the action space. A prevalent challenge associated with NES is their sample efficiency. In Atari experiments, we mirror the training configuration in [Salimans *et al.*, 2017]. Specifically, we train the policy for a duration of 1 hour using a 500-core machine. Furthermore, we set an upper limit on the interaction budget at 10M steps.

Env Name	Noise Ratio	TRPO	DDPG	TD3	Vanilla NES	ANF-SAC	NES-L_1	NES+HT(Ours)
HalfCheetah	$\times5$	198.3	1369.8	665.7	819.2	28.8	1152.7	**1851.8**
	$\times10$	31.6	1285.2	197.7	805.9	8.5	918.4	**1722.5**
	$\times20$	19.2	843.4	198.5	773.4	8.4	669.1	**1213.9**
Hopper	$\times5$	266.9	917.6	972.5	803.7	1014.3	679.4	**1354.5**
	$\times10$	43.7	824.3	918.0	241.2	1010.7	204.5	**1187.1**
	$\times20$	23.0	809.5	**991.2**	147.7	1015.4	62.9	**1006.6**
Walker2d	$\times5$	441.9	1030.1	1000.7	784.0	986.5	745.2	**1043.8**
	$\times10$	420.3	907.3	559.7	384.7	**966.6**	364.2	940.3
	$\times20$	229.9	780.6	485.8	240.6	**960.2**	33.8	675.2

Table 1: Comparison on the performance of RL and ES algorithms on Mujoco locomotion tasks with varying levels of noise. The term **Noise Ratio** indicates the amount of redundancy observations (Gaussian noise). For example, $\times5$ signifies that the dimension of Gaussian noises is five times that of the environment-provided observations. For RL algorithms, we train for 1 million environment steps while 10 million steps for ES approaches. We report the average scores received by last 10 evaluations across 20 random seeds. Notably, the results for the NESHT algorithm are from runs with a fixed hard-thresholding ratio, $\beta = 0.9$.

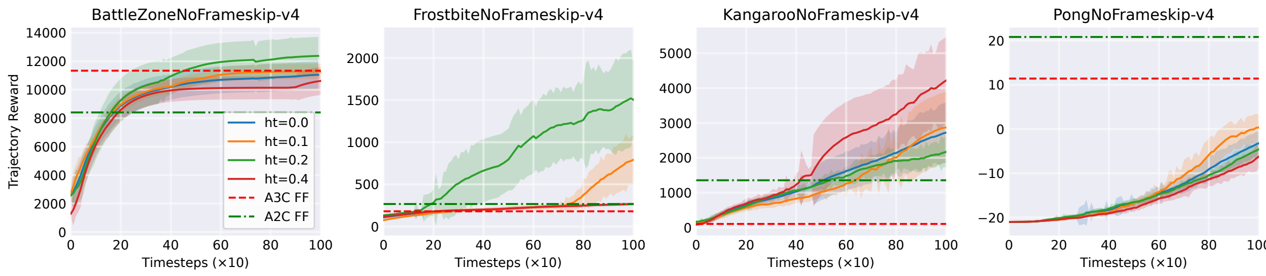

Figure 2: Comparison of NES in four representative Atari tasks with and without HT. We report the outcomes of NESHT under varying hard-thresholding ratios, based on 20 random seeds. Results from the A3C and A2C algorithms are adapted from [Salimans *et al.*, 2017].

5.2 Efficacy of the Hard-Thresholding Operator

We begin by evaluating the effectiveness of the hard-thresholding operator in identifying and truncating task-irrelevant features. To do this, we devise a toy example using the Mujoco Hopper-V3 environment with added noise. Specifically, the environment-provided 11-dimension features are augmented with 110-dimension i.i.d. Gaussian noise. We set the hard-thresholding ratio to 0.9, ensuring that only the top 10% of the large learned weights are retained.

The main focus of our analysis is to examine the norm of the learned weights across each *segment* of observations. As shown in Figure 1, the first segment (index 0) relates to the features provided by the environment. This visualization showcases the weight norms (L_1 norm) for the genuine features (0-th segment) and the task-irrelevant features (remaining 10 segments). By employing the hard-thresholding operation iteratively, the learned weights for relevant features remain large, while those for irrelevant features are truncated. By comparing results with and without the use of the hard-thresholding operator in Figure 1, we offer empirical evidence of its utility, i.e., sparse policy learned from the NESHT can select relevant features. Notably, it's essential to apply hard-thresholding iteratively during each update rather than just once post-learning.

5.3 Performance on Noised Mujoco Tasks

As illustrated in the preceding section, the hard-thresholding operator can successfully assist the NES algorithm in capturing task-relevant features. In this section, we evaluate the performance of the proposed NESHT algorithm in comparison to the original NES algorithm, as well as other commonly employed methods for decision-making tasks, such as RL algorithms.

Baselines. Experiments are conducted in three popular Mujoco environments: Hopper, Walker2d, and HalfCheetah. We introduce Gaussian noise ranging from 5-fold to 20-fold, which is merged with the environment-provided features. Our baseline algorithms fall into three categories:

- Vanilla NES policy: We follow the implementation in [Salimans *et al.*, 2017], but with a modification: the agent is instantiated with a one-layer linear network.
- Classic RL algorithms: Since ES algorithms are often viewed as alternatives to RL, we include algorithms like TRPO [Schulman *et al.*, 2015], DDPG [Lillicrap *et al.*, 2015], and TD3 [Fujimoto *et al.*, 2018] in our baselines.
- Other Solutions: One effective way to address irrelevant features is by incorporating an L_1-norm penalty. We explored this approach with the NES algorithm. A complexity penalty, the L_1-norm of the learned weights, is subtracted from the fitness score, and we refer to this as NES-L_1. Additionally, we include a solution in the literature of RL, ANF-SAC [Grooten *et al.*, 2023b]. This baseline algorithm is designed for RL with very dense rewards, while ours can handle the sparse reward signal.

It's worth noting that while NES, NESHT and NES-L_1 agents utilize one-layer linear networks, all other baseline

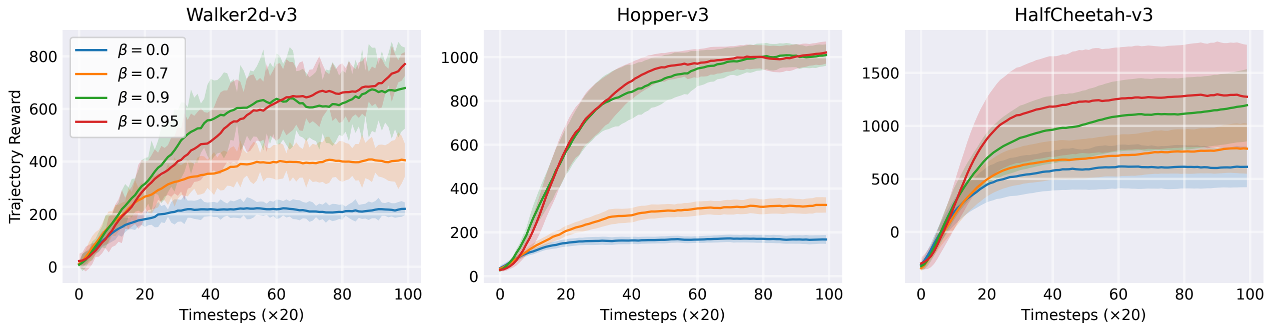

Figure 3: Ablation study. We assess the impact of hard-thresholding operation in the presence of Gaussian noise with a 20× noise dimension on the Mujoco environment. We vary the value of β from 0.0 (corresponding to Vanilla NES) to 0.95 while retaining only 5% of the neurons.

algorithms for comparison are based on three-layer non-linear networks. We report the average scores across the last ten evaluations. Notably, we find ANF-SAC shows a failure after achieving impressive performance in HalfCheetah tasks.

Results. The results of our experiments are presented in Table 1. Our findings highlight the viability of NES as an alternative to RL approaches. Specifically, the performance of the NES agent consistently aligns with the state-of-the-art DDPG and TD3 methods. However, the NES policy shows dramatic performance degradation when confronted with increasing task-irrelevant observations. This trend highlights a harmful assumption within NES algorithms, which assumes that all features are task-relevant. Fortunately, the introduction of the hard-thresholding operator successfully mitigates this performance drop. Compared to vanilla NES, NESHT not only demonstrates enhanced resilience against irrelevant observations, but also consistently outperforms the RL baselines, emphasizing its effectiveness.

5.4 Comparison on Visual Atari Benchmarks

We expand our comparison to include Atari environments that involve visual inputs. This extension is motivated by concerns raised about the presence of artificial noises in previous evaluations. In Atari games, task-irrelevant features are from the environment and vary significantly. For instance, elements such as scoreboards in each game (with different locations) are considered important yet distracting observations that can hinder performance, as noted in [Mnih *et al.*, 2015].

We present the results on four representative Atari games in Figure 2. Remarkably, we observed a significant improvement in the performance of NES algorithms when employing the hard-thresholding operator. It is important to note that the optimal hard-thresholding ratio, denoted as β, differs across these environments. We will discuss its influence in the next part. It is worth mentioning that this investigation does not include a comprehensive review of solutions in the RL or ES literature. Our main focus here is not to determine whether NES is a competitive solution (as already demonstrated by [Salimans *et al.*, 2017]), nor to claim that NESHT is the ultimate solution for Atari-type tasks. Instead, our goal is to explore the impact of the hard-thresholding operator on NES performance in decision-making tasks with pixel inputs.

5.5 Hyper-Parameter Study

The NESHT algorithm, proposed in this work, introduces a critical hyperparameter known as the hard-thresholding ratio, denoted as β, which plays a fundamental role in controlling the activation of neurons. In this section, we assess its impact. We employ the noised Mujoco locomotion tasks, where we can control the amount of task-irrelevant features. Specifically, we use three environments with $\times 20$ Gaussian noises, where the noise dimension is 20 times that of the environment-provided observations. Results depicted in Figure 3 illustrate the effectiveness of hard-thresholding in mitigating the impact of noisy observations.The vanilla NES algorithm ($\beta = 0.0$) struggles to learn from the noisy observations for policy improvement. However, as we increase β, truncating more small weights, we observe a significant performance improvement. This suggests that the hard-thresholding operation effectively filters out noise and enhances the algorithm's ability to learn from challenging, noisy environments.

6 Conclusion

Hard-thresholding emerges as a promising solution for L_0-constrained optimization, offering a solution to address task-irrelevant features frequently encountered in real-world scenarios. Yet, the compatibility between HT and NES gradients remains an area of active inquiry, rendering the practical implementation of such algorithms a subject of caution. In this study, we provide a theoretical foundation that establishes the convergence of the NES gradient descent (ascent) when paired with the HT operator, thus bolstering the credibility of our NESHT algorithm. Empirical assessments conducted across both Mujoco and Atari environments further substantiate the efficacy of our proposed method.

Contribution Statement

CG and **WV** contributed equally to this work. CG organized the experiment and wrote all sections except the theoretical analysis parts. WV proved the theoretical properties of the algorithm and wrote the relevant proofs. **HZ** reviewed the proofs and offered valuable insights during the writing process. **BG** and **ZX** initiated the discussion on the use of hard-thresholding in the NES algorithm. They supervised the research, proposed ideas, and assisted in writing the paper.

References

[Bellemare *et al.*, 2013] Marc G Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47:253–279, 2013.

[Blalock *et al.*, 2020] Davis W. Blalock, Jose Javier Gonzalez Ortiz, Jonathan Frankle, and John V. Guttag. What is the state of neural network pruning? In *Proceedings of Machine Learning and Systems 2020, MLSys 2020, Austin, TX, USA, March 2-4, 2020*. mlsys.org, 2020.

[Blumensath and Davies, 2009] Thomas Blumensath and Mike E Davies. Iterative hard thresholding for compressed sensing. *Applied and computational harmonic analysis*, 27(3):265–274, 2009.

[Chen *et al.*, 2023] Jou-An Chen, Wei Niu, Bin Ren, Yanzhi Wang, and Xipeng Shen. Survey: Exploiting data redundancy for optimization of deep learning. *ACM Comput. Surv.*, 55(10):212:1–212:38, 2023.

[Deleu and Bengio, 2021] Tristan Deleu and Yoshua Bengio. Structured sparsity inducing adaptive optimizers for deep learning. *arXiv preprint arXiv:2102.03869*, 2021.

[Ermoliev and Norkin, 1995] Yuri M Ermoliev and Vladimir Ivanovich Norkin. On nonsmooth problems of stochastic systems optimization. 1995.

[François-Lavet *et al.*, 2015] Vincent François-Lavet, Raphaël Fonteneau, and Damien Ernst. How to discount deep reinforcement learning: Towards new dynamic strategies. *CoRR*, abs/1512.02011, 2015.

[Fujimoto *et al.*, 2018] Scott Fujimoto, Herke Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *International conference on machine learning*, pages 1587–1596. PMLR, 2018.

[Garg and Khandekar, 2009] Rahul Garg and Rohit Khandekar. Gradient descent with sparsification: an iterative algorithm for sparse recovery with restricted isometry property. In *Proceedings of the 26th Annual International Conference on Machine Learning, ICML 2009, Montreal, Quebec, Canada, June 14-18, 2009*, volume 382 of *ACM International Conference Proceeding Series*, pages 337–344. ACM, 2009.

[Grooten *et al.*, 2023a] Bram Grooten, Ghada Sokar, Shibhansh Dohare, Elena Mocanu, Matthew E. Taylor, Mykola Pechenizkiy, and Decebal Constantin Mocanu. Automatic noise filtering with dynamic sparse training in deep reinforcement learning. In *Proceedings of the 2023 International Conference on Autonomous Agents and Multiagent Systems, AAMAS 2023, London, United Kingdom, 29 May 2023 - 2 June 2023*, pages 1932–1941. ACM, 2023.

[Grooten *et al.*, 2023b] Bram Grooten, Ghada Sokar, Shibhansh Dohare, Elena Mocanu, Matthew E. Taylor, Mykola Pechenizkiy, and Decebal Constantin Mocanu. Automatic noise filtering with dynamic sparse training in deep reinforcement learning. *The 22nd International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 2023. URL: https://arxiv.org/abs/2302.06548.

[Hubara *et al.*, 2017] Itay Hubara, Matthieu Courbariaux, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Quantized neural networks: Training neural networks with low precision weights and activations. *J. Mach. Learn. Res.*, 18:187:1–187:30, 2017.

[Li and Tang, 2015] Zechao Li and Jinhui Tang. Unsupervised feature selection via nonnegative spectral analysis and redundancy control. *IEEE Trans. Image Process.*, 24(12):5343–5355, 2015.

[Lillicrap *et al.*, 2015] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

[Majid *et al.*, 2021] Amjad Yousef Majid, Serge Saaybi, Tomas van Rietbergen, Vincent François-Lavet, R. Venkatesha Prasad, and Chris J. M. Verhoeven. Deep reinforcement learning versus evolution strategies: A comparative survey. *CoRR*, abs/2110.01411, 2021.

[Mania *et al.*, 2018] Horia Mania, Aurelia Guy, and Benjamin Recht. Simple random search of static linear policies is competitive for reinforcement learning. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pages 1805–1814, 2018.

[Metel, 2023] Michael R Metel. Sparse training with lipschitz continuous loss functions and a weighted group l_0-norm constraint. *Journal of Machine Learning Research*, 24(103):1–44, 2023.

[Mnih *et al.*, 2015] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin A. Riedmiller, Andreas Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nat.*, 518(7540):529–533, 2015.

[Nguyen *et al.*, 2017] Nam Nguyen, Deanna Needell, and Tina Woolf. Linear convergence of stochastic iterative greedy algorithms with sparse constraints. *IEEE Trans. Inf. Theory*, 63(11):6869–6895, 2017.

[Qian and Yu, 2021] Hong Qian and Yang Yu. Derivative-free reinforcement learning: a review. *Frontiers Comput. Sci.*, 15(6):156336, 2021.

[Rockafellar, 1976] R Tyrrell Rockafellar. Monotone operators and the proximal point algorithm. *SIAM journal on control and optimization*, 14(5):877–898, 1976.

[Salimans *et al.*, 2017] Tim Salimans, Jonathan Ho, Xi Chen, and Ilya Sutskever. Evolution strategies as a scalable alternative to reinforcement learning. *CoRR*, abs/1703.03864, 2017.

[Schulman *et al.*, 2015] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International conference on machine learning*, pages 1889–1897. PMLR, 2015.

[Sokar *et al.*, 2022] Ghada Sokar, Elena Mocanu, Decebal Constantin Mocanu, Mykola Pechenizkiy, and Peter Stone. Dynamic sparse training for deep reinforcement learning. In *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI 2022, Vienna, Austria, 23-29 July 2022*, pages 3437–3443. ijcai.org, 2022.

[Such *et al.*, 2017] Felipe Petroski Such, Vashisht Madhavan, Edoardo Conti, Joel Lehman, Kenneth O. Stanley, and Jeff Clune. Deep neuroevolution: Genetic algorithms are a competitive alternative for training deep neural networks for reinforcement learning. *CoRR*, abs/1712.06567, 2017.

[Todorov *et al.*, 2012] Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ international conference on intelligent robots and systems*, pages 5026–5033. IEEE, 2012.

[Wierstra *et al.*, 2014] Daan Wierstra, Tom Schaul, Tobias Glasmachers, Yi Sun, Jan Peters, and Jürgen Schmidhuber. Natural evolution strategies. *J. Mach. Learn. Res.*, 15(1):949–980, 2014.

[Xu *et al.*, 2019] Yi Xu, Rong Jin, and Tianbao Yang. Non-asymptotic analysis of stochastic methods for non-smooth non-convex regularized problems. *Advances in Neural Information Processing Systems*, 32, 2019.

[Yu *et al.*, 1992] Ermoliev Yu, V Norkin, and R Wets. The minimization of discontinuous functions: Mollifier subgradients. Technical report, Working Paper, International Institute for Applied Systems Analysis . . . , 1992.