# Training Diverse High-Dimensional Controllers by Scaling Covariance Matrix Adaptation MAP-Annealing

Bryon Tjanaka, Matthew C. Fontaine, David H. Lee, Aniruddha Kalkar, and Stefanos Nikolaidis

***Abstract*—Pre-training a diverse set of neural network controllers in simulation has enabled robots to adapt online to damage in robot locomotion tasks. However, finding diverse, high-performing controllers requires expensive network training and extensive tuning of a large number of hyperparameters. On the other hand, Covariance Matrix Adaptation MAP-Annealing (CMA-MAE), an evolution strategies (ES)-based quality diversity algorithm, does not have these limitations and has achieved state-of-the-art performance on standard QD benchmarks. However, CMA-MAE cannot scale to modern neural network controllers due to its quadratic complexity. We leverage efficient approximation methods in ES to propose three new CMA-MAE variants that scale to high dimensions. Our experiments show that the variants outperform ES-based baselines in benchmark robotic locomotion tasks, while being comparable with or exceeding state-of-the-art deep reinforcement learning-based quality diversity algorithms.**

***Index Terms*—Evolutionary robotics, reinforcement learning.**

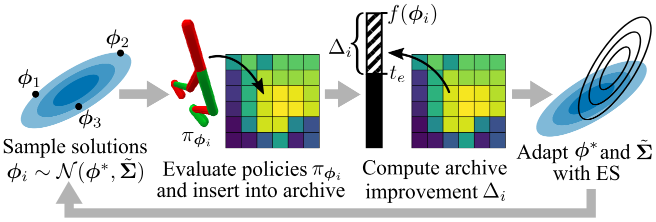


Fig. 1. We propose variants of the CMA-MAE algorithm that scale to high-dimensional controllers. The variants maintain a Gaussian search distribution with mean $\phi^*$ and approximate covariance matrix $\tilde{\boldsymbol{\Sigma}}$. Solutions $\phi_i$ sampled from the Gaussian are evaluated and inserted into an archive, where they generate improvement feedback $\Delta_i$ based on their objective value $f(\phi_i)$ and a threshold $t_e$ that each archive cell maintains. Finally, the Gaussian is updated with an evolution strategy (ES). Our variants differ from CMA-MAE by incorporating scalable ESs, as the CMA-ES used in CMA-MAE has $\Theta(n^2)$ time complexity per sampled solution.

## I. Introduction

By GENERATING a diverse collection of controllers, we can endow a robot with a variety of useful behaviors. For example, one popular approach in robotic locomotion has been to train a collection of neural network controllers to enable a walking robot to adapt to damage [1], [2], [3], [4]. The controllers differ by how often each foot contacts the ground, such that if a foot is damaged, the robot can select a controller that does not rely on that foot.

Searching for diverse controllers may be viewed as a quality diversity (QD) optimization problem [5]. In QD, the goal is to find solutions $\phi$ that are diverse with respect to one or more measure functions $m_i(\phi)$ while maximizing an objective function $f(\phi)$. In the locomotion example presented, we search for neural network controller policies $\pi_\phi$ parameterized by $\phi$. Each controller should satisfy a unique output of the measure function by using its feet in a different manner from the other controllers, while optimizing the objective by walking forward quickly.

A QD algorithm must balance two aspects given a limited compute budget: *exploring* measure space and *optimizing* the objective. In our locomotion example, exploration finds new controllers that use the robot's feet a different amount, and optimization makes existing controllers walk faster.

Prior algorithms [3], [4] seem to strike a balance between these two aspects of QD, leading to state-of-the-art results. However, these algorithms have practical limitations due to their dependence on deep reinforcement learning (RL) methods. Namely, they must perform time-consuming training of a neural network and have many hyperparameters.

Recent work [6] suggests evolution strategies (ES) as a compelling alternative to deep RL methods when optimizing a single controller. Compared with deep RL, ESs do not require network training, and ESs such as the Covariance Matrix Adaptation Evolution Strategy (CMA-ES) [7] are designed to have almost no hyperparameters. Given these benefits, prior work [2], [3] has developed QD algorithms based on ESs, but these methods have not yet matched the performance of deep RL-based QD methods.

On the other hand, the recently proposed ES-based Covariance Matrix Adaptation MAP-Annealing (CMA-MAE) algorithm [8] has proven adept at trading off the exploration and

Manuscript received 12 May 2023; accepted 26 August 2023. Date of publication 7 September 2023; date of current version 14 September 2023. This letter was recommended for publication by Associate Editor S. Funabashi and Editor T. Ogata upon evaluation of the reviewers' comments. This work was supported in part by the NSF CAREER under Grant 2145077, in part by NSF NRI under Grant 2024949, in part by NSF GRFP under Grant DGE-1842487, and in part by NVIDIA Academic Hardware Grant. *(Corresponding author: Bryon Tjanaka.)*

The authors are with the Viterbi School of Engineering, Computer Science Department, University of Southern California, Los Angeles, CA 90007 USA (e-mail: tjanaka@usc.edu; mfontain@usc.edu; dhlee@usc.edu; kalkar@usc.edu; nikolaid@usc.edu).

This letter has supplementary downloadable material available at https://doi.org/10.1109/LRA.2023.3313012, provided by the authors.

Digital Object Identifier 10.1109/LRA.2023.3313012

objective optimization aspects of QD. Tuning a single hyperparameter $\alpha \in [0, 1]$ in CMA-MAE enables blending these two aspects, yielding state-of-the-art performance on QD benchmarks. We hypothesize that CMA-MAE's ability to balance this tradeoff would enable it to excel in training neural network controllers for robotic locomotion tasks.

While CMA-MAE excels at moderate-dimensional domains, it is intractable for modern neural network controllers because such controllers are *high-dimensional*, i.e., they have thousands or even millions of parameters. Internally, CMA-MAE guides the QD search with one or more CMA-ES instances [7]. Since CMA-ES's time complexity is quadratic in the number of parameters, it cannot scale to such controllers.

CMA-ES's complexity arises from how it models the search distribution with a Gaussian that has a full rank $n \times n$ covariance matrix. However, by replacing this full matrix with sparse approximations, prior work [9] creates variants of CMA-ES that scale to high-dimensional problems.

*Our key insight is that we can scale CMA-MAE to high-dimensional controllers by adopting such approximations in its CMA-ES components*. Following this insight, we propose three scalable CMA-MAE variants (Section III). To understand their performance and runtime properties, we study these variants on optimization benchmarks (Section IV). Next, we evaluate the variants on robotic locomotion tasks (Section V). We show that our variants are the highest-performing QD methods based solely on ES. Furthermore, they are comparable to or exceed the state-of-the-art deep RL-based QD method PGA-ME [4] on three of four tasks, while inheriting the aforementioned practical benefits of ES. We are excited about future applications in other domains, such as robotic manipulation [10] and scenario generation [11], and we have open-sourced our variants in the pyribs library [12].

## II. Background

### A. Formulation

*Quality diversity (QD):* Drawing from the definition in prior work [13], QD considers an objective function $f(\boldsymbol{\phi})$ and $k$-dimensional measure function $\boldsymbol{m}(\boldsymbol{\phi})$,[1] where $\boldsymbol{\phi} \in \mathbb{R}^n$ is an $n$-dimensional solution. The outputs of $\boldsymbol{m}$ form a $k$-dimensional *measure space* $\mathcal{X}$. The *QD objective* is to find, for every $\boldsymbol{x} \in \mathcal{X}$, a solution $\boldsymbol{\phi}$ such that $\boldsymbol{m}(\boldsymbol{\phi}) = \boldsymbol{x}$ and $f(\boldsymbol{\phi})$ is maximized. Solving this QD objective would require infinite memory since $\mathcal{X}$ is a continuous space, so algorithms based on MAP-Elites [14] relax the QD objective by discretizing $\mathcal{X}$ into a tesselation $\mathcal{Y}$ of $M$ cells. Then, the QD objective is to maximize the (sum of) objective values of an *archive* $\mathcal{A}$ containing solutions $\boldsymbol{\phi}_{1..M}$, i.e., $\max_{\boldsymbol{\phi}_{1..M}} \sum_{i=1}^{M} f(\boldsymbol{\phi}_i)$. Furthermore, $\boldsymbol{\phi}_{1..M}$ are constrained such that each $\boldsymbol{\phi}_i$ has measures $\boldsymbol{m}(\boldsymbol{\phi}_i)$ corresponding to a unique cell in $\mathcal{Y}$.

*Quality diversity reinforcement learning (QD-RL):* As defined in prior work [3], QD-RL is a special instance of QD where $\boldsymbol{\phi}$ parameterizes a reinforcement learning (RL) agent's policy $\pi_{\boldsymbol{\phi}}$, and the objective is the agent's expected discounted return in a Markov Decision Process (MDP) [15]. QD-RL also includes a $k$-dimensional measure function $\boldsymbol{m}(\boldsymbol{\phi})$ that describes the agent's behavior during an episode.

### B. Large-Scale Evolution Strategies

An evolution strategy (ES) [16] optimizes continuous parameters by adapting a population of solutions such that the population is more likely to attain high performance. A *large-scale* ES scales to high-dimensional search spaces.

OpenAI-ES [6] is one large-scale ES notable for performing well in RL domains. It represents a population with an isotropic Gaussian and updates only the Gaussian's mean by passing approximated gradients to Adam [17].

Several large-scale ESs build on Covariance Matrix Adaptation Evolution Strategy (CMA-ES) [7], an approximate second-order method that achieves state-of-the-art results in black-box optimization [18]. CMA-ES models a distribution of search directions with a Gaussian $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$. Every iteration, CMA-ES samples $\lambda$ solutions from this Gaussian and updates it based on rankings of the solutions' performance.

CMA-ES itself does not scale to high dimensions, as it requires $\Theta(n^2)$ space and $\Theta(n^2)$ runtime per sampled solution. The space is due to the $n \times n$ covariance matrix $\boldsymbol{\Sigma}$, while runtime stems from two operations. First, updating $\boldsymbol{\Sigma}$ requires matrix-vector multiplications. Second, since it is easy to sample from the standard Gaussian $\mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$ on a computer, sampling from $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ is implemented as:

$$\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma}) \sim \boldsymbol{\mu} + \mathcal{N}(\boldsymbol{0}, \boldsymbol{\Sigma}) \sim \boldsymbol{\mu} + \boldsymbol{\Sigma}^{\frac{1}{2}} \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}) \qquad (1)$$

The *transformation matrix* $\boldsymbol{\Sigma}^{\frac{1}{2}}$ requires an $O(n^3)$ eigendecomposition, which CMA-ES amortizes to $O(n^2)$ per sampled solution by only recomputing $\boldsymbol{\Sigma}^{\frac{1}{2}}$ every $\frac{n}{\lambda}$ iterations.

Multiple variants [9] of CMA-ES scale to high dimensions by replacing the full covariance matrix with an efficient approximation. We incorporate OpenAI-ES and two such variants to scale CMA-MAE to high dimensions.

### C. MAP-Elites

Many QD algorithms, including those in this work, build on Multi-dimensional Archive of Phenotypic Elites (MAP-Elites) [14]. The vanilla version of MAP-Elites divides the measure space into an archive of evenly-sized grid cells. Then, it generates solutions by sampling existing solutions from the archive and applying a genetic operator. These new solutions are inserted into archive cells based on their measures. If they land in the same cell as a previous solution, they replace the solution only if they have a higher objective.

One recent line of work integrates CMA-ES into MAP-Elites to optimize for the QD objective (Section II-A). In Covariance Matrix Adaptation MAP-Elites (CMA-ME) [19], CMA-ES directly samples solutions, adapting the search distribution to find solutions that create the greatest archive improvement. CMA-ME runs multiple CMA-ES instances in parallel, each

[1] It is common to define $\boldsymbol{m}(\boldsymbol{\phi})$ via $k$ separate measure functions $m_i(\boldsymbol{\phi})$. Prior work also refers to measure function outputs as behavior descriptors or behavior characteristics.

encapsulated in an *emitter* — emitters are QD algorithm components that generate solutions for evaluation [12], [19]. Meanwhile, Covariance Matrix Adaptation MAP-Elites via a Gradient Arborescence (CMA-MEGA) [13] operates in the differentiable quality diversity (DQD) setting, where exact objective and measure gradients are available. Here, instead of sampling solution parameters, CMA-ES branches from a solution point by sampling coefficients that form linear combinations of the objective and measure gradients.

Multiple methods extend MAP-Elites to train neural network controllers, as vanilla MAP-Elites performs poorly in such problems [2], [3], [4]. For instance, CMA-MEGA cannot be applied to QD-RL since it assumes gradients are provided, and such gradients are often unavailable in RL due to non-differentiable environments. Hence, recent work [3] introduces CMA-MEGA variants that instead approximate the gradients. Meanwhile, MAP-Elites with Evolution Strategies (ME-ES) [2] integrates OpenAI-ES to improve the objective value of a solution point or move the point to a new area of the archive. Finally, Policy Gradient Assisted MAP-Elites (PGA-ME) [4] replaces the genetic operator with two operations: 1) gradient ascent, performed with TD3 [20], and 2) crossover, performed with a genetic algorithm [21]. We include these methods as experimental baselines.

### *D. CMA-MAE*

We extend Covariance Matrix Adaptation MAP-Annealing (CMA-MAE) [8], a method that builds on CMA-ME and achieves state-of-the-art performance on QD benchmarks.

The key difference between CMA-MAE and CMA-ME is a *soft archive* that enables balancing between optimizing the objective and searching for solutions with new measure values. This soft archive records a *threshold* $t_e$ for each cell $e$. $t_e$ is initialized to a minimum objective $min_f$. When a solution $\boldsymbol{\phi}$ is inserted into the archive, it is placed into its corresponding cell $e$ if its objective value $f(\boldsymbol{\phi})$ exceeds $t_e$. Then, $t_e$ is updated via polyak averaging $t_e \leftarrow (1-\alpha)t_e + \alpha f(\boldsymbol{\phi})$, where $\alpha \in [0,1]$ is the *archive learning rate*. Finally, the insertion returns an *improvement value* $\Delta_i \leftarrow f(\boldsymbol{\phi}) - t_e$, where higher values indicate greater archive improvement. Note that during insertion, the solution's objective value only needs to cross the threshold, rather than the objective value of the solution previously in the cell. Thus, implementations must track the best solutions separately, as the archive does not always store them like MAP-Elites does.

Like CMA-ME, CMA-MAE maintains one or more emitters. Each emitter contains a CMA-ES instance that directly samples solutions from a Gaussian. By updating the Gaussian based on rankings of the solutions' improvement values, CMA-ES moves the Gaussian towards solutions more likely to generate high improvement.

The archive learning rate $\alpha$ is a key parameter in CMA-MAE. When $\alpha = 0$, the threshold remains at $min_f$, so the improvement $\Delta_i$ always equals the objective $f(\boldsymbol{\phi})$ (minus a constant $min_f$). This makes CMA-MAE equivalent to CMA-ES, as it optimizes solely for the objective. When $\alpha = 1$, the threshold is equal to the objective value of the solution currently in the cell, which means there is minimal improvement for inserting a solution into a cell with an existing solution. In this case, CMA-MAE is equivalent to CMA-ME, which always prioritizes discovering new solutions in measure space over improving existing solutions. Varying $\alpha$ from 0 to 1 smoothly trades off between these two extremes.

## III. Scaling CMA-MAE

**Algorithm 1:** CMA-MAE Variants. Highlighted Lines Show Differences From CMA-MAE [8].

```
CMA-MAE variants ($eval, \boldsymbol{\phi}_0, N, \psi, \lambda, \sigma, \alpha, min_f$):
    Input: $eval$ function that rolls out policy $\pi_{\boldsymbol{\phi}}$ and
           outputs objective $f(\boldsymbol{\phi})$ and measures
           $\boldsymbol{m}(\boldsymbol{\phi})$, initial solution $\boldsymbol{\phi}_0$, iterations $N$,
           number of emitters $\psi$, batch size $\lambda$, initial
           step size $\sigma$, archive learning rate $\alpha$,
           minimum objective $min_f$
    Result: Generates $N\psi\lambda$ solutions, storing elites in
            an archive $\mathcal{A}$
    Initialize archive $\mathcal{A}$ and threshold $t_e \leftarrow min_f$ for
      every cell $e$
    Initialize $\psi$ emitters, each with mean $\boldsymbol{\phi}^* \leftarrow \boldsymbol{\phi}_0$,
      covariance matrix approximation $\tilde{\boldsymbol{\Sigma}} \leftarrow \sigma\boldsymbol{I}$, and
      internal parameters $\boldsymbol{p}$
    for $iter \leftarrow 1..N$ do
        for Emitter 1 .. Emitter $\psi$ do
            for $i \leftarrow 1..\lambda$ do
                $\boldsymbol{\phi}_i \sim \mathcal{N}(\boldsymbol{\phi}^*, \tilde{\boldsymbol{\Sigma}})$
                $f(\boldsymbol{\phi}_i), \boldsymbol{m}(\boldsymbol{\phi}_i) \leftarrow eval(\boldsymbol{\phi}_i)$
                $e \leftarrow$ calculate_archive_cell($\mathcal{A}, \boldsymbol{m}$)
                $\Delta_i \leftarrow f(\boldsymbol{\phi}_i) - t_e$
                if $f(\boldsymbol{\phi}_i) > t_e$ then
                    Replace the solution in cell $e$ of
                      archive $\mathcal{A}$ with $\boldsymbol{\phi}_i$
                    $t_e \leftarrow (1-\alpha)t_e + \alpha f(\boldsymbol{\phi}_i)$
            Rank $\boldsymbol{\phi}_i$ by $\Delta_i$
            Adapt $\boldsymbol{\phi}^*, \tilde{\boldsymbol{\Sigma}}, \boldsymbol{p}$ based on improvement
            ranking $\Delta_i$
            if ES algorithm converges then
                Restart emitter with $\boldsymbol{\phi}^* \leftarrow$ a randomly
                selected elite in $\mathcal{A}$, $\tilde{\boldsymbol{\Sigma}} \leftarrow \sigma\boldsymbol{I}$, and new
                internal parameters $\boldsymbol{p}$
```

In CMA-MAE, each emitter uses CMA-ES to update its Gaussian search distribution. Since CMA-ES requires $\Theta(n^2)$ space and $\Theta(n^2)$ runtime per solution (with $n$ the solution dimension), CMA-MAE cannot train high-dimensional neural network controllers. To scale CMA-MAE, we propose three variants that replace CMA-ES with large-scale ESs. These variants differ primarily in the complexity of their covariance matrix approximation, and each variant is named by taking its large-scale ES's name and replacing "ES" with "MAE":

- **LM-MA-MAE** substitutes Limited-Memory Matrix Adaptation ES (LM-MA-ES) [22], a large-scale CMA-ES variant that approximates the transformation matrix $\boldsymbol{\Sigma}^{\frac{1}{2}}$ with $k \ll n$ $n$-dimensional vectors, each representing a

TABLE I
OPTIMIZATION BENCHMARK RESULTS

| | Sphere 100 | | | | Sphere 1000 | | | | Arm 100 | | | | Arm 1000 | | | | Maze | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | QD | Cov | Best | Time | QD | Cov | Best | Time | QD | Cov | Best | Time | QD | Cov | Best | Time | QD | Cov | Best | Time |
| CMA-MAE | 0.541 | 0.64 | 98.82 | 2.00 | 0.027 | **0.03** | **100.00** | 195.60 | 0.787 | **0.79** | **99.98** | 2.00 | 0.763 | 0.76 | 99.96 | 193.41 | 0.663 | 0.71 | **100.00** | 1.06 |
| LM-MA-MAE | 0.545 | 0.65 | 99.14 | 1.35 | **0.028** | **0.03** | **100.00** | 13.42 | 0.784 | **0.79** | **99.98** | 1.36 | **0.766** | **0.77** | **99.98** | 13.29 | 0.640 | 0.69 | **100.00** | 0.72 |
| sep-CMA-MAE | **0.553** | **0.66** | 98.74 | 0.97 | **0.028** | **0.03** | **100.00** | 3.87 | **0.789** | **0.79** | **99.98** | 0.98 | 0.763 | 0.76 | 99.96 | 4.18 | 0.741 | 0.79 | **100.00** | 0.47 |
| OpenAI-MAE | 0.007 | 0.01 | **100.00** | **0.66** | 0.007 | 0.01 | **100.00** | **2.67** | 0.770 | 0.78 | 99.97 | **0.79** | 0.714 | 0.72 | 99.96 | **3.19** | **0.751** | **0.81** | **100.00** | **0.38** |

We display the metrics described in Sec. IV-A as the mean over 10 trials. QD score is shown as a multiple of $10^6$, and Time is measured in minutes.

TABLE II
ONE-WAY AND WELCH'S ONE-WAY ANOVA RESULTS FOR EACH DEPENDENT VARIABLE ON EACH BENCHMARK

| | QD Score | Execution Time |
|---|---|---|
| Sphere 100 | Welch's $F(3, 15.31) = 63{,}320$ | Welch's $F(3, 18.95) = 57{,}608$ |
| Sphere 1000 | Welch's $F(3, 15.80) = 688.45$ | Welch's $F(3, 15.62) = 3.08\text{e}6$ |
| Arm 100 | $F(3, 36) = 5.46$ | Welch's $F(3, 18.48) = 67{,}102$ |
| Arm 1000 | $F(3, 36) = 258.21$ | Welch's $F(3, 19.10) = 2.18\text{e}6$ |
| Maze | $F(3, 36) = 0.978\,(p > 0.05)$ | $F(3, 36) = 891.73$ |

All $p$-values are less than 0.005, except for QD Score in Maze. Note that large between-group variation led to several high $F$ statistics.

different direction of the search distribution. This rank-$k$ approximation leads to $\Theta(kn)$ complexity.

- **sep-CMA-MAE** substitutes Separable CMA-ES (sep-CMA-ES) [23], a large-scale CMA-ES variant that constrains the covariance matrix $\mathbf{\Sigma}$ to be diagonal, yielding $\Theta(n)$ complexity.
- **OpenAI-MAE** substitutes OpenAI-ES [6]. As OpenAI-ES is not a CMA-ES variant, it differs from CMA-ES in several mechanisms, but it nevertheless represents the search distribution with a Gaussian, specifically an isotropic Gaussian with constant covariance $\sigma \boldsymbol{I}$. Though the covariance is constant, vector operations on the solutions still necessitate $\Theta(n)$ complexity.

The listed complexities refer to 1) the space required per emitter, as each emitter maintains its own ES instance, and 2) the runtime required per sampled solution, which is the same regardless of the number of emitters.

Algorithm 1 and Fig. 1 show an overview of the variants. Each variant begins by initializing the archive along with each emitter's ES parameters (lines 2–3). This step includes initializing the covariance matrix approximation $\tilde{\mathbf{\Sigma}}$ in lieu of the full covariance matrix $\mathbf{\Sigma}$ used in CMA-MAE. Next, each variant repeatedly queries the emitters for solutions (line 4). Each emitter (line 5) samples $\lambda$ solutions from the distribution $\mathcal{N}(\boldsymbol{\phi}^*, \tilde{\mathbf{\Sigma}})$ (lines 6–7). Note that the sampling procedure depends on the approximation employed by the variant. Once sampled, the solutions are evaluated and inserted into the archive if they cross their cell's threshold $t_e$ (lines 9–13). Then, $\boldsymbol{\phi}^*$, $\tilde{\mathbf{\Sigma}}$, and the ES's parameters are updated based on the solutions' improvement ranking (lines 14–15), such that the emitter is more likely to sample solutions with high improvement on the next iteration. Finally, the emitter restarts if the ES converges (lines 16–17). We adopt default update and convergence rules from each ES.

## IV. OPTIMIZATION BENCHMARKS

Replacing CMA-ES with large-scale ESs in our CMA-MAE variants raises two questions: 1) Since our variants model the search distribution with an approximate Gaussian rather than a full Gaussian, how do they perform relative to each other and relative to CMA-MAE? 2) In practice, are the variants faster than CMA-MAE? While our goal is to train neural network controllers for robotic locomotion, it is impractical to answer these questions in that domain, since CMA-MAE's quadratic complexity prevents it from training high-dimensional controllers. Thus, we first study the variants on lower-dimensional benchmarks.

### A. Experimental Setup

*Domains:* We consider three QD benchmarks: 1) In *sphere linear projection* [19], the objective is the sphere function $f(\boldsymbol{x}) = \sum_{i=1}^{n} x_i^2$, and the measure function linearly projects solutions into a 2D space. 2) *Arm repertoire* [21] considers a planar robotic arm with $n$ equally-sized links. The objective is to find configurations of the $n$ joint angles where the angles have low variance, giving the arm a smooth appearance. The measures indicate the $x$-$y$ position of the end of the arm. 3) *Hard maze* [24] considers a robot that navigates a maze for 250 timesteps. We use the Kheperax [25] implementation, where the objective is the robot's energy consumption, and the measures are the final $x$-$y$ position. The robot is controlled by a neural network with two hidden layers of size 8 and 138 parameters total. In all domains, we linearly transform the objective to the range $[0, 100]$. We consider 100- and 1000-dimensional versions of sphere and arm, yielding five domains: *Sphere 100*, *Sphere 1000*, *Arm 100*, *Arm 1000*, *Maze*.

We select these benchmarks since they are well-studied in the QD literature and exhibit different properties. For instance, Sphere has a separable objective, and its measure space is intentionally distorted to make it difficult to find new archive solutions. In contrast, the variance objective in Arm is non-separable, but its measure space tends to be easier to explore, with prior work [8] showing that even vanilla MAP-Elites fills most of the archive. Finally, as a small-scale QD-RL benchmark, Maze has a less intuitive mapping from neural network parameters to objectives and measures.

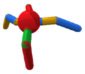
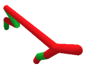
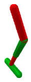
*Metrics:* Our primary metric is *QD score* [5], which holistically measures algorithm performance by summing the objectives of all archive solutions. To ensure no solution subtracts from the score (this happens if objectives are negative), we subtract the minimum objective (i.e., CMA-MAE's $min_f$) from all solutions' objectives before computing the score. Note that $min_f = 0$ in all domains in this section, but $min_f < 0$ in all environments in Section V. We also record *archive coverage* (fraction of archive cells containing a solution), *best*

TABLE III
RESULTS FROM TRAINING HIGH-DIMENSIONAL CONTROLLERS

| | QD Ant | | | | | QD Half-Cheetah | | | | | QD Hopper | | | | | QD Walker | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | QD | Cov | Best | Time | Space | QD | Cov | Best | Time | Space | QD | Cov | Best | Time | Space | QD | Cov | Best | Time | Space |
| LM-MA-MAE | **0.761** | 0.36 | 2,297.91 | 7.74 | 112 | 2.830 | 0.62 | 2,243.82 | 6.45 | 87 | 1.135 | 0.54 | 2,845.35 | 4.45 | 77 | 0.327 | 0.42 | 1,289.64 | 5.44 | 84 |
| sep-CMA-MAE | 0.730 | 0.36 | 2,199.41 | 8.01 | 112 | **2.892** | **0.63** | 2,317.81 | **5.77** | 87 | **1.173** | 0.55 | **2,884.33** | **3.91** | 77 | 0.325 | 0.41 | 1,258.44 | 4.67 | 84 |
| OpenAI-MAE | 0.638 | 0.34 | 2,073.62 | **7.25** | 111 | 2.675 | 0.61 | 2,172.15 | 6.30 | 86 | 1.153 | 0.52 | 2,656.07 | 4.51 | 77 | 0.360 | 0.45 | 1,262.87 | 5.67 | 84 |
| PGA-ME | 0.582 | 0.34 | **2,711.65** | 18.19 | 374 | 2.682 | 0.56 | **2,722.17** | 18.19 | 325 | 1.002 | 0.53 | 2,740.55 | 11.89 | 216 | **0.865** | 0.50 | **2,685.16** | 12.42 | 291 |
| CMA-MEGA (ES) | 0.591 | 0.37 | 2,122.80 | 7.43 | **110** | 2.574 | 0.59 | 2,238.03 | 7.14 | **85** | 0.502 | 0.53 | 1,508.15 | 3.93 | **76** | 0.145 | 0.47 | 781.89 | **3.78** | **83** |
| CMA-MEGA (TD3, ES) | 0.598 | **0.40** | 2,349.77 | 14.92 | 374 | 2.693 | 0.59 | 2,495.93 | 18.85 | 325 | 0.935 | 0.52 | 2,498.05 | 11.18 | 216 | 0.789 | **0.54** | 2,291.46 | 10.96 | 291 |
| ME-ES | 0.138 | 0.14 | 955.12 | 9.28 | 111 | 0.805 | 0.42 | 848.60 | 10.09 | 86 | 0.185 | 0.42 | 1,043.35 | 4.46 | 77 | 0.037 | 0.30 | 388.58 | 4.32 | 84 |
| MAP-Elites | 0.444 | 0.38 | 1,160.22 | 7.28 | **110** | 2.371 | 0.60 | 1,712.17 | 7.09 | **85** | 0.833 | **0.56** | 2,420.35 | 4.40 | **76** | 0.139 | 0.51 | 753.96 | 5.53 | **83** |

We display the corrected metrics described in Sec. V-A as the mean over 10 trials. QD score is shown as a multiple of $10^6$, and Time is measured in hours. We also display the memory usage of each algorithm's components (Space), measured in megabytes. Note that Space is constant across all trials.

TABLE IV
QDGYM LOCOMOTION ENVIRONMENTS [26]

| | QD Ant | QD Half-Cheetah | QD Hopper | QD Walker |
|---|---|---|---|---|
| |  |  |  | |
| Archive Grid | [6,6,6,6] | [32,32] | [1024] | [32,32] |
| Parameters | 21,256 | 20,742 | 18,947 | 20,230 |
| Min. Objective | -374.70 | -2,797.52 | -362.09 | -67.17 |

*performance* (highest objective in the archive), and *execution time* (wall-clock time of the experiment). In our tables, we abbreviate these metrics as "QD", "Cov", "Best", and "Time."

*Procedure:* In each domain (Sphere 100, Sphere 1000, Arm 100, Arm 1000, Maze), we conduct a between-groups study with the algorithm (CMA-MAE, LM-MA-MAE, sep-CMA-MAE, OpenAI-MAE) as independent variable and the QD score and execution time as dependent variables. We repeat experiments for 10 trials, where each trial executes an algorithm in a domain for 2 million solution evaluations. All experiments run on a single CPU core, except in Maze, where we run evaluations on an NVIDIA RTX A6000.

*Hyperparameters:* CMA-MAE and its variants run with archive learning rate $\alpha = 0.001$ (except $\alpha = 0.01$ in Maze) and $\psi = 5$ emitters. Each emitter has batch size $\lambda = 40$ and initial step size $\sigma = 0.02$. LM-MA-MAE sets $k = \lambda = 40$. The Adam optimizer for OpenAI-ES in OpenAI-MAE uses learning rate 0.01 and L2 regularization coefficient 0.005.

*Hypotheses:* All methods considered model their search distribution with a Gaussian or approximate Gaussian. We predict that methods with a more complex distribution will perform better but take longer to execute. To elaborate, the first and simplest algorithm in this ranking is OpenAI-MAE, which models a fixed isotropic Gaussian. Since this Gaussian has a constant shape that cannot adapt to the search space, we predict OpenAI-MAE will have the lowest performance. However, since OpenAI-MAE only updates the mean of the Gaussian, it should be the fastest algorithm.

Second, sep-CMA-MAE models a diagonal Gaussian. Since this distribution can change shape and adapt over time, we predict it will lead to higher performance when guiding the QD search. While the diagonal Gaussian gives sep-CMA-MAE the same linear complexity as OpenAI-MAE, sep-CMA-MAE will likely be slower, as it requires additional operations to update the diagonal covariance matrix.

Third, LM-MA-MAE uses a rank-$k$ approximation. While the Gaussian in sep-CMA-MAE is limited to being axis-aligned since it is diagonal, the rank-$k$ approximation can represent a more complex Gaussian that is not necessarily axis-aligned. This property should give LM-MA-MAE greater flexibility to adapt to the search space, leading to higher performance. However, the $\Theta(kn)$ complexity will likely make LM-MA-MAE slower than sep-CMA-MAE.

Finally, CMA-MAE maintains a full Gaussian, which should be highly flexible and able to adeptly guide the QD search. The $\Theta(n^2)$ complexity will likely make it the slowest algorithm. Our hypotheses may be summarized as:

*H1:* The QD score will be ranked OpenAI-MAE < sep-CMA-MAE < LM-MA-MAE < CMA-MAE.

*H2:* The execution time will be ranked OpenAI-MAE < sep-CMA-MAE < LM-MA-MAE < CMA-MAE.

### B. Results

Table I summarizes our results. To analyze the results, we ran an ANOVA for each dependent variable in each domain. Before running the ANOVAs, we verified the data were normally distributed through visual inspection and the Shapiro-Wilk test. Next, we checked homoscedasticity with Levene's test. In homoscedastic settings, we ran a one-way ANOVA, and in non-homoscedastic settings, we ran Welch's one-way ANOVA. In almost all domains, we found significant differences across the algorithms for both dependent variables (Table II). To analyze the rankings in H1 and H2, we performed pairwise comparisons with Tukey's HSD test or a Games-Howell test, depending on whether the data were homoscedastic or not, respectively.

*H1:* In all Sphere and Arm domains, OpenAI-MAE underperformed all other methods. There were no significant differences among the other methods, except that sep-CMA-MAE outperformed CMA-MAE in Sphere 100. In Maze, while there was a trend towards OpenAI-MAE being the best-performing, large variances meant that there were no significant differences among any methods.

Overall, our results fail to support **H1**. Namely, we find that more complex search distributions do not necessarily yield better QD score. On one hand, as predicted, the most basic distribution (OpenAI-MAE's isotropic Gaussian) underperforms CMA-MAE in Sphere and Arm. However, there is no significant difference between OpenAI-MAE and CMA-MAE in Maze. Furthermore, we found no significant differences in any domain

TABLE V
PAIRWISE COMPARISONS FOR QD SCORE AMONG THE VARIANTS (FOR **H3**) AND BETWEEN THE VARIANTS AND THE BASELINES (FOR **H4**) IN THE LOCOMOTION ENVIRONMENTS

| | QD Ant | | | | | | | | QD Half-Cheetah | | | | | | | | QD Hopper | | | | | | | | QD Walker | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | LM-MA-MAE | sep-CMA-MAE | OpenAI-MAE | PGA-ME | MEGA (ES) | MEGA (TD3, ES) | ME-ES | MAP-Elites | LM-MA-MAE | sep-CMA-MAE | OpenAI-MAE | PGA-ME | MEGA (ES) | MEGA (TD3, ES) | ME-ES | MAP-Elites | LM-MA-MAE | sep-CMA-MAE | OpenAI-MAE | PGA-ME | MEGA (ES) | MEGA (TD3, ES) | ME-ES | MAP-Elites | LM-MA-MAE | sep-CMA-MAE | OpenAI-MAE | PGA-ME | MEGA (ES) | MEGA (TD3, ES) | ME-ES | MAP-Elites |
| LM-MA-MAE | ∅ | − | > | > | > | > | > | > | ∅ | − | − | − | − | − | > | > | ∅ | − | − | > | > | > | > | > | ∅ | − | − | < | > | < | > | > |
| sep-CMA-MAE | − | ∅ | − | > | − | − | > | > | − | ∅ | − | − | − | − | > | > | − | ∅ | − | > | > | > | > | > | − | ∅ | − | < | > | < | > | > |
| OpenAI-MAE | < | − | ∅ | − | − | − | > | > | − | − | ∅ | − | − | − | > | > | − | − | ∅ | > | > | > | > | > | − | − | ∅ | < | > | < | > | > |

Each entry compares the method in the row to the method in the column; e.g., LM-MAMAE was significantly better than OpenAI-MAE in QD Ant. The symbols used are < (significantly less), − (no significant difference), > (significantly greater), ∅ (invalid comparison). We abbreviate CMA-MEGA to MEGA for brevity.

TABLE VI
WELCH'S ONE-WAY ANOVA RESULTS IN EACH LOCOMOTION ENVIRONMENT

| | QD Score | Execution Time |
|---|---|---|
| QD Ant | Welch's $F(7, 30.30) = 287.34$ | Welch's $F(7, 29.19) = 54.83$ |
| QD Half-Cheetah | Welch's $F(7, 30.66) = 207.25$ | Welch's $F(7, 29.55) = 97.66$ |
| QD Hopper | Welch's $F(7, 30.33) = 1017.19$ | Welch's $F(7, 30.39) = 60.56$ |
| QD Walker | Welch's $F(7, 29.29) = 500.89$ | Welch's $F(7, 29.88) = 111.61$ |

All $p$-values are less than 0.001.

between a simple diagonal Gaussian (sep-CMA-MAE) and a full Gaussian (CMA-MAE).

The only case where increasing search distribution complexity increases performance is with sep-CMA-MAE outperforming OpenAI-MAE in Sphere and Arm. Yet, increasing the complexity further (i.e., LM-MA-MAE's rank-$k$ approximation and CMA-MAE's full Gaussian) fails to garner further improvement.

*H2:* Pairwise comparisons found that the execution time of the algorithms matched the rankings in **H2**. The difference between CMA-MAE and the variants was particularly pronounced in the higher-dimensional Sphere 1000 and Arm 1000, where, on average, CMA-MAE took 14.5 times longer than LM-MA-MAE, the slowest variant. These results validate **H2**, showing that the variants are empirically faster to run than CMA-MAE, and that the variants become faster as their search distribution becomes simpler.

## V. TRAINING HIGH-DIMENSIONAL CONTROLLERS

We evaluate our CMA-MAE variants' abilities to train diverse, high-performing neural network controllers for robotic locomotion tasks in the QDGym benchmark [26].

### A. Experimental Setup

*Environments:* Table IV shows the QDGym environments considered in this work. These environments are *unidirectional*, i.e., the objective is to walk forward quickly, and the measures track the proportion of time that each of the robot's feet touches the ground, e.g., if a robot has four legs, it has four measures. As prior work [3] notes, the challenge in these environments arises from performing objective optimization across the entire archive. Namely, it is easy to find a single high-performing controller and fill the rest of the archive with controllers that stand in place and lift their legs to achieve different measures. However, it is difficult to make the robot walk quickly at all points in the measure space.

As in prior work [3], [4], each domain uses an archive with grid cells. The robot controller is a neural network mapping states to actions. The network has two hidden layers of size 128 and tanh activations and is initialized with Xavier initialization. For the minimum objective $min_f$, QDGym does not have predefined minimum objectives, but we adopt values from prior work [3] that recorded the minimum objective inserted into an archive during their experiments. Table IV includes the archive dimensions, number of parameters, and minimum objective in each domain.

*Baselines:* We compare our variants with five baselines: PGA-ME [4], two CMA-MEGA variants [3] that approximate gradients (CMA-MEGA (ES) and CMA-MEGA (TD3, ES)), ME-ES [2], and MAP-Elites. We adopt hyperparameters from the original papers for PGA-ME, the CMA-MEGA variants, and ME-ES, except ME-ES uses a population size of 200. Our MAP-Elites baseline uses isotropic Gaussian noise mutations with standard deviation $\sigma = 0.02$ and batch size 100. The CMA-MAE variants themselves use the same parameters as in the optimization benchmarks (Section IV-A).

*Procedure:* We conduct a between-groups study in each environment (QD Ant, QD Half-Cheetah, QD Hopper, QD Walker) with the algorithm (LM-MA-MAE, sep-CMA-MAE, OpenAI-MAE, PGA-ME, CMA-MEGA (ES), CMA-MEGA (TD3, ES), ME-ES, MAP-Elites) as independent variable and QD score and execution time as dependent variables. We repeat experiments for 10 trials, where each trial executes an algorithm for 1 million solution evaluations. Each algorithm runs single-threaded and has 100 CPUs allocated for solution evaluations on a high-performance cluster. In addition to these 100 CPUs, PGA-ME and CMA-MEGA (TD3, ES) are allocated one NVIDIA Tesla P100 GPU to train TD3.

*Corrected Metrics:* To save computation, we evaluate each solution for only one episode. However, unlike the optimization benchmarks, the locomotion environments are stochastic since each episode's initial state is randomly sampled. Thus, solutions may be inserted into archives due to inaccurate evaluations, e.g., a solution may obtain a high objective by chance. Hence, we report *corrected metrics* [27], [28], where we first re-evaluate all solutions in each final archive for 10 episodes, inserting them into a new, *corrected* archive based on their mean scores. We

TABLE VII
RESULTS FROM VARYING THE ARCHIVE LEARNING RATE $\alpha$ IN SEP-CMA-MAE IN THE LOCOMOTION ENVIRONMENTS

| | QD Ant | | | QD Half-Cheetah | | | QD Hopper | | | QD Walker | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | QD | Cov | Best | QD | Cov | Best | QD | Cov | Best | QD | Cov | Best |
| $\alpha = 0.0$ | 0.314 | 0.13 | **3,001.76** | 1.577 | 0.48 | 2,106.15 | 0.232 | 0.31 | 1,177.58 | 0.109 | 0.24 | 1,019.89 |
| $\alpha = 0.001$ | **0.730** | 0.36 | 2,199.41 | 2.892 | 0.63 | 2,317.81 | **1.173** | 0.55 | **2,884.33** | 0.325 | 0.41 | 1,258.44 |
| $\alpha = 0.01$ | 0.629 | **0.41** | 1,914.61 | 2.839 | 0.62 | 2,260.56 | 1.099 | 0.56 | 2,670.70 | **0.358** | 0.52 | **1,318.63** |
| $\alpha = 0.1$ | 0.595 | 0.38 | 2,285.24 | **2.939** | 0.63 | 2,413.25 | 1.054 | **0.57** | 2,696.78 | 0.318 | **0.53** | 1,277.05 |
| $\alpha = 1.0$ | 0.387 | 0.31 | 2,132.11 | 2.897 | **0.63** | **2,464.52** | 0.647 | 0.55 | 1,964.28 | 0.147 | 0.48 | 853.18 |

We display the corrected metrics described in Sec. V-A as the mean over 10 trials. QD score is shown as a multiple of $10^6$.

then compute the metrics from Section IV-A over this corrected archive.

*Hypotheses:* Among the variants, our performance (QD score) prediction remains the same as on the optimization benchmarks (Section IV-A), i.e., the variant with the simplest search distribution (OpenAI-MAE) will perform worst, and more powerful search distributions (sep-CMA-MAE followed by LM-MA-MAE) will improve performance. While the optimization benchmark results (Section IV-B) did not support this prediction, higher dimensionality in the locomotion environments may highlight differences between the variants.

Compared to the baselines, we believe the smooth improvement ranking in the variants will enable balancing objective optimization and measure space exploration, yielding better performance. To elaborate, CMA-MEGA (ES) and CMA-MEGA (TD3, ES) use a standard MAP-Elites archive (equivalent to setting $\alpha = 1$ in CMA-MAE's soft archive), so we think they will focus too much on exploration. PGA-ME and ME-ES separate measure space exploration from objective optimization with distinct operations; this separation may be less effective than blending the two aspects.

We predict that execution time differences among the variants will be the same as on the optimization benchmarks; i.e., variants with simpler search distributions will have faster runtimes. Compared to the baselines, we believe a key factor will be whether a method includes deep RL components. Unlike the CMA-MAE variants, PGA-ME and CMA-MEGA (TD3, ES) include components of TD3 [20] to train actor and critic networks, and these training steps are often time-consuming. Our hypotheses may be summarized as follows:

*H3:* The performances of the CMA-MAE variants will be ordered as OpenAI-MAE < sep-CMA-MAE < LM-MA-MAE.

*H4:* All CMA-MAE variants will outperform all baselines.

*H5:* The execution times of the CMA-MAE variants will be ordered as OpenAI-MAE < sep-CMA-MAE < LM-MA-MAE.

*H6:* All CMA-MAE variants will be faster than deep RL-based baselines, i.e., PGA-ME and CMA-MEGA (TD3, ES).

## B. *Results*

Table III summarizes our results. Following our analysis procedure in Section IV-B, we first verified normality through visual inspection and the Shapiro-Wilk test. Next, Levene's test showed homoscedasticity was violated in all environments. Thus, we ran Welch's one-way ANOVA (Table VI), finding significant differences in all cases. Finally, we performed pairwise comparisons with the Games-Howell test.

*H3:* Table V shows pairwise comparisons of corrected QD scores for **H3** and **H4**. We find **H3** unsupported, as there tends to be no significant difference among the variants. While these results do not align with Section IV-B's findings that OpenAI-MAE often underperforms the other variants, both experiments show that more complex search distributions do not necessarily yield higher performance.

*H4:* Table V shows that the CMA-MAE variants outperform or are not significantly different from prior ES-based methods (CMA-MEGA (ES) and ME-ES), making them the highest-performing ES-based methods in QD-RL. Compared to deep RL-based methods PGA-ME and CMA-MEGA (TD3, ES), the variants also tend to perform better or have no significant difference. In particular, both sep-CMA-MAE and LM-MA-MAE outperform PGA-ME on QD Ant and QD Hopper while having no significant difference in QD Half-Cheetah. While the variants underperform the deep RL-based methods on QD Walker, prior work [3] highlights the importance of deep RL in this task, as only algorithms with TD3 have performed well here. In short, these results partially support **H4**, showing that the variants often but not always outperform the baselines.

*H5:* H5 was not supported. We found no significant differences between the variants' runtimes, except sep-CMA-MAE was significantly faster than the other variants in QD Walker. This outcome may arise from the more complex hardware setup of this experiment. Compared to the single CPU used to run the optimization benchmarks, the evaluations here run on 100 CPUs across multiple nodes. Slight differences among the nodes may create runtime variance that obscures differences caused by the search distribution complexity.

*H6:* All CMA-MAE variants were significantly faster than PGA-ME and CMA-MEGA (TD3, ES) in all domains. The two deep RL-based algorithms took more than twice as long to run as the variants. While variance in compute nodes may have contributed to this difference as we believe it did in **H5**, we believe the majority of the difference stems from the internal algorithm runtime, specifically the aforementioned network training performed in the deep RL-based methods.

*Memory Usage:* To better understand resource requirements, we report the memory usage of each algorithm's internal components in Table III. Many algorithms have similar usage due to creating similarly sized components. For instance, in the CMA-MAE variants, CMA-MEGA (ES), ME-ES, and MAP-Elites, memory is dominated by the archive, with negligible space for components like emitters. Meanwhile, PGA-ME and CMA-MEGA (TD3, ES) require more memory to store their TD3 replay buffers.

### C. Ablation of Archive Learning Rate

We believe the soft archive and improvement ranking play a key role in the CMA-MAE variants' performance. Thus, we ablate this mechanism by varying the archive learning rate $\alpha$ in sep-CMA-MAE. Table VII shows the result of varying $\alpha \in [0, 1]$; note that all experiments thus far used $\alpha = 0.001$. These results show that, similar to CMA-MAE in benchmark QD domains [8], performance (QD score) falls at the extreme values $\alpha = 0$ and $\alpha = 1$, when sep-CMA-MAE focuses entirely on objective optimization or archive exploration, respectively. In contrast, intermediate values blend both aspects to achieve high performance.

## VI. Discussion and Conclusion

We create variants of CMA-MAE that scale to neural network controllers for robotic locomotion by replacing CMA-MAE's CMA-ES component with efficient approximations. Our results on optimization benchmarks (Section IV) help distinguish the variants' properties, while our results on locomotion tasks (Section V) showcase the effectiveness of the variants compared to existing methods. Furthermore, compared to state-of-the-art deep RL-based methods, our variants bring attractive practical benefits:

1) The CMA-MAE variants are light on computation. PGA-ME and CMA-MEGA (TD3, ES) both train deep RL components with TD3, a lengthy process that significantly increases runtime as shown in the results of **H6**.

2) The CMA-MAE variants have very few hyperparameters since they depend on CMA-ES and its variants, which are designed to be parameterized by only an initial step size $\sigma$ and batch size λ. Hence, the CMA-MAE variants only require 5 hyperparameters ($\psi, \lambda, \sigma, \alpha, min_f$, see Algorithm 1). In contrast, deep RL-based methods require many more parameters: 18 for PGA-ME, 15 for CMA-MEGA (TD3, ES). Methods without deep RL require fewer hyperparameters: 5 for CMA-MEGA (ES), 6 for ME-ES, 2 for MAP-Elites.[2] However, our experiments show that such methods do not perform as well as the CMA-MAE variants.

We emphasize that our CMA-MAE variants are black-box methods that do not leverage the MDP structure of the QD-RL problem, making them suitable for settings beyond QD-RL. Hence, we envision future applications of our variants in areas such as manipulation [10] and scenario generation [11].

## References

[1] A. Cully, J. Clune, D. Tarapore, and J.-B. Mouret, "Robots that can adapt like animals," *Nature*, vol. 521, pp. 503–507, 2015.

[2] C. Colas, V. Madhavan, J. Huizinga, and J. Clune, "Scaling map-elites to deep neuroevolution," in *Proc. Genet. Evol. Comput. Conf.*, 2020, pp. 67–75.

[3] B. Tjanaka, M. C. Fontaine, J. Togelius, and S. Nikolaidis, "Approximating gradients for differentiable quality diversity in reinforcement learning," in *Proc. Genet. Evol. Comput. Conf.*, 2022, pp. 1102–1111.

[4] O. Nilsson and A. Cully, "Policy gradient assisted map-elites," in *Proc. Genet. Evol. Computation Conf.*, 2021, pp. 866–875.

[5] J. K. Pugh, L. B. Soros, and K. O. Stanley, "Quality diversity: A new frontier for evolutionary computation," *Front. Robot. AI*, vol. 3, 2016, Art. no. 40.

[6] T. Salimans, J. Ho, X. Chen, S. Sidor, and I. Sutskever, "Evolution strategies as a scalable alternative to reinforcement learning," 2017, *arXiv:1703.03864*.

[7] N. Hansen, "The CMA evolution strategy: A tutorial," 2016, *arXiv:1604.00772*.

[8] M. Fontaine and S. Nikolaidis, "Covariance matrix adaptation map-annealing," in *Proc. Genet. Evol. Comput. Conf.*, 2023, pp. 456–465.

[9] K. Varelas et al., "A Comparative study of large-scale variants of CMA-ES," in *Proc. Int. Conf. Parallel Problem Solving Nature*, 2018, pp. 3–15.

[10] A. Morel, Y. Kunimoto, A. Coninx, and S. Doncieux, "Automatic acquisition of a repertoire of diverse grasping trajectories through behavior shaping and novelty search," in *Proc. Int. Conf. Robot. Automat.*, 2022, pp. 755–761.

[11] V. Bhatt et al., "Deep surrogate assisted generation of environments," in *Proc. Adv. Neural Inf. Process. Syst.*, 2022, pp. 37762–37777.

[12] B. Tjanaka et al., "Pyribs: A bare-bones python library for quality diversity optimization," in *Proc. Genet. Evol. Comput. Conf.*, 2023, pp. 220–229.

[13] M. C. Fontaine and S. Nikolaidis, "Differentiable quality diversity," in *Proc. Adv. Neural Inf. Process. Syst.*, 2021, vol. 34, pp. 10040–10052.

[14] J. Mouret and J. Clune, "Illuminating search spaces by mapping elites," 2015, *arXiv:1504.04909*.

[15] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, 2nd ed. Cambridge, MA, USA: MIT Press, 2018.

[16] H.-G. Beyer and H.-P. Schwefel, "Evolution strategies: A comprehensive introduction," *Natural Comput.*, vol. 1, no. 1, pp. 3–52, Mar. 2002.

[17] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. 3rd Int. Conf. Learn. Representations*, 2015. [Online]. Available: https://dblp.org/rec/journals/corr/KingmaB14.html?view=bibtex

[18] N. Hansen and A. Ostermeier, "Completely derandomized self-adaptation in evolution strategies," *Evol. Comput.*, vol. 9, no. 2, pp. 159–195,2001.

[19] M. C. Fontaine, J. Togelius, S. Nikolaidis, and A. K. Hoover, "Covariance matrix adaptation for the rapid illumination of behavior space," in *Proc. Genet. Evol. Comput. Conf.*, 2020, pp. 94–102.

[20] S. Fujimoto, H. V. Hoof, and D. Meger, "Addressing function approximation error in actor-critic methods," in *Proc. 35th Int. Conf. Mach. Learn.*, 2018, pp. 1587–1596.

[21] V. Vassiliades and J.-B. Mouret, "Discovering the elite hypervolume by leveraging interspecies correlation," in *Proc. Genet. Evol. Comput. Conf.*, 2018, pp. 149–156.

[22] I. Loshchilov, T. Glasmachers, and H.-G. Beyer, "Large scale black-box optimization by limited-memory matrix adaptation," *IEEE Trans. Evol. Computation*, vol. 23, no. 2, pp. 353–358, Apr. 2019.

[23] R. Ros and N. Hansen, "A simple modification in CMA-ES achieving linear time and space complexity," in *Proc. Int. Conf. Parallel Problem Solving Nature*, 2008, pp. 296–305.

[24] J. Lehman and K. O. Stanley, "Abandoning objectives: Evolution through the search for novelty alone," *Evol. Comput.*, vol. 19, no. 2, pp. 189–223, 2011.

[25] L. Grillotti and A. Cully, "Kheperax: A lightweight JAX-based robot control environment for benchmarking quality-diversity algorithms," in *Proc. Companion Conf. Genet. Evol. Comput.*, . 2023, pp. 2163–2165.

[26] O. Nilsson, "Qdgym," 2021. [Online]. Available: https://github.com/ollenilsson19/QDgym

[27] M. Flageat and A. Cully, "Fast and stable MAP-Elites in noisy domains using deep grids," in *Proc. Conf. Artif. Life*, 2020, pp. 273–282.

[28] M. Flageat, F. Chalumeau, and A. Cully, "Empirical analysis of PGA-MAP-elites for neuroevolution in uncertain domains," *ACM Trans. Evol. Learn. Optim.*, vol. 3, no. 1, pp. 1–32, 2023.

[2] These counts are based on prior listings [3] of hyperparameters.