Behavioral Foundation Models for Quality Diversity

Nazim Bendib
Sorbonne Université, ISIR
Paris, France
bendib@isir.upmc.fr

Nicolas Perrin-Gilbert
Sorbonne Université, ISIR
Paris, France
nicolas.perrin@isir.upmc.fr

Olivier Sigaud
Sorbonne Université, ISIR
Paris, France
olivier.sigaud@isir.upmc.fr

Abstract

Behavioral Foundation Models (BFMs) are an emerging paradigm in reinforcement learning, playing a role analogous to large language models in natural language processing: they have shown remarkable versatility, enabling zero-shot performance, fast imitation, and online adaptation, all by exploiting the structure of a latent space. In this work, we investigate whether the latent behavioral space induced by BFMs can serve as an effective search space to discover large repertoires of behaviorally diverse and high-performing policies through Quality-Diversity (QD) methods. While QD methods generally search directly in high-dimensional policy parameter space, in this paper, we present BFM-QD, a framework that performs QD search in the compact latent space of a BFM. We further show that the BFM-QD framework provides a closed-form, gradient-free policy improvement operator that approximates a policy gradient update, but requires no critic training and no backpropagation. Across continuous-control benchmarks spanning dense locomotion, sparse navigation, and contact-rich manipulation, BFM-QD consistently outperforms parameter-space baselines, with particularly stark gains in sparse and deceptive settings, where all tested parameter-space QD methods collapse to near-zero performance. These results show the effectiveness of the BFM-QD framework, benefiting from the synergy between dimensionality reduction of the search space and offline pretraining from diverse behavioral data. This positions BFMs as a general-purpose backbone for QD optimization, extending their utility beyond zero-shot task solving to the discovery of diverse behavioral repertoires.

1 Introduction

The history of machine learning is marked by a recent and decisive shift: rather than training models from scratch for each new task, the field has converged on the paradigm of large-scale pretraining followed by downstream reuse. In natural language processing, this inflection point arrived with large language models: pretrained representations so rich that a single backbone could be adapted, prompted, or searched over to solve tasks that no individual fine-tuned model could reach alone. Reinforcement Learning (RL) appears to be approaching an analogous moment.

Two families are emerging as the RL analogue of this paradigm. On one hand, Vision-Language-Action models (VLAs) [25] benefit from large-scale pretraining on expert demonstrations mainly for text-conditioned robotic manipulation tasks. On the other hand, Behavioral Foundation Models (BFMs) [3, 35, 34] take a complementary approach. Trained once on large offline non-expert datasets of diverse agent behavior, a BFM encodes a diverse behavioral manifold into a compact latent space

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

$\mathcal{Z} \subset \mathbb{R}^d$, such that varying a single continuous vector z is sufficient to instantiate a wide spectrum of motor skills, navigation strategies, and manipulation primitives, all without any additional training. This representational efficiency has enabled remarkable results in zero-shot task generalization [35, 36], fast imitation [30], and online adaptation [32], placing BFMs at the frontier of generalist agent design. Yet a striking limitation unites all existing work on BFMs: they all extract exactly one optimal behavior from the model. Given a task, the latent code z^* that maximizes expected return is inferred or optimized, and the resulting single policy is deployed. The vast behavioral structure embedded in its latent space, containing diverse, distinct, and potentially complementary behaviors, is underexploited. From this perspective, this paper asks a simple but consequential question: rather than extracting a single optimal behavior from a BFM, can we discover a large repertoire of diverse, high-performing behaviors from its latent space?

Quality-Diversity (QD) [26] is the natural framework to unlock this diversity: it discovers large repertoires of diverse, high-performing behaviors. Such repertoires matter because a single policy is optimal only under fixed assumptions. An archive enables, for example, damage recovery (selecting a policy that avoids a damaged leg without retraining), sim-to-real transfer (testing several candidates on hardware), and skill libraries for hierarchical control. Yet standard QD methods search directly in the policy parameter space $\Theta \subset \mathbb{R}^D$, which is high-dimensional, unstructured, and almost entirely composed of incoherent behaviors (random perturbations of network weights almost never produce rewarding behaviors). This is not a limitation of the QD objective, but of the search space it is applied to. The key insight is that in a well-trained BFM, the parameter space $\mathcal{Z}$ is behaviorally rich: points $z \in \mathcal{Z}$ in the latent space decode to behaviors that cover a wide spectrum of task-relevant features. So, by design, $\mathcal{Z}$ is exactly the kind of search space QD needs.

We propose **BFM-QD**, a framework that exploits this synergy by relocating QD search from the policy parameter space to the latent space of a pretrained BFM, providing an ideal search space for QD without modifying its objective. Across a benchmark spanning dense locomotion (Walker, HalfCheetah), long-horizon sparse navigation (AntMaze-medium), and contact-rich manipulation (Cube-single), BFM-QD consistently outperforms all parameter-space baselines. On easy and dense tasks, the margin is modest, but convergence is faster. On hard and sparse tasks, the gap is stark: all parameter-space methods fail, while BFM-QD maintains strong behavioral coverage and high performance. Together, these results position BFMs not merely as tools for single-task policy extraction, but as general-purpose backbones for QD, extending their utility from zero-shot and few-shot task solving to the generation of large repertoires of efficient policies.

Contributions: In this paper:

- We present BFM-QD, the first framework to use the latent space of a BFM as the search space for QD optimization.
- We show that BFMs provide a closed-form, gradient-free policy improvement operator that matches a policy-gradient variation operator, requiring no critic training and no backpropagation.
- Through an extensive benchmark spanning dense locomotion, sparse navigation, and contact-rich manipulation, we show that BFM-QD consistently outperforms all parameter-space baselines, with stark gains in sparse and deceptive settings where all tested parameter-space QD methods fail.
- We characterize two key properties of the BFM latent space: behavioral richness, where random latent vectors decode to diverse efficient behaviors, and a structure where interpolation between solutions produces meaningful offspring.
- We show that QD performance saturates early in BFM pretraining, well before zero-shot performance converges, revealing BFM-QD does not require a fully trained BFM.

2 Related Work

2.1 Behavioral Foundation Models

Behavioral Foundation Models are policies trained on large offline datasets of diverse behaviors, providing a pretrained latent space that can be adapted to new tasks without additional training. Successor Features [10, 3, 6] provide an early instance of this paradigm, decomposing the value function into task-independent state features and task-specific reward weights, enabling zero-shot

transfer across reward functions. Forward-Backward representations [35] extend this idea by jointly learning a forward map over successor measures and a backward map that projects reward functions into the latent space, enabling closed-form task inference and zero-shot policy retrieval. TD-JEPA [2] further advances this direction by leveraging latent-predictive representations that capture long-term, policy-conditioned dynamics from offline reward-free data. Beyond zero-shot task solving, BFMs have demonstrated broad utility as reusable behavioral backbones. They have been applied to fast imitation learning [30] using a small set of expert demonstrations, and to fast online adaptation [32], where the latent space enables rapid fine-tuning to new tasks with minimal environment interaction. More recently, BFMs have been scaled to humanoid robot control [34], demonstrating that the learned latent behavioral structure transfers to high-dimensional, contact-rich settings. Our work extends this line of research in a new direction: rather than extracting a single optimal behavior from the BFM, we use its latent space as the search space for quality-diversity optimization, discovering entire repertoires of diverse and high-performing behaviors.

2.2 Quality-Diversity algorithms

Quality-Diversity algorithms aim to discover large collections of diverse, high-performing solutions rather than a single optimum. MAP-Elites (ME) [26] is the canonical algorithm, maintaining an archive of elite solutions indexed by behavioral descriptors and evolving them through random mutations. CMA-ME [15] improves upon this by replacing Gaussian mutation with Covariance Matrix Adaptation (CMA-ES) [20], enabling more structured and efficient parameter-space search. However, these methods rely purely on undirected exploration and struggle in environments where random parameter perturbations rarely produce rewarding behaviors. To address this, policy gradient methods have been integrated into QD. PGA-ME [27] augments ME with TD3 [16] actor updates, using a learned critic to guide mutations toward higher-fitness regions. QDRL [24] similarly combines QD with deep RL, alternating between diversity-seeking and reward-maximizing updates. These approaches improve performance on dense-reward tasks but rely on a well-trained critic, which degrades under sparse or deceptive rewards. A complementary line of work augments the gradient-based variation operator with explicit descriptor conditioning. DCG-ME [12] enhances the policy gradient variation operator with a descriptor-conditioned critic, providing gradients that jointly optimize fitness and target descriptors. DCRL-ME [13] builds on this by additionally exploiting the descriptor-conditioned actor as a generative model, injecting diverse solutions directly into the offspring batch at each generation. Our work is complementary to these: rather than improving the descriptor or critic, we relocate the search space entirely to the latent space of a pretrained BFM. Another line of work searches learned latent spaces. Latent space illumination [14] runs QD in the latent space of a GAN to generate game levels; it is a content-generation task without policies or control. Policy Manifold Search [31] and DDE-Elites [17] encode archive solutions with an autoencoder over policy parameters, so the latent space is learned online from the archive. In contrast, we search the latent space of a policy-generating model that is pretrained offline without rewards.

3 Background

3.1 Behavioral Foundation Models

A Behavioral Foundation Model (BFM) is an agent pretrained on a reward-free offline dataset of environment transitions, such that it can produce optimal policies for a large class of reward functions specified at test time, without any additional training or planning. Most BFMs are built on top of the successor measure [5, 10] framework. The successor measure of a policy π at state-action (s, a) is the discounted distribution of future states:

$$M^\pi(X \mid s, a) := \sum_{t \geq 0} \gamma^t \Pr(s_t \in X \mid s_0 = s, a_0 = a, \pi), \quad \forall X \subset \mathcal{S}. \tag{1}$$

Successor measures disentangle environment dynamics from the reward function: for any reward r and policy π, the Q-function decomposes as $Q_r^\pi(s, a) = M^\pi r(s, a)$. Given a feature map $\phi : \mathcal{S} \to \mathbb{R}^d$, the corresponding successor features are $\psi^\pi(s, a) = \sum_{t \geq 0} \gamma^t \mathbb{E}[\phi(s_{t+1}) \mid s, a, \pi]$, which allow expressing the Q-function compactly as $Q_r^\pi(s, a) = \psi^\pi(s, a)^\top z$ for any reward $r_z(s) = \phi(s)^\top z$.

BFMs instantiate this framework by learning a family of policies $\{\pi_z\}_{z \in \mathcal{Z}}$ parameterized by a latent code $z \in \mathcal{Z} \subset \mathbb{R}^d$, alongside their successor features $\psi(s, a, z)$. At test time, given a reward function

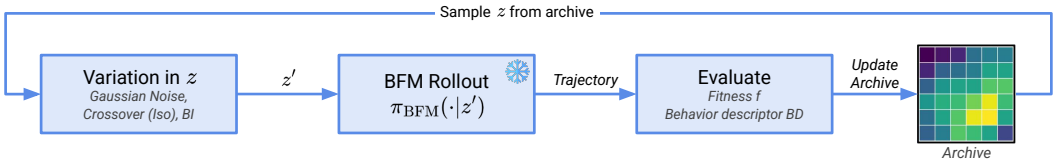


Figure 1: **Overview of BFM-QD.** A BFM is pretrained once on an offline dataset and then frozen. The loop is one QD generation: latent codes z are sampled from the archive, mutated into z', and evaluated by rolling out $\pi_{\text{BFM}}(\cdot|z')$. Offspring update the archive.

r, the optimal latent code is inferred by projecting r onto the learned features:

$$z_r = \arg\min_{z \in \mathbb{R}^d} \mathbb{E}_{s\sim\rho}\left[(r(s) - \phi(s)^\top z)^2\right] = \mathbb{E}_{s\sim\rho}[\phi(s)\phi(s)^\top]^{-1}\mathbb{E}_{s\sim\rho}[\phi(s)r(s)], \tag{2}$$

and the policy π_{z_r} is deployed zero-shot. In practice z_r is projected to a hypersphere of radius $\sqrt{d_z}$ with $d_z = \dim(\mathcal{Z})$.

Different BFMs differ in how they learn ϕ. Forward-Backward (FB) representations [35] jointly learn ϕ and ψ through a successor measure consistency loss, enabling closed-form task inference at test time. TD-JEPA [2] instead learns ϕ and ψ through a temporal-difference latent-predictive loss, training a policy-conditioned predictor that approximates successor features directly in latent space. Other BFMs are presented in Appendix B. In all cases, the result is a single pretrained model $\pi_{\text{BFM}}(.|z)$ whose latent space $\mathcal{Z}$ encodes a wide range of behaviors, all accessible by varying z.

3.2 Quality-Diversity algorithms

Quality-Diversity (QD) algorithms aim to discover a large collection of diverse, high-performing solutions rather than converging to a single optimum. A QD problem is defined by a fitness function $r : \mathcal{S} \to \mathbb{R}$, measuring how good a solution is, and a behavioral descriptor $\beta : \mathcal{S} \to \mathcal{B}$, a low-dimensional summary of how a solution behaves (e.g., the foot contact pattern of a walking robot). The descriptor space $\mathcal{B}$ is discretized into a finite grid of cells, and the goal is to find the best possible solution for every cell, that is, to fill the grid with a diverse repertoire of high-performing policies, each with a distinct behavior. Solutions are maintained in an archive $\mathcal{C}$ storing at most one elite per cell; a candidate replaces the existing elite only if it achieves higher fitness in the same cell.

ME [26] is the canonical QD algorithm: parents are sampled from the archive, perturbed by a variation operator, evaluated, and inserted if they improve their cell. The typical variation operator is a Gaussian mutation, but it can be replaced with more sophisticated methods, such as covariance-based adaptation [15] or policy gradient updates that leverage a learned critic to guide mutations toward higher-fitness regions [27, 24, 12, 13]. See Appendix C for more details. Each generation thus repeats four steps: select parents from the archive, vary them, evaluate fitness and descriptor with a rollout, and insert offspring into their cells (Algorithm 1).

The performance of QD methods is measured by three metrics: **coverage**, the fraction of cells populated in the archive; the **QD-score**, the sum of fitness values across all filled cells, which jointly captures whether the repertoire is both diverse and high-performing; and **maximum fitness**, the fitness of the single best solution found, measuring peak performance regardless of diversity.

4 Method

Standard QD methods parameterize policies as neural networks $\theta \in \Theta$ and mutate directly in Θ. We instead leverage a pretrained BFM, specifically its pretrained actor $\pi_{\text{BFM}} : \mathcal{S} \times \mathcal{Z} \to \mathcal{A}$, which produces diverse behaviors by just varying $z \in \mathcal{Z}$. The BFM is trained once on an offline dataset and then kept frozen throughout all QD runs.

4.1 Latent Space Quality-Diversity

The central observation of our work is that $\mathcal{Z}$ is a far more efficient search space for QD than Θ: it is low-dimensional ($d_z \ll |\theta|$) and behaviorally rich. We therefore propose to run QD directly in $\mathcal{Z}$. The archive stores latent codes z rather than policy weights θ, and all variation operators act on z.

Each candidate is evaluated by rolling out $\pi_{\text{BFM}}(\cdot \mid z)$ in the environment and computing its fitness and behavioral descriptor. The QD loop is otherwise standard: solutions compete for cells in the archive based on fitness, and the archive is used to seed the next generation of mutations. Since the BFM is trained with a normalization constraint, we re-project every mutated candidate back onto the sphere after perturbation. See Figure 1 for an overview. See Algorithm 2 for pseudocode.

This constitutes our framework, which we refer to as **BFM-QD**. A key property of the BFM-QD framework is that it is modular by design, meaning any BFM can be paired with any QD optimizer, yielding variants such as FB-ME, FB-CMA-ME, TD-JEPA-ME, or TD-JEPA-CMA-ME.

4.2 Closed-Form Policy Improvement via Backward Inference

Standard QD methods rely on gradient-free exploration, which is sample-inefficient. Recent work has addressed this by integrating policy gradient methods into QD [27, 13]: a critic is trained on the fitness function, and policies are updated to maximize it, improving solution quality while the archive maintains diversity. However, this requires training and maintaining a critic online, and doing multiple backpropagations to update the solutions.

BFMs enable sidestepping this limitation. A key advantage of BFMs is their zero-shot capabilities: Given a fitness function and a set of evaluated transitions, the latent code z^* corresponding to the optimal policy for the observed reward function can be estimated in closed form following Eq 2. This inference requires no gradient computation, no critic training, and no additional environment interactions beyond those already collected during archive evaluation. We use this inference, which we call the **Backward Inference (BI)** operator, as a directed variation operator. For each solution z selected from the archive, we then produce an offspring by interpolating toward this global z^*:

$$z' \leftarrow (1-\alpha)\cdot z + \alpha \cdot z^*, \tag{3}$$

where $\alpha \in]0,1[$ controls the step size toward the inferred optimum. z' is then normalized and projected back onto the BFM latent sphere. Note that we do not set $z' \leftarrow z^*$ directly: doing so for every solution would collapse the entire archive toward a single behavior, destroying the diversity that QD is designed to maintain. Instead, we push each solution individually toward higher fitness while preserving its distinctiveness. At each generation, we (i) sample state–reward pairs $(s, r(s))$ from the replay buffer, which stores all past evaluation trajectories; (ii) compute z^* with Eq. 2; and (iii) for each parent z selected from the archive, produce an offspring with Eq. 3 and project it onto the sphere. Following PGA-ME, half of the offspring are produced by BI ($\alpha = 0.02$) and half by Gaussian mutation ($\sigma = 1$). A pseudo-algorithm is presented in Algorithm 3.

Interestingly, this interpolation is a Newton step on a local quadratic surrogate of the policy improvement objective, moving z toward z^* (Appendix A). The step itself is generic; the BFM makes it practical by providing z^* in closed form, with no critic training and no backpropagation. The approximation error is proportional to the reward projection residual.

5 Experiments

We evaluate BFM-QD on a suite of continuous control tasks spanning a spectrum of reward densities, from dense locomotion to sparse navigation and contact-rich manipulation. Our experiments are designed to answer four questions: **(1)** Does BFM-QD outperform standard QD baselines across task types and reward densities? **(2)** What properties of the FB latent space drive BFM-QD's performance? **(3)** Can Backward Inference in BFMs replace policy gradient updates? **(4)** Does BFM-QD require a large pretraining budget to be effective?

5.1 Experimental Setup

Environments. We evaluate on 18 tasks spanning four environments and a spectrum of difficulty levels. **Walker2D_uni** and **HalfCheetah_uni** are standard continuous control benchmarks from QDax [8], representing dense, simple locomotion, with pretraining data collected via Random Network Distillation (RND) [7]. **AntMaze-medium** from OGBench [28] uses the `antmaze-medium-explore` dataset collected by random policies and introduces long-horizon navigation with both dense and sparse rewards. **Cube-single** from OGBench uses the `cube-single-noisy` dataset, consisting of

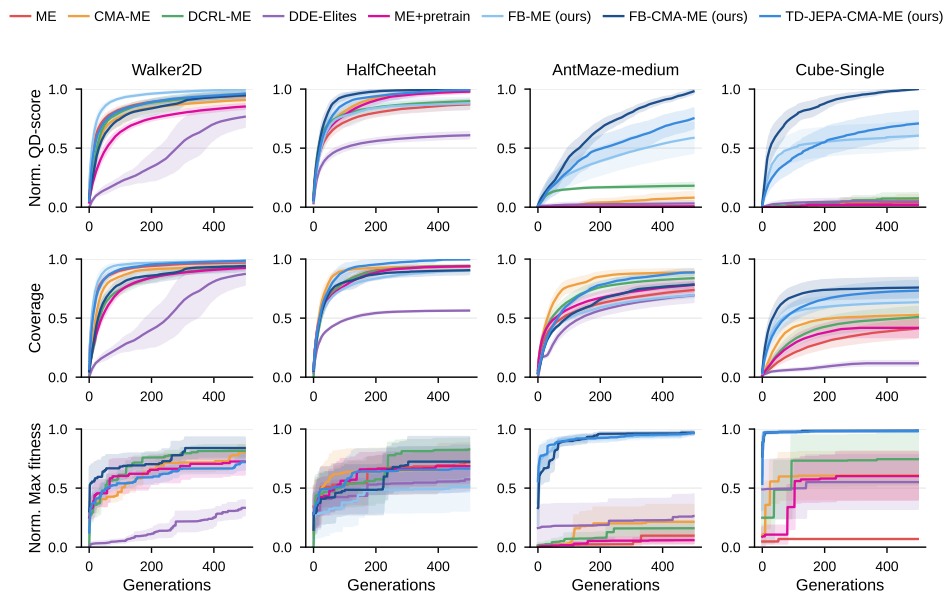


Figure 2: **Normalized QD-score, coverage, and normalized max fitness across all environments.** BFM-QD variants consistently match or outperform all baselines, especially on harder tasks.

noisy trajectories, and introduces manipulation under sparse rewards (hardest setting). Full details of environments, behavioral descriptors, and fitness functions are provided in Appendix D.1.

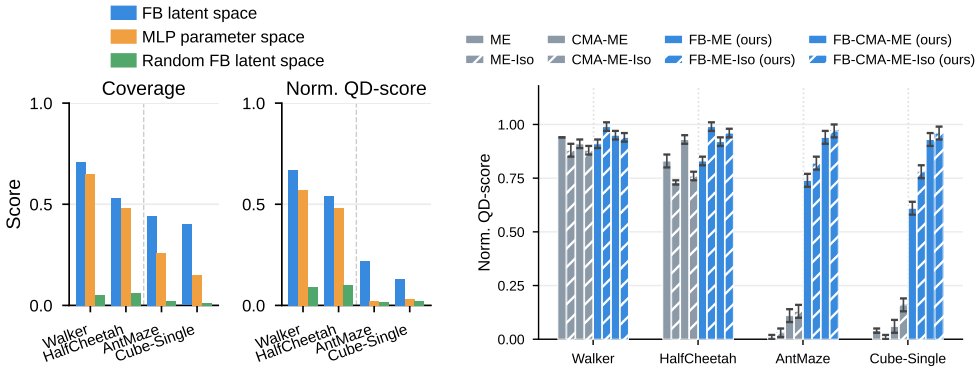
Baselines. We compare BFM-QD against five parameter-space baselines. **ME** [26] is the standard QD algorithm, operating in MLP parameter space with Gaussian mutation. **CMA-ME** [15] augments ME with CMA-ES. **DCRL-ME** [13] combines ME with TD3 policy gradient using a descriptor-conditioned policy. **DDE-Elites** [17] iteratively trains a Variational Auto-Encoder (VAE) [29] on the archive solutions to learn a compact latent space for more sample-efficient search. This baseline isolates the effect of searching in a low-dimensional space. **ME+Pretrain** initializes MLP policies via offline TD3 trained on the same dataset as the BFM, using the task fitness function as a reward. This baseline isolates the effect of pretraining on diverse data. All baselines use the same MLP architecture for a fair comparison. For more details, see Appendix D.2.

BFM-QD Variants. We instantiate three BFM-QD variants: **FB-ME** pairs FB with ME using Gaussian mutation; **FB-CMA-ME** pairs FB with CMA-ME; **TD-JEPA-CMA-ME** pairs TD-JEPA with CMA-ME. Other variants of BFM-QD are presented in Appendix E.4.

Implementation Details. The FB model is pretrained once per environment on the offline dataset and kept frozen throughout all QD runs (See Appendix E.11 for reported compute time). The latent space has dimension $d_z = 50$. The archive is a 50×50 grid. Each run consists of 500 generations of 400 parallel rollouts, for 200k evaluations in total. A rollout is one full episode of 500 steps (Walker, HalfCheetah) or 1000 steps (AntMaze, Cube-single), so a run collects 100M or 200M transitions, respectively (200k or 400k per generation). Since each environment contains multiple tasks with different reward scales, we normalize QD-scores using min-max normalization before aggregating across tasks (see Appendix D.1 for details). For parameter-based methods, we use MLPs of the shape (state_dim, 128, 128, action_dim). All results are averaged over 5 random seeds unless stated otherwise. Standard deviations are reported.

5.2 Main Results

Figure 2 reports the QD-score, coverage, and maximum fitness across all methods and environments. BFM-QD variants consistently achieve the highest QD-score across all tasks. On dense locomotion tasks (Walker, HalfCheetah), the margin over parameter-space baselines is modest, but BFM-QD variants converge faster, confirming that the behavioral prior accelerates search even when the task is easy. Notably, BFM-QD variants achieve this despite operating in a far lower-dimensional space ($d_z = 50$ versus $\sim 20k$ parameters), demonstrating that the BFM latent space is sufficiently expressive to reproduce near-optimal behaviors across a wide range of fitness functions.



(a) **Random sampling.** Coverage and QD-score of 10,000 random FB latent samples vs. random MLP samples.

(b) **Interpolation operator.** QD-score for ME and CMA-ME, with and without the isoline operator, in parameter space and FB latent space.

Figure 3: **The FB latent space is expressive and behaviorally rich, and supports interpolation between behaviors.** (a) Random FB latent codes yield higher coverage and QD-score compared to random MLP policies. (b) Interpolation between solutions is only beneficial in the FB latent space, not in parameter space.

The advantage of BFM-QD grows substantially as task difficulty increases. On AntMaze-medium and Cube-single, all parameter-space baselines collapse to near-zero QD-score and low coverage, while BFM-QD variants maintain high performance across all metrics. Crucially, BFM-QD variants also achieve substantially higher maximum fitness on hard tasks. Two additional comparisons sharpen this finding. **DDE-Elites**, which also searches in a compact latent space, fails to match the BFM-QD performance, demonstrating that dimensionality reduction alone is not sufficient: what matters is the behavioral richness of the latent space, not its dimensionality. Similarly, **ME+Pretrain**, which benefits from offline pretraining, also fails, demonstrating that the benefit of BFM-QD is not simply a matter of pretraining on diverse data, but of searching in the structured latent space the BFM provides. Two further baselines that combine offline pretraining with latent-space search (DDE-Elites+Pretrain with TD3 or BC policies) and Policy Manifold Search also fall short of FB-ME (Appendix E.7).

Takeaway: Searching in the BFM latent space consistently outperforms searching in the parameter space, with stark gains in sparse and deceptive settings. The gains do not come from dimensionality reduction or pretraining alone, but from the compact behavioral representation of the BFM.

5.3 Expressivity of the latent space

We investigate two properties of the FB latent space that underpin the success of BFM-QD: the expressivity of the learned latent space and its smooth structure, where linear interpolations between solutions produce behaviorally intermediate policies.

5.3.1 The latent space is behaviorally rich by construction

To isolate the contribution of the behavioral prior from the QD optimizer, we remove all optimization entirely and evaluate the quality of random sampling alone.

Experiment We sample $N = 10{,}000$ random latent vectors from **(1)** a trained FB latent space, **(2)** the MLP parameter space, and **(3)** an untrained FB latent space (same architecture, random weights) to further isolate the contribution of pretraining from that of architecture. We report the resulting coverage and QD-score for each environment. No optimization is performed in any condition.

Results Results are shown in Figure 3a. Even without optimization, random FB samples already achieve higher coverage and QD-score than random MLP samples across all environments, with the gap growing monotonically with task difficulty (AntMaze-medium and Cube-single), where random MLP samples produce near-zero QD-score (see Appendix E.3 for archive visualization). Crucially, the untrained FB model performs significantly worse than the trained FB, confirming that

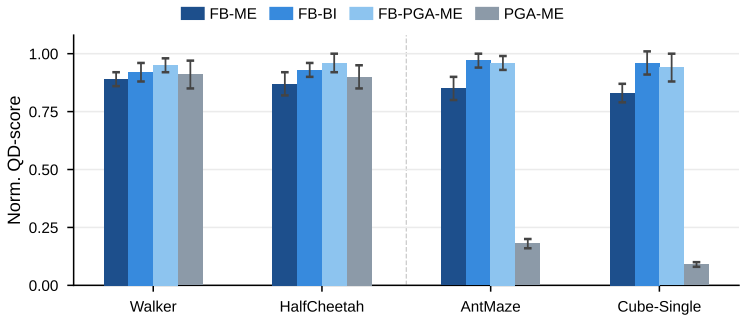


Figure 4: **The BFM Backward Inference (BI) matches a policy gradient update.** FB-BI (backward inference) matches FB-PGA-ME across all environments, while PGA-ME fails on hard tasks.

this behavioral richness stems specifically from offline pretraining, not from the architecture or the geometry of the latent sphere.

Takeaway: Through pretraining, the FB latent space naturally produces behaviors that cover a wide spectrum of task-relevant features.

5.3.2 The latent space structure makes interpolation meaningful

A key property of the BFM latent space is its geometric smoothness: nearby latent codes induce behaviorally similar policies. We investigate whether this smoothness is sufficient to make interpolation between archive solutions semantically meaningful, enabling a class of efficient crossover-based variation operators. By contrast, such operators can be destructive in parameter space as the interpolant typically collapses to an incoherent solution [11]. We introduce an isoline operator (Iso) that, given two parent solutions x_1, x_2 from the archive, produces an offspring by linear interpolation:

$$x_{\text{offspring}} = (1-\alpha)\, x_1 + \alpha\, x_2, \quad \alpha \sim \mathcal{U}[0,1].$$

A pseudo-algorithm is presented in Algorithm 4.

Experiment We compare ME and CMA-ME with and without the interpolation operator, both in parameter space (ME-Iso, CMA-ME-Iso) and the FB latent space (FB-ME-Iso, FB-CMA-ME-Iso), isolating whether the gain comes from the operator itself or the smoothness of the latent space.

Results Results are shown in Figure 3b. Adding the isoline operator to parameter-space methods (ME-Iso, CMA-ME-Iso) yields no consistent improvement, confirming that interpolation in parameter space can be destructive. In contrast, FB-ME-Iso and FB-CMA-ME-Iso consistently improve over their non-Iso counterparts, especially on AntMaze-medium and Cube-single. This extra gain can be attributed to the structure of the latent space: interpolated codes produce behaviorally intermediate policies, enabling a better exploration of the descriptor space that random mutation cannot replicate.

Takeaway: Interpolation between solutions is only beneficial in the BFM latent space due to its structure and smoothness, unlike in the neural-network parameter space.

5.4 Can Backward Inference in BFMs replace policy gradient updates?

A natural question is whether gradient-based variation operators can further improve BFM-QD. PGA-ME [27] augments ME with TD3 policy gradient updates, using a learned critic to guide mutations toward higher-fitness regions. However, BFMs and typically FB provide a closed-form estimate of the task vector that best explains an observed reward signal (Equation 2), requiring no critic training and no additional environment interactions. We investigate whether this zero-shot inference approach can substitute for a full policy gradient update.

Experiment We compare three methods: **PGA-ME**, which runs TD3 policy gradient updates in parameter space; **FB-PGA-ME**, which runs the same gradient updates but in the FB latent space; and **FB-BI**, which replaces gradient updates entirely with the Backward Inference (BI) operator (Equation 3). See Appendix D.2 for more details.

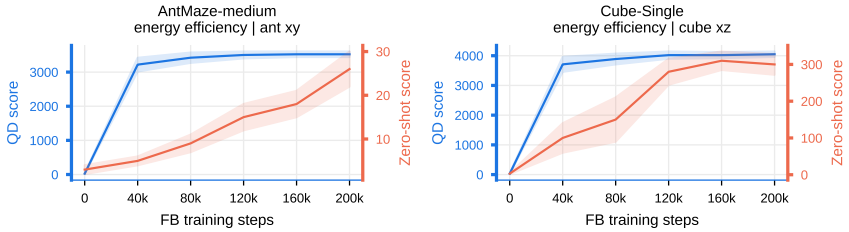


Figure 5: **BFM-QD does not require a fully trained BFM.** QD-score (blue) and zero-shot performance (red) as a function of FB training budget on AntMaze-medium and Cube-single

Results Results are shown in Figure 4. PGA-ME matches FB-PGA-ME and FB-BI methods on locomotion tasks but drastically fails on AntMaze-medium and Cube-single, where the reward signal is too deceptive and sparse to train an accurate critic. FB-BI matches FB-PGA-ME across all environments, achieving the same QD-score without any critic training. This shows that closed-form backward inference can substitute for a learned critic.

Takeaway: The BFM Backward Inference (BI) operator matches a policy-gradient variation operator, without critic training or backpropagation.

5.5 BFM-QD does not require a large pretraining budget

We investigate whether BFM-QD requires a large BFM pretraining budget to be effective for QD search. A priori, one might expect that QD performance follows zero-shot performance and thus requires a fully trained BFM. We test whether this holds by measuring QD performance and zero-shot performance at different stages of BFM training.

Experiment We train FB and extract checkpoints every 40k steps. At each checkpoint, we run FB-ME and report the resulting QD-score and the zero-shot performance of the checkpoint on the same task. We do this for (`energy efficiency |ant xy`) from AntMaze-medium and (`energy efficiency |cube xz`) from Cube-single.

Results Results are shown in Figure 5. QD performance rises sharply in the first 40k training steps and plateaus early, well before zero-shot performance converges. In both environments, a partially trained BFM already enables strong QD performance, while zero-shot performance continues to improve long after the QD-score has saturated. This decoupling confirms that the latent space acquires sufficient behavioral structure for QD search early in training, even before it can reliably solve individual tasks zero-shot.

We distinguish two properties of the BFM latent space: behavioral diversity, the repertoire of distinct behaviors it contains, and value precision, the accuracy of (ϕ, ψ) for computing z_r in closed form (Equation 2). Zero-shot solving requires both: diversity provides good candidates, and precision selects the best one. QD search requires only diversity. We attribute the decoupling in Figure 5 to the two properties emerging at different times: diversity is acquired early in pretraining, while value precision keeps improving later.

Takeaway: A partially trained BFM already provides a latent space rich enough for strong QD performance, amortizing pretraining cost further.

5.6 Additional Analyses

Beyond the main experiments, we conduct seven additional analyses in Appendix E, whose findings we summarize here:

1. **The choice of a BFM matters at scale (E.4):** Across five BFMs (Laplacian [37], BYOLγ [22], FB, BTD-FB [4], and TD-JEPA), zero-shot performance is a reliable proxy for QD performance; the gap between strong and weak BFMs widens sharply on hard tasks, making BFM quality a critical design decision in sparse settings, while weaker BFMs can still perform well on easy tasks.

2. **Diversity aids optimization, not just coverage (E.5):** FB-CMA-ME outperforms a single-objective CMA-ES [19] in the same latent space even on maximum fitness, demonstrating that maintaining a diverse population acts as an implicit exploration bonus that prevents premature convergence to local optima in the latent space.
3. **Exploration data dominates dataset quality (E.6):** Random exploration datasets consistently outperform expert datasets for BFM pretraining, even when spatially restricted, because limited coverage in the behavioral space in the offline data results in bounded latent space diversity.
4. **Pretraining and latent-space search are not sufficient (E.7):** DDE-Elites initialized with policies pretrained offline on the BFM dataset (TD3 or behavioral cloning) and Policy Manifold Search do not match FB-ME, showing that the gains come from the structure of the BFM latent space and not from combining pretraining with a learned latent space.
5. **Backward Inference outperforms sampling around z^* (E.8):** Sampling offspring around the inferred optimum confines the search to a small region of the latent space, whereas BI moves each parent individually toward z^* and preserves diversity.
6. **QD-collected data is a worse pretraining source than RND (E.9):** A BFM pretrained on data collected by a QD search performs worse than one pretrained on RND data, because QD refines solutions near existing elites instead of covering new transitions.
7. **Backward Inference is robust to its step size (E.10):** FB-BI is stable for $\alpha \in [0.005, 0.1]$ and degrades for larger steps, which pull offspring toward the single point z^* and collapse diversity.

6 Limitations

BFM-QD requires a behaviorally diverse offline dataset and a sufficiently pretrained BFM before any QD search can begin, though Section 5.5 shows that full convergence of the BFM is not necessary. Besides, a single BFM can be reused across all subsequent tasks and QD runs in the same environment, amortizing this cost (see Appendix E.11 for training time). Furthermore, performance degrades with narrower datasets (Appendix E.6) and with data collected by QD instead of RND (Appendix E.9), as the diversity of the latent space is bounded by the diversity of the training data. A deeper limitation is that the frozen BFM imposes a hard ceiling on behavioral expressivity: novel behaviors outside the pretrained latent space remain inaccessible. Finetuning the BFM during the QD run is a promising direction we leave to future work. Moreover, each BFM is trained for a single environment, so environment-agnostic QD remains open. Finally, as in standard QD, we assume the reward and behavioral descriptor are given; BFM-QD can only produce behaviors captured by the learned features $\phi(s)$, and we do not measure how far the discovered behaviors lie outside the pretraining data.

7 Conclusion

We proposed BFM-QD, a framework that replaces parameter-space exploration in QD search with exploration in the latent space of a pretrained BFM. BFM-QD sidesteps the fundamental bottleneck of standard QD methods by searching in a compact, behaviorally dense latent space, consistently outperforming parameter-space baselines across dense locomotion, sparse navigation, and contact-rich manipulation, with stark gains in sparse and deceptive settings where all tested parameter-space methods collapse. These results establish BFM latent space exploration as a principled foundation for the next generation of QD algorithms, and position BFMs as general-purpose backbones for diversity-seeking optimization beyond their original use case of single-task policy extraction.

Acknowledgments

Experiments presented in this paper were carried out using the HPC resources of IDRIS under the allocation 2025-[AD011016374] made by GENCI. This work was supported by the Sorbonne Center for Artificial Intelligence (SCAI)

Competing interests. We declare no competing interests.

References

[1] Peter Auer, Nicolo Cesa-Bianchi, and Paul Fischer. Finite-time analysis of the multiarmed bandit problem. Machine learning, 47(2):235–256, 2002.

[2] Marco Bagatella, Matteo Pirotta, Ahmed Touati, Alessandro Lazaric, and Andrea Tirinzoni. TD-JEPA: Latent-predictive representations for zero-shot reinforcement learning. arXiv preprint arXiv:2510.00739, 2025.

[3] André Barreto, Will Dabney, Rémi Munos, Jonathan J Hunt, Tom Schaul, Hado P Van Hasselt, and David Silver. Successor features for transfer in reinforcement learning. Advances in neural information processing systems, 30, 2017.

[4] Nazim Bendib, Nicolas Perrin-Gilbert, and Olivier Sigaud. Improving zero-shot offline RL via behavioral task sampling. arXiv preprint arXiv:2604.25496, 2026.

[5] Léonard Blier, Corentin Tallec, and Yann Ollivier. Learning successor states and goal-dependent values: A mathematical viewpoint. arXiv preprint arXiv:2101.07123, 2021.

[6] Diana Borsa, André Barreto, John Quan, Daniel Mankowitz, Rémi Munos, Hado Van Hasselt, David Silver, and Tom Schaul. Universal successor features approximators. arXiv preprint arXiv:1812.07626, 2018.

[7] Yuri Burda, Harrison Edwards, Amos Storkey, and Oleg Klimov. Exploration by random network distillation. arXiv preprint arXiv:1810.12894, 2018.

[8] Felix Chalumeau, Bryan Lim, Raphael Boige, Maxime Allard, Luca Grillotti, Manon Flageat, Valentin Macé, Guillaume Richard, Arthur Flajolet, Thomas Pierrot, et al. QDAX: A library for quality-diversity and population-based algorithms with hardware acceleration. Journal of Machine Learning Research, 25(108):1–16, 2024.

[9] Shuangshuang Chen and Wei Guo. Auto-encoders in deep learning: A review with new perspectives. Mathematics, 11(8):1777, 2023.

[10] Peter Dayan. Improving generalization for temporal difference learning: The successor representation. Neural computation, 5(4):613–624, 1993.

[11] Rahim Entezari, Hanie Sedghi, Olga Saukh, and Behnam Neyshabur. The role of permutation invariance in linear mode connectivity of neural networks. arXiv preprint arXiv:2110.06296, 2021.

[12] Maxence Faldor, Félix Chalumeau, Manon Flageat, and Antoine Cully. Map-elites with descriptor-conditioned gradients and archive distillation into a single policy. In Proceedings of the Genetic and Evolutionary Computation Conference, pages 138–146, 2023.

[13] Maxence Faldor, Félix Chalumeau, Manon Flageat, and Antoine Cully. Synergizing quality-diversity with descriptor-conditioned reinforcement learning. ACM Transactions on Evolutionary Learning, 5(1):1–35, 2025.

[14] Matthew C Fontaine, Ruilin Liu, Ahmed Khalifa, Jignesh Modi, Julian Togelius, Amy K Hoover, and Stefanos Nikolaidis. Illuminating mario scenes in the latent space of a generative adversarial network. In Proceedings of the AAAI Conference on Artificial Intelligence, volume 35, pages 5922–5930, 2021.

[15] Matthew C Fontaine, Julian Togelius, Stefanos Nikolaidis, and Amy K Hoover. Covariance matrix adaptation for the rapid illumination of behavior space. In Proceedings of the 2020 genetic and evolutionary computation conference, pages 94–102, 2020.

[16] 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.

[17] Adam Gaier, Alexander Asteroth, and Jean-Baptiste Mouret. Discovering representations for black-box optimization. In Proceedings of the 2020 Genetic and Evolutionary Computation Conference, pages 103–111, 2020.

[18] Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Guo, Mohammad Gheshlaghi Azar, et al. Bootstrap your own latent: A new approach to self-supervised learning. Advances in neural information processing systems, 33:21271–21284, 2020.

[19] Nikolaus Hansen. The CMA evolution strategy: A tutorial. arXiv preprint arXiv:1604.00772, 2016.

[20] Nikolaus Hansen, Sibylle D Müller, and Petros Koumoutsakos. Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (CMA-ES). Evolutionary computation, 11(1):1–18, 2003.

[21] Michael Laskin, Denis Yarats, Hao Liu, Kimin Lee, Albert Zhan, Kevin Lu, Catherine Cang, Lerrel Pinto, and Pieter Abbeel. URLB: Unsupervised reinforcement learning benchmark. arXiv preprint arXiv:2110.15191, 2021.

[22] Daniel Lawson, Adriana Hugessen, Charlotte Cloutier, Glen Berseth, and Khimya Khetarpal. Self-predictive representations for combinatorial generalization in behavioral cloning. arXiv preprint arXiv:2506.10137, 2025.

[23] Yann LeCun et al. A path towards autonomous machine intelligence version 0.9. 2, 2022-06-27. Open Review, 62(1):1–62, 2022.

[24] Bryan Lim, Manon Flageat, and Antoine Cully. Understanding the synergies between quality-diversity and deep reinforcement learning. In Proceedings of the Genetic and Evolutionary Computation Conference, pages 1212–1220, 2023.

[25] Yueen Ma, Zixing Song, Yuzheng Zhuang, Jianye Hao, and Irwin King. A survey on vision-language-action models for embodied ai. arXiv preprint arXiv:2405.14093, 2024.

[26] Jean-Baptiste Mouret and Jeff Clune. Illuminating search spaces by mapping elites. arXiv preprint arXiv:1504.04909, 2015.

[27] Olle Nilsson and Antoine Cully. Policy gradient assisted map-elites. In Proceedings of the Genetic and Evolutionary Computation Conference, pages 866–875, 2021.

[28] Seohong Park, Kevin Frans, Benjamin Eysenbach, and Sergey Levine. Ogbench: Benchmarking offline goal-conditioned RL. arXiv preprint arXiv:2410.20092, 2024.

[29] Lucas Pinheiro Cinelli, Matheus Araújo Marins, Eduardo Antúnio Barros da Silva, and Sérgio Lima Netto. Variational autoencoder. In Variational methods for machine learning with applications to deep networks, pages 111–149. Springer, 2021.

[30] Matteo Pirotta, Andrea Tirinzoni, Ahmed Touati, Alessandro Lazaric, and Yann Ollivier. Fast imitation via behavior foundation models. In NeurIPS 2023 Foundation Models for Decision Making Workshop, 2023.

[31] Nemanja Rakicevic, Antoine Cully, and Petar Kormushev. Policy manifold search: Exploring the manifold hypothesis for diversity-based neuroevolution. In Genetic and Evolutionary Computation Conference, 2021.

[32] Harshit Sikchi, Andrea Tirinzoni, Ahmed Touati, Yingchen Xu, Anssi Kanervisto, Scott Niekum, Amy Zhang, Alessandro Lazaric, and Matteo Pirotta. Fast adaptation with behavioral foundation models. arXiv preprint arXiv:2504.07896, 2025.

[33] Richard S Sutton and Andrew G Barto. Reinforcement learning: An introduction, volume 1. MIT press Cambridge, 1998.

[34] Andrea Tirinzoni, Ahmed Touati, Jesse Farebrother, Mateusz Guzek, Anssi Kanervisto, Yingchen Xu, Alessandro Lazaric, and Matteo Pirotta. Zero-shot whole-body humanoid control via behavioral foundation models. arXiv preprint arXiv:2504.11054, 2025.

[35] Ahmed Touati and Yann Ollivier. Learning one representation to optimize all rewards. Advances in Neural Information Processing Systems, 34:13–23, 2021.

[36] Ahmed Touati, Jérémy Rapin, and Yann Ollivier. Does zero-shot reinforcement learning exist? arXiv preprint arXiv:2209.14935, 2022.

[37] Yifan Wu, George Tucker, and Ofir Nachum. The laplacian in RL: Learning representations with efficient approximations. arXiv preprint arXiv:1810.04586, 2018.

Appendix Table of Contents

A Proof of the Backward Inference Operator **15**

B Background on Behavioral Foundation Models **17**

B.1 Successor Features 17

B.2 Forward-Backward Representations 18

B.3 TD-JEPA 19

B.4 BTD-FB 19

C Background on Quality-Diversity Methods **20**

C.1 MAP-Elites 20

C.2 CMA-ME 20

C.3 PGA-ME 20

C.4 DCRL-ME 21

C.5 DDE-Elites 21

C.6 ME+Pretrain 21

D Experimental Details **22**

D.1 Environments and Tasks 22

D.2 Baselines Implementation Details 23

D.3 BFM Implementation Details 26

E Additional Results **31**

E.1 Expanded main results 31

E.2 Archive Visualization 32

E.3 The latent space is behaviorally rich by construction 32

E.4 Effect of the Behavioral Foundation Model 33

E.5 Diversity as an Optimization Strategy 33

E.6 Sensitivity to the Offline Dataset Quality 34

E.7 Additional Latent-Space Search Baselines 35

E.8 Backward Inference vs. Sampling Around the Inferred Optimum 36

E.9 Pretraining Data Collected by QD 36

E.10 Sensitivity to the Backward Inference Step Size 36

E.11 Compute Resources 37

E.12 Results Table 38

A Proof of the Backward Inference Operator

In the following, we denote $\pi_z(\cdot) := \pi_{\mathrm{BFM}}(\cdot \mid z)$ the policy instantiated by the BFM at latent code $z \in \mathcal{Z}$.Consider the policy improvement objective:

$$J(z) = \mathbb{E}_{s\sim\rho}\big[-Q^{\pi_z}(s, \pi_z(s))\big], \tag{4}$$

where $Q^{\pi_z}(s,a) = \mathbb{E}\big[\sum_{t\geq 0}\gamma^t r(s_t) \mid s_0 = s, a_0 = a, \pi_z\big]$ is the action-value function of policy π_z under reward r, and ρ is the mixture state distribution induced by the archive policies (equivalent to a replay buffer). A standard first-order policy gradient step on J takes the form:

$$z \leftarrow z - \alpha\, \nabla_z J(z). \tag{5}$$

We will show that the BI operator (Equation 3) actually approximates a second-order Newton step of J, which subsumes this update.

By construction of z^* (Equation 2), the task vector z^* is the least-squares projection of r onto the feature space of ϕ:

$$z^* = \arg\min_{z\in\mathbb{R}^{d_z}} \mathbb{E}_\rho\big[(r(s) - \phi(s)^\top z)^2\big], \tag{6}$$

so that $r(s) \approx \phi(s)^\top z^*$ for all s [1]. Substituting into the definition of Q^{π_z}:

$$Q^{\pi_z}(s,a) = \mathbb{E}\left[\sum_{t\geq 0}\gamma^t r(s_t)\,\middle|\, s, a, \pi_z\right] \tag{7}$$

$$\approx \mathbb{E}\left[\sum_{t\geq 0}\gamma^t \phi(s_t)^\top z^*\,\middle|\, s, a, \pi_z\right] \tag{8}$$

$$= \underbrace{\mathbb{E}\left[\sum_{t\geq 0}\gamma^t \phi(s_t)\,\middle|\, s, a, \pi_z\right]^\top}_{\psi^{\pi_z}(s,a)} z^* \;=\; \psi^{\pi_z}(s,a)^\top z^* \;=\; Q^{\pi_z}_{z^*}(s,a), \tag{9}$$

where $\psi^{\pi_z}(s,a)$ are the successor features of π_z [3]. The policy gradient objective therefore becomes:

$$J(z) \approx \mathbb{E}_{s\sim\rho}\big[-Q^{\pi_z}_{z^*}(s,\pi_z(s))\big] = \mathbb{E}_{s\sim\rho}\big[-\psi^{\pi_z}(s,\pi_z(s))^\top z^*\big]. \tag{10}$$

Denote this surrogate objective $\tilde{J}(z) := \mathbb{E}_{s\sim\rho}\big[-\psi^{\pi_z}(s,\pi_z(s))^\top z^*\big]$, so that $J(z) \approx \tilde{J}(z)$ with error proportional to the reward projection residual $\|r - \phi^\top z^*\|_2$. By definition, π_{z^*} is the greedy policy with respect to $Q^{\pi_z}_{z^*}$:

$$\pi_{z^*}(s) = \arg\max_a \psi^{\pi_{z^*}}(s,a)^\top z^*, \tag{11}$$

and is therefore optimal for task z^*. By Bellman optimality [33], for any other policy π_z:

$$Q^{\pi_{z^*}}_{z^*}(s,a) \geq Q^{\pi_z}_{z^*}(s,a) \quad \forall\, z,\ s,\ a, \tag{12}$$

which implies $z^* = \arg\min_z \tilde{J}(z)$ and therefore $\nabla_z \tilde{J}\big|_{z=z^*} = 0$.

A second-order Taylor expansion of $\tilde{J}$ around z^* gives:

$$\tilde{J}(z) = \tilde{J}(z^*) + \underbrace{\nabla_z \tilde{J}\big|^\top_{z^*}}_{=0}(z - z^*) + \frac{1}{2}(z - z^*)^\top H\,(z - z^*) + O\big(\|z - z^*\|^3\big), \tag{13}$$

where $H = \nabla^2_z \tilde{J}\big|_{z^*} \succeq 0$. Differentiating gives $\nabla_z \tilde{J}(z) \approx H(z - z^*)$. Assuming a strict positive definiteness of the hessian ($H \succ 0$), a Newton step on $\tilde{J}$ yields:

$$z \leftarrow z - \alpha\, H^{-1}\nabla_z \tilde{J}(z) \tag{14}$$

$$= z - \alpha\, H^{-1}H(z - z^*) \tag{15}$$

$$= (1-\alpha)\, z + \alpha\, z^*. \tag{16}$$

[1] This approximation holds exactly when r lies in the linear span of ϕ, and approximately otherwise, with error proportional to the projection residual $\|r - \phi^\top z^*\|_\rho$.

Projecting back onto the latent sphere $\|z\| = \sqrt{d_z}$ recovers the BI operator (Equation 3). A Newton step on $\tilde{J}$ therefore approximates a Newton step on the true objective J, with residual error proportional to the reward projection quality $\|r - \phi^\top z^*\|_2$.

Remark. The BI operator constitutes a second-order policy gradient update in the BFM latent space. The critic gradient direction of PGA-ME is replaced by the analytically inferred z^*, obtained in closed form from archive rollouts with no critic training and no backpropagation. The proof extends to transition-dependent rewards $r(s, a)$ by replacing $\phi(s)$ with $\phi(s, a)$ throughout.

B Background on Behavioral Foundation Models

Behavioral foundation models (BFMs) aim to pretrain a single agent from reward-free interactions such that it can produce optimal policies for any downstream reward at test time, with no or minimal additional learning. The central challenge is to learn a compact representation of the environment's long-term dynamics that is simultaneously expressive enough to cover all tasks and structured enough to support efficient zero-shot policy extraction.

B.1 Successor Features

The standard Q-function conflates environment dynamics and task rewards, making it expensive to reuse across tasks. [3] propose Successor Features (SF) to decouple them. They assume that the reward on each transition can be linearly decomposed as

$$r(s, a, s') \;=\; \phi(s, a, s')^\top \mathbf{w}, \tag{17}$$

where $\phi : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}^d$ is a feature map shared across all tasks and $\mathbf{w} \in \mathbb{R}^d$ is a task-specific reward weight vector. Under this factorization, the action-value function of any policy π on task $\mathbf{w}$ decomposes as

$$Q^\pi_{\mathbf{w}}(s, a) \;=\; \underbrace{\mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t \phi(s_t, a_t, s_{t+1}) \;\middle|\; s_0 = s,\, a_0 = a,\, \pi\right]}_{\psi^\pi(s,a)}{}^\top \mathbf{w} \;=\; \psi^\pi(s, a)^\top \mathbf{w}, \tag{18}$$

where $\psi^\pi(s, a) \in \mathbb{R}^d$ are the successor features of π.

Training. Successor features satisfy a Bellman equation and can therefore be estimated via temporal difference (TD) learning [10]. Given a replay buffer $\mathcal{D} = \{(s, a, s')\}$ and a fixed target policy $\bar{\pi}$, the standard TD loss is

$$\mathcal{L}_{\mathrm{SF}}(\psi) \;=\; \mathbb{E}_{(s,a,s')\sim\mathcal{D}} \left\| \psi(s, a) - \phi(s, a, s') - \gamma\, \bar{\psi}(s', \bar{\pi}(s')) \right\|^2, \tag{19}$$

where $\bar{\psi}$ denotes a stop-gradient (target network) copy of ψ. Once ψ^π has been learned under any exploration policy, adapting to a new task $\mathbf{w}'$ only requires re-fitting the scalar weights using linear regression on a handful of evaluated samples:

$$\mathbf{w}^* = \arg\min_{\mathbf{w}\in\mathbb{R}^d} \mathbb{E}_{s\in\mathcal{D}}\big[(\phi(s)^\top \mathbf{w} - r(s))^2\big],$$

and the zero-shot policy is $\pi_{\mathbf{w}^*}(s) = \arg\max_a \psi(s, a, \mathbf{w}^*)^\top \mathbf{w}^*$, requiring no additional environment interaction.

The central limitation is that the feature map ϕ must be specified in advance (hand-coded or learned), and the expressivity of SF is bounded by the linear span of ϕ, making the quality of the representation a critical bottleneck. Some prominent approaches to learn ϕ directly from data are described below:

- **Autoencoder** [9] learns a compact representation $\phi(s)$ by minimizing the reconstruction error of the input state through an encoder-decoder architecture, capturing the most salient features of the state space. Here, ϕ is the encoder $\phi(s)$ and f is the decoder:
$$\min_{f,\phi} \mathbb{E}_{s\sim\mathcal{D}} \left[(f(\phi(s)) - s)^2\right].$$
- **Transition Model** learns representations by training a model to predict the next state s_{t+1} given the current state-action pair (s_t, a_t), thereby encoding the environment's local dynamics into the latent space. Here, ϕ is the encoder $\phi(s)$ and f is the latent transition predictor:
$$\min_{f,\phi} \mathbb{E}_{(s_t,a_t,s_{t+1})\sim\mathcal{D}} \left[(f(\phi(s_t), a_t) - s_{t+1})^2\right].$$
- **Laplacian (Lap)** [37] learns representations by approximating the eigenfunctions of the graph Laplacian induced by an exploratory policy, encouraging temporally adjacent states to have similar representations while keeping all state representations globally spread apart via an orthonormality regularization. Here, f is the state encoder:

$$\min_f \mathop{\mathbb{E}}_{(s_t,s_{t+1})\sim\mathcal{D}} \left[\|f(s_t) - f(s_{t+1})\|^2\right] + \mathop{\mathbb{E}}_{\substack{s\sim\mathcal{D}\\ s'\sim\mathcal{D}}} \left[(f(s)^\top f(s'))^2 - \|f(s)\|^2 - \|f(s')\|^2\right].$$

- **Lower-Rank Approximation of the transition probability** [36] factorizes the one-step transition probability $P(s'|s,a)$ into a low-rank product $f(s,a)^\top \phi(s')$, effectively performing a spectral decomposition of the transition operator. Here, f and ϕ represent the left and right singular vectors of the transition matrix:

$$\min_{f,\phi} \frac{1}{2}\mathbb{E}_{\substack{(s_t,a_t)\sim\mathcal{D}\\ s'\sim\mathcal{D}}}\left[\left(\left(f(s_t,a_t)^\top\phi(s')\right)\right)^2\right] - \mathbb{E}_{(s_t,a_t,s_{t+1})\sim\mathcal{D}}\left[f(s_t,a_t)^\top\phi(s_{t+1})\right].$$

- **Lower-Rank Approximation of the successor measure** [36] learns a low-rank factorization of the successor measure $M(s,s')$ using temporal difference learning. Here, f and ϕ are the learned factors that represent the state and its temporally extended features:

$$\min_{f,\phi} \mathbb{E}_{\substack{(s_t,s_{t+1})\sim\mathcal{D}\\ s'\sim\mathcal{D}}}\left[\left(f(s_t)^\top\phi(s') - \gamma f(s_{t+1})^\top\bar{\phi}(s')\right)^2\right] - 2\mathbb{E}_{(s_t,s_{t+1})\sim\mathcal{D}}\left[f(s_t)^\top\phi(s_{t+1})\right].$$

- **Bootstrap Your Own Latent (BYOL) [18]** learns a latent space by predicting the next state representation from the current state-action pair using a predictor and a target network. Here, ϕ is the encoder and ψ is the online predictor:

$$\min_{\phi,\psi} \mathbb{E}_{(s_t,a_t,s_{t+1})\sim\mathcal{D}}\left[\|\psi(\phi(s_t),a_t) - \mathrm{sg}(\phi(s_{t+1}))\|_2^2\right].$$

- **Bootstrap Your Own Latent with Bidirectional Prediction (BYOLγ) [22]** captures long-range temporal consistency by predicting future states at horizons sampled geometrically. Here, ψ_f is the forward predictor towards future states and ψ_b is the backward predictor towards past states:

$$\min_{\phi,\psi_f,\psi_b} \mathbb{E}_{\substack{(s_t,a)\sim\mathcal{D}\\ k\sim\mathrm{Geom}(1-\gamma)}}\left[\|\psi_f(\phi(s_t),a) - \mathrm{sg}(\phi(s_{t+k}))\|_2^2 + \|\psi_b(\mathrm{sg}(\phi(s_{t+k}))) - \phi(s_t)\|_2^2\right].$$

B.2 Forward-Backward Representations

Successor features require ϕ to be designed before training and restrict expressivity to the linear span of those features. [35] address both shortcomings by replacing the feature-based decomposition with a direct low-rank factorization of the Successor Measure [10, 5]. For a family of policies $\{\pi_z\}_{z\in\mathcal{Z}}$ parameterized by task embeddings $z \in \mathcal{Z} \subseteq \mathbb{R}^d$, the discounted successor measure of π_z starting from (s,a) is

$$M^{\pi_z}(s,a,X) \;=\; \sum_{t=0}^{\infty}\gamma^t\,\Pr\big(s_{t+1}\in X \mid s_0=s,\, a_0=a,\, \pi_z\big), \qquad X\subseteq\mathcal{S}. \tag{20}$$

The action-value function for any reward r can then be recovered via the linear functional $Q_r^{\pi_z}(s,a) = \mathbb{E}_{s^+\sim M^{\pi_z}(\cdot|s,a)}[r(s^+)]$, which makes the successor measure a sufficient statistic for all tasks simultaneously.

Training. Forward-Backward representations factorize the normalized successor measure into a bilinear product via two neural networks: a Forward map $F : \mathcal{S}\times\mathcal{A}\times\mathcal{Z}\to\mathbb{R}^d$ and a Backward map $B : \mathcal{S}\to\mathbb{R}^d$, trained so that $F(s,a,z)^\top B(s') \approx M^{\pi_z}(s,a,\{s'\})/\rho(s')$, where ρ is a reference state distribution. The Bellman consistency of the successor measure, $M^{\pi_z}(s,a,\cdot) = P(\cdot|s,a) + \gamma\,\mathbb{E}_{a'\sim\pi_z(\cdot|s')}[M^{\pi_z}(s',a',\cdot)]$, yields a tractable TD objective:

$$\mathcal{L}_{\mathrm{FB}}(F,B) \;=\; \mathbb{E}_{\substack{(s,a,s')\sim\mathcal{D},\, z\sim\mathcal{Z}\\ a'\sim\pi_z(\cdot|s'),\, s^+\sim\mathcal{D}}}\left(F(s,a,z)^\top B(s^+) - \gamma\,F(s',a',z)^\top B(s^+)\right)^2 \\ -\; 2\,\mathbb{E}_{(s,a,s')\sim\mathcal{D},\, z\sim\mathcal{Z}}\big[F(s,a,z)^\top B(s')\big], \tag{21}$$

where the first term enforces Bellman consistency and the second anchors the diagonal of the successor measure (i.e., forces $F(s,a,z)^\top B(s') = 1$ on transitions actually visited under π_z). At test time, a new reward r is encoded into a task embedding $z_r = \mathbb{E}[r(s)B(s)]$ and the zero-shot policy is $\pi_{z_r}(s) = \arg\max_a F(s,a,z_r)^\top z_r$, requiring no additional environment interaction.

B.3 TD-JEPA

FB learns the successor measure factorization directly in the state space, without any explicit state encoder. This makes it difficult to scale to raw pixel inputs, where compressed representations are essential, and prevents the reuse of learned features for representation-based transfer. Furthermore, FB offers no mechanism to learn a state encoder that is aligned with long-term, policy-conditioned dynamics. [2] introduce TD-JEPA, which replaces the explicit successor measure factorization of FB with a latent-predictive objective inspired by the Joint-Embedding Predictive Architecture (JEPA) paradigm of [23].

Training TD-JEPA trains four components end-to-end: a state encoder $\phi : \mathcal{S} \to \mathbb{R}^{d_\phi}$, a task encoder $\psi : \mathcal{S} \to \mathbb{R}^{d_\psi}$, a policy-conditioned predictor $T_\phi : \mathbb{R}^{d_\phi} \times \mathcal{A} \times \mathcal{Z} \to \mathbb{R}^{d_\psi}$, and a family of parameterized policies $\{\pi_z\}_{z \in \mathcal{Z}}$.

The predictor is trained to approximate multi-step, policy-conditioned latent dynamics. TD-JEPA uses the Bellman equation for successor features in latent space, yielding the off-policy, single-step TD-JEPA loss:

$$\mathcal{L}_{\text{TD-JEPA}}(\phi, T_\phi) \;=\; \mathbb{E}_{\substack{(s,\,a,\,s') \sim \mathcal{D},\;\; z \sim \mathcal{Z} \\ a' \sim \pi_z(\cdot|s')}} \left\| T_\phi\big(\phi(s),\, a,\, z\big) - \bar{\phi}(s') - \gamma\, T_\phi\big(\bar{\phi}(s'),\, a',\, z\big) \right\|^2, \quad (22)$$

where $\bar{\phi}$ denotes a stop-gradient copy of ϕ (target encoder).

Zero-shot policy extraction follows the FB framework: at test time a, reward r is projected onto the task encoder as $z_r = \mathbb{E}[r(s)\,\psi(s)]$, and the policy $\pi_{z_r}(s) = \arg\max_a T_\phi(\phi(s), a, z_r)^\top z_r$ is executed without any environment interaction.

B.4 BTD-FB

Forward-Backward representations learn a rich, task-agnostic successor measure factorization, but their policy training component still relies on uniformly sampled task vectors $z \sim \text{Unif}(\mathcal{S}^{d-1})$. [4] argue that this uniform sampling strategy suffers from a signal dilution effect: in high-dimensional latent spaces, uniformly drawn task vectors are nearly orthogonal to the behavioral space, causing the variance of returns across policies to vanish and the learning signal to collapse. BTD-FB addresses this by replacing the uniform task prior in FB with the Behavioral Task Distribution (BTD), a data-driven distribution fitted directly to the offline dataset.

Training. BTD-FB first trains Forward and Backward maps F and B using the standard FB objective (Eq. 21). Following training, the state embedding is fixed as $\phi(s) = \mathbb{E}[BB^\top]^{-1}B(s)$. To construct the BTD, N_τ sub-trajectories of random lengths are sampled from the offline dataset, and their empirical feature occupancies are computed and normalized to yield task vectors:

$$z_\tau = \frac{\tilde{\psi}^\tau}{\|\tilde{\psi}^\tau\|_2}, \qquad \tilde{\psi}^\tau = \sum_{t=0}^{|\tau|} \gamma^t \phi(s_t). \quad (23)$$

A Gaussian Mixture Model $p_\theta(z)$ is then fitted by maximum likelihood to the resulting empirical task set. The conditional policy is trained from scratch on these extracted tasks instead.

C Background on Quality-Diversity Methods

Quality-Diversity (QD) algorithms discover large collections of diverse, high-performing solutions simultaneously. A QD problem is defined by a fitness function $r : \mathcal{S} \to \mathbb{R}$ and a behavioral descriptor $\beta : \mathcal{S} \to \mathcal{B}$, where $\mathcal{B}$ is discretized into a finite grid of cells. An archive $\mathcal{C}$ stores at most one elite per cell, replaced only when a candidate achieves strictly higher fitness in the same cell. Performance is measured by *coverage* (fraction of populated cells) and *QD-score* (sum of all elite fitnesses). Most QD variants share this skeleton and differ only in their choice of variation operator, as summarized in Algorithm 1.

Algorithm 1 Base Quality-Diversity Algorithm

Require: Fitness function r, descriptor ϕ, budget T
Initialize archive $\mathcal{C} \leftarrow \emptyset$
for $t = 1$ **to** T **do**
  Sample parent $\theta \sim \mathcal{C}$ ▷ randomly if $\mathcal{C} = \emptyset$
  $\theta' \leftarrow \textsc{Operator}(\theta, \mathcal{C})$ ▷ differs across methods
  Evaluate fitness $f \leftarrow r(\theta')$ and descriptor $b \leftarrow \phi(\theta')$
  if $f > r(\mathcal{C}[b])$ **then**
    $\mathcal{C}[b] \leftarrow (\theta',\ f)$
  end if
end for
return $\mathcal{C}$

C.1 MAP-Elites

MAP-Elites (ME) [26] is the canonical QD algorithm. It addresses the exploration-exploitation tension in a single-objective optimizer by explicitly maintaining one best solution per behavioral cell, so that the search covers the descriptor space rather than collapsing onto a single high-fitness region.

Operator. Parents are sampled from the archive and perturbed by an isotropic Gaussian mutation:

$$\theta' = \theta + \varepsilon, \qquad \varepsilon \sim \mathcal{N}(0, \sigma^2 I). \tag{24}$$

Each offspring is evaluated and inserted into its behavioral cell if it improves the current elite.

ME is simple and highly parallelizable. But because mutation is isotropic, ME does not exploit any structure of the fitness landscape: every direction in parameter space is equally likely to be explored, regardless of which directions have previously led to archive improvements.

C.2 CMA-ME

CMA-ME [15] addresses the sample-inefficiency of isotropic mutation by replacing it with CMA-ES [20], which maintains an adaptive covariance matrix $\mathbf{C}$ estimated from directions that previously improved the archive.

Operator. Offspring are sampled from a multivariate Gaussian whose covariance is updated from archive improvement signals:

$$\theta' = \theta + \varepsilon, \qquad \varepsilon \sim \mathcal{N}(0, \sigma^2 \mathbf{C}). \tag{25}$$

Despite adapting the mutation distribution, CMA-ME still relies purely on undirected sampling: it does not use any task-specific gradient information to steer offspring towards higher-fitness regions.

C.3 PGA-ME

PGA-ME [27] introduces a gradient-based variation operator to guide the search towards high-fitness regions, complementing random mutation with directed improvement steps.

Operator. Half of each generation's offspring are produced by Gaussian mutation as in ME; the other half by gradient ascent on a learned critic Q_ϕ trained via TD3:

$$\theta \leftarrow \theta + \alpha \nabla_\theta Q_\phi\big(s,\ \pi_\theta(s)\big). \tag{26}$$

The critic is trained on transitions collected during archive evaluation.

The limitation with PGA-ME is that gradient steps optimize fitness without any awareness of the behavioral descriptor, so the method cannot explicitly target underpopulated regions of the archive, and exploration can collapse on the optimal behavior.

C.4 DCRL-ME

Motivation. DCRL-ME [13] extends PGA-ME by conditioning both actor and critic on a target behavioral descriptor b^*, so that gradient steps jointly optimize fitness and steer the offspring toward a desired region of the descriptor space.

Operator. The policy gradient update becomes descriptor-conditioned:

$$\theta \leftarrow \theta + \alpha \nabla_\theta Q_\phi\big(s,\, \pi_\theta(s, b^*),\, b^*\big), \tag{27}$$

and the conditioned actor $\pi_\theta(\cdot, b^*)$ is also used as a generative model, injecting candidates directly into target archive cells.

Both the actor and critic in DCRL-ME are trained from scratch using only transitions collected during archive evaluation, making them dependent on the quality and density of the reward signal encountered online.

C.5 DDE-Elites

Motivation.

DDE-Elites [17] addresses the problem of high dimensionality of the policy parameter space by learning a compact Data-Driven Encoding (DDE) from the archive elites using a VAE, and using it as a variation operator within ME.

Operators. At each generation, a VAE is retrained on all current archive solutions by minimzing this loss (maximizing the Evidence Lower Bound):

$$\mathcal{L}_{\mathrm{VAE}} = -\mathbb{E}_{q_{\theta_e}(z|\theta)}[\log p_{\theta_d}(\theta \mid z)] + \mathrm{KL}(q_{\theta_e}(z|\theta) \,\|\, \mathcal{N}(0, I))\,. \tag{28}$$

Three variation operators are combined at each generation: **(i) Isometric mutation** applies Gaussian noise directly in parameter space; **(ii) line mutation** applies directional Gaussian noise, where the variance per dimension scales with the difference between two archive elites, biasing perturbations toward directions of known diversity; **(iii) reconstructive crossover** shifts a solution toward the VAE-learned distribution without adding noise:

$$\theta' = \tfrac{1}{2}\big(\theta + D_{\theta_d}(E_{\theta_e}(\theta))\big). \tag{29}$$

The mix of operators is selected at each generation by a UCB1 bandit algorithm [1], which tracks the success rate of each operator and balances exploration and exploitation automatically.

The compact latent space simplifies search, but the VAE is trained only on solutions already in the archive, so the latent space does not necessarily reflect task-relevant structure.

C.6 ME+Pretrain

This is a baseline that we introduce to isolate the contribution of offline initialization from the structure of a behavioral latent space. In ME+Pretrain, MLP policies are warm-started via offline TD3 on the same dataset as the BFM, using the task fitness as reward, before running standard ME.

Operator. The pretraining objective is:

$$\theta^\star = \arg\max_\theta\ \mathbb{E}_{(s,a)\sim\mathcal{D}}\big[Q_\omega\big(s,\, \pi_\theta(s)\big)\big] \tag{30}$$

after which ME runs from $\theta^\star$.

D Experimental Details

D.1 Environments and Tasks

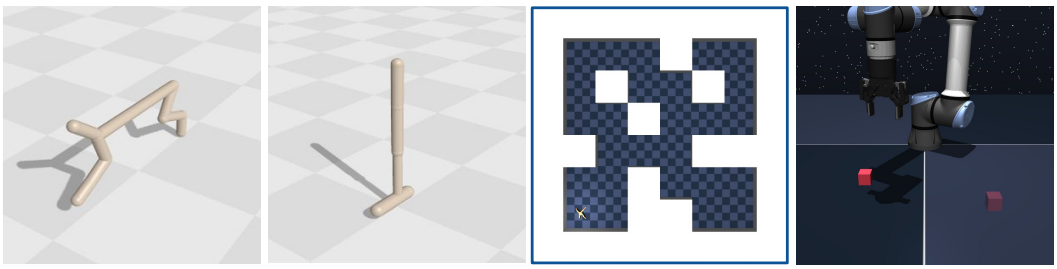


Figure 6: **Evaluation environments.** The four benchmarks used in this work: HalfCheetah_uni, Walker2d_uni, AntMaze-medium, and Cube-single (from left to right). Difficulty increases from left to right, with AntMaze-medium and Cube-single introducing long-horizon tasks and contact-rich interaction, respectively.

HalfCheetah_uni:

A planar 2D robot from QDax [8]. It has a state dimension of 17 and an action dimension of 6. It represents the simplest setting with dense rewards and straightforward dynamics.

Behavioral descriptor:

- **Foot contact pattern (2D):** The ratio of the contact of the front/back feet with the ground.

Fitness functions:

- **Walk:** Reward is linearly proportional to forward velocity, capped at 2 m/s.
- **Run:** Reward is linearly proportional to forward velocity, capped at 5 m/s.
- **Walk backward:** Reward is linearly proportional to backward velocity, capped at -2 m/s.
- **Run backward:** Reward is linearly proportional to backward velocity, capped at -5 m/s.

Walker2D_uni:

A bipedal walker from QDax, with slightly more complex dynamics than HalfCheetah_uni. It has a state dimension of 17 and an action dimension of 6.

Behavioral descriptor:

- **Foot contact pattern (2D):** The ratio of the contact of the front/back feet with the ground.

Fitness functions:

- **Run forward**: Maximize the velocity of the center-of-mass of the Walker agent.
- **Run backward**: Maximize the backward velocity of the point-of-mass of the Walker agent.
- **Upright walk**: Maximize forward velocity while maintaining an upright torso above 0.3.
- **Jump**: Maintain a center-of-mass height above 1.2.

AntMaze-medium:

A quadrupedal Ant robot navigating a medium-sized maze from OGBench [28]. It has a state dimension of 29 and an action dimension of 8, and introduces long-horizon navigation with deceptive reward landscapes.

Behavioral descriptor:

- (x, y) **position:** the coordinate of the position of the ant in the maze.

- **Foot contact pattern (4D):** The ratio of the contact of the ant's feet with the ground.

Fitness functions:

- **Energy efficiency**: Minimize arm energy consumption (norm of the actions), i.e. maximize the fitness function $f = \sqrt{d_A} - \|a_t\|_2$, with $d_A = 8$.
- **Reach center**: Reach the center of the maze in position (12, 12) with a radius of 1.
- **Top wall**: Reach the top wall of the maze after $x = 20$.
- **Right wall**: Reach the right wall of the maze after $y = 20$.
- **Top wall dense**: The dense version of **top wall**, with fitness $= e^{(\text{ant.x}-20)/3}$
- **Right wall dense**: The dense version of **right wall**, with fitness $= e^{(\text{ant.y}-20)/3}$

Cube-single: A 6-DoF robotic arm from OGBench. The state space dimension is 28. The action space is 5-dimensional ($\Delta x, \Delta y, \Delta z, \Delta$yaw, gripper speed), and it represents the hardest setting requiring temporally extended contact-rich manipulation.

Behavioral descriptor:

- (x, y) **position**: the (x, y) position of the cube.
- (x, z) **position**: replaces the y position with the height of the cube.
- **Grasp style**: (Relative grasp angle, how wide the grip is).
- **Wrist configuration**: (the angle in joint_3, the angle in joint_4)

Fitness functions:

- **Energy efficiency**: Minimize arm energy consumption (norm of the actions), i.e. maximize the fitness function $f = \sqrt{d_A} - \|a_t\|_2$, with $d_A = 5$.
- **Pick-and-place**: Binary success of lifting and placing the cube at a target location.

QD-score normalization. Each environment contains multiple tasks with different reward scales, making raw QD-scores incomparable across tasks. To aggregate results within an environment, we apply per-task min-max normalization: for each task, we compute the minimum and maximum QD-score observed across all methods and all seeds, and rescale each method's score to $[0, 1]$ accordingly. Normalized scores are then averaged across all tasks within an environment to produce the aggregate curves shown in Figure 2. This normalization is applied identically to QD-score and max-fitness curves. Unnormalized per-task QD-scores are reported in Appendix E.12.

D.2 Baselines Implementation Details

All baselines share the same environment setup, archive configuration, training budget, and random seeds as BFM-QD for a fair comparison.

Table 2: Shared hyperparameters across all methods.

Parameter	Value
Archive grid size	50×50
Number of generations	500
Parallel environments	400
Episode length (Brax)	500 steps
Episode length (OGBench)	1 000 steps

MAP-Elites. Offspring are produced by isotropic Gaussian mutation. Five operator instances of Gaussian mutation with a sigma ladder run in parallel, each handling an equal share of candidates per generation. Archive policies are 2-hidden-layer MLPs with ReLU activations: `input` $\rightarrow$ `128` $\rightarrow$

Table 1: Summary of environments, behavioral descriptors, and fitness functions used in our evaluation. Difficulty combines reward density and task complexity: **Easy** = dense reward, simple dynamics; **Medium** = dense reward, complex dynamics or navigation; **Hard** = dense but deceptive or long-horizon; **Very Hard** = sparse reward, contact-rich manipulation.

Environment	Behavioral Descriptor	Fitness Function	Density	Difficulty
HalfCheetah	Foot contact	Run backward	Dense	Easy
		Walk backward	Dense	Easy
		Walk forward	Dense	Easy
		Run forward	Dense	Easy
Walker	Foot contact	Run backward	Dense	Easy
		Run forward	Dense	Easy
		Upright walk	Dense	Medium
		Jump	Dense	Medium
AntMaze-medium	(x, y) position	Energy efficiency	Dense	Medium
	Foot contact	Reach center	Dense	Hard
		Reach right wall	Dense	Hard
		Reach top wall	Dense	Hard
		Reach right wall	Sparse	Very Hard
		Reach top wall	Sparse	Very Hard
Cube-single	(x, y) position of cube	Energy efficiency	Dense	Hard
	(x, z) position of cube	Energy efficiency	Dense	Hard
	Grasp style	Pick-and-place	Sparse	Very Hard
	Wrist configuration	Pick-and-place	Sparse	Very Hard

`relu` $\rightarrow$ `128` $\rightarrow$ `output` $\rightarrow$ `tanh`, taking the state observation as input and producing actions through a $\tanh$ output layer.

Table 3: MAP-Elites hyperparameters.

Parameter	Value
Mutation σ	$\{0.1, 0.5, 1.0, 1.0, 5.0\}$

CMA-ME. Uses the same emitter configuration and archive policy architecture as ME, but replaces isotropic mutation with CMA-ES, adapting the covariance matrix from archive improvement signals at each generation.

PGA-ME. Archive policies are 2-hidden-layer MLPs with 128 units per layer, ReLU activations, and a $\tanh$ output layer. Half of each generation's offspring are produced by Gaussian mutation; the other half by one step of gradient ascent on a TD3 critic. The critic is a twin Q-network with two hidden layers of 256 units and ReLU activations, taking the concatenation of state and action as input. The TD3 actor has two hidden layers of 128 units with ReLU activations and produces a deterministic action through a $\tanh$ output layer. Both are trained online from transitions collected during archive evaluation. A warmup phase is applied before any PG variation, allowing the replay buffer to fill before critic-guided mutations begin.

Table 4: PGA-ME hyperparameters.

Parameter	Value
Archive policy MLP	2×128 hidden units
Critic hidden width	256
Critic learning rate	3×10^{-4}
Actor learning rate	3×10^{-4}
Replay buffer size	1 000 000
Batch size	256
Discount γ	0.99
Target smoothing τ	0.005
TD3 policy noise σ	0.2
TD3 noise clip	0.5
Target update frequency	every 2 critic steps
Critic updates per gen.	256
Warmup	20 generations

DCRL-ME. Extends PGA-ME with a descriptor-conditioned actor and critic, both taking the normalized 2-dimensional behavioral descriptor concatenated to their respective inputs (state for the actor, state-action for the critic). The archive policy MLP, actor, and critic retain the same widths and activations as ME. The offspring batch is split across three variation operators: Gaussian mutation (GA), descriptor-conditioned PG, and Actor Injection (AI). For AI, a target descriptor b^* is sampled uniformly from the descriptor space and the actor's first layer is analytically specialized to it: since the first layer computes $W[s;\, b^*] + b$, fixing b^* reduces it to a standard affine map $W_s\, s + (W_{b^*} b^* + b)$, yielding a valid archive policy with no additional forward pass. TD3 training hyperparameters are otherwise identical to PGA-ME (Table 4).

Table 5: DCRL-ME hyperparameters. TD3 parameters are identical to PGA-ME (Table 4).

Parameter	Value
Offspring split (GA / PG / AI)	50% / 25% / 25%
Similarity length scale L	0.1

DDE-Elites. Archive policies use the same MLP architecture as ME. The VAE has a symmetric encoder and decoder, each consisting of two hidden layers of 256 units with ReLU activations. The encoder maps a flattened policy parameter vector to a 50-dimensional latent space, producing a mean and log-variance from which a latent code is sampled via the reparameterization trick. The VAE is retrained every generation on all current archive weight vectors. Three variation operators are used: isometric mutation, line mutation, and reconstructive crossover (averaging a solution with its VAE reconstruction). Their mix is selected each generation by a UCB1 bandit. When fewer than the minimum number of elites are present, only isometric and line mutation are used.

Table 6: DDE-Elites hyperparameters.

Parameter	Value
Archive policy MLP	2×128 hidden units
VAE latent dimension	50
VAE encoder/decoder width	256
VAE learning rate	3×10^{-4}
VAE training epochs	5
VAE KL weight	0.01
VAE retraining frequency	every generation
Min. elites to retrain	50
σ isometric mutation	0.003
σ line mutation	0.1
Bandit sliding window	1000

ME+Pretrain. MLP policies are initialized by offline TD3 training on the same dataset used to train the BFM, using the task fitness as reward. The offline actor and critic use the same architecture as in PGA-ME and DCRL-ME. The pretrained weights are then used as the starting point for ME with the same Gaussian mutation and emitter configuration as the ME baseline.

Table 7: ME+Pretrain hyperparameters (offline TD3 phase).

Parameter	Value
Offline training steps	2 000 000
Batch size	2 048
Learning rate (actor, critic)	10^{-4}
Target Polyak τ	0.01
Hidden width	1024
Feature dimension	512

D.3 BFM Implementation Details

Laplacian (Lap). Lap learns only one state encoder f: $|\mathcal{S}| \xrightarrow{\texttt{ntanh}} 1024 \xrightarrow{\texttt{ReLU}} 1024 \xrightarrow{\texttt{ReLU}} d_z$.

BYOLγ representation. BYOLγ learns three networks:
• The state encoder g: $|\mathcal{S}| \xrightarrow{\texttt{ntanh}} 1024 \xrightarrow{\texttt{ReLU}} d_z$ [2], with a target network $\bar{g}$ updated by EMA.
• The forward predictor f: $(d_z + |\mathcal{A}|) \xrightarrow{\texttt{ntanh}} 1024 \xrightarrow{\texttt{ReLU}} 1024 \xrightarrow{\texttt{ReLU}} d_z$.
• The backward predictor b: $d_z \xrightarrow{\texttt{ntanh}} 1024 \xrightarrow{\texttt{ReLU}} 1024 \xrightarrow{\texttt{ReLU}} d_z$.
All three are trained jointly with an orthogonality regularizer on g.

[2] `ntanh` is layer normalization followed by a Tanh activation

Table 8: SF-TD3 hyperparameters (shared across Lap and BYOLγ variants).

Parameter	Value
Latent dimension d_z	50
SF training steps	2 000 000
SF batch size	1 024
SF learning rate	10^{-4}
SF target Polyak τ	0.01
Actor/critic training steps	2 000 000
Actor/critic learning rate	10^{-4}
TD3 batch size	2 048
Discount γ	0.99
Target EMA τ	0.01
z-inference batch size	10 000

Forward-Backward (FB) model. The FB model follows [35]. The forward map F takes (s, a, z) through two parallel streams:

• obs–action: $(|\mathcal{S}| + |\mathcal{A}|) \xrightarrow{\texttt{ntanh}} 512$ [ReLU].

• obs–z: $(|\mathcal{S}| + d_z) \xrightarrow{\texttt{ntanh}} 512$ [ReLU].

Their concatenation passes through a shared trunk $1024 \xrightarrow{\texttt{ReLU}} 1024 \xrightarrow{\texttt{ReLU}} d_z$.

The backward network B is a 4-hidden-layer MLP: $|\mathcal{S}| \xrightarrow{\texttt{ntanh}} 1024 \xrightarrow{\texttt{ReLU}} 1024 \xrightarrow{\texttt{ReLU}} 1024 \xrightarrow{\texttt{ReLU}} d_z$, ℓ_2-normalized and rescaled to radius $\sqrt{d_z}$. The actor shares the same preprocessed trunk as F (obs-only and obs–z streams) with a $\tanh$ action head.

Table 9: FB hyperparameters.

Parameter	Value
Latent dimension d_z	50
Training steps	2 000 000
Batch size	1 024
Learning rate	10^{-4}
Discount γ	0.99
EMA τ	0.01
z-inference batch size	10 000

TD-JEPA. TD-JEPA learns four components:

• State encoder ϕ: $|\mathcal{S}| \xrightarrow{\texttt{ntanh}} 1024 \xrightarrow{\texttt{ReLU}} d_\phi$, where $d_\phi = 512$.

• Task encoder ψ: $|\mathcal{S}| \xrightarrow{\texttt{ntanh}} 1024 \xrightarrow{\texttt{ReLU}} d_z$, ℓ_2-normalized and rescaled to radius $\sqrt{d_z}$.

• Forward predictor T_ϕ (predicting future ψ from current ϕ, twin MLP): $(d_\phi + d_z + |\mathcal{A}|) \xrightarrow{\texttt{ntanh}} 1024 \xrightarrow{\texttt{ReLU}} d_z$.

• Backward predictor T_ψ (predicting future ϕ from current ψ, twin MLP): $(d_z + d_z + |\mathcal{A}|) \xrightarrow{\texttt{ntanh}} 1024 \xrightarrow{\texttt{ReLU}} d_\phi$.

Each twin predictor averages its two heads before computing the TD-JEPA loss. Target networks are maintained for ϕ, ψ, T_ϕ, and T_ψ via EMA.

• The actor: $(d_\phi + d_z) \xrightarrow{\texttt{ntanh}} 1024 \xrightarrow{\texttt{ReLU}} 1024 \xrightarrow{\texttt{ReLU}} |\mathcal{A}|$, with $\tanh$ output.

Table 10: TD-JEPA hyperparameters.

Parameter	Value
Latent dimension d_z	50
Training steps	2 000 000
Batch size	1 024
Learning rate	10^{-4}
Target EMA τ	0.01
Actor std. dev. σ	0.2
Actor std. dev. clip	0.3
Ortho. coeff. λ_ϕ	1.0
Ortho. coeff. λ_ψ	1.0
z-inference batch size	10 000

Backward Inference (BI) Operator. At each generation, half of the offspring are produced by Gaussian mutation and half by the BI operator. For the BI operator, state-reward pairs are sampled from the replay buffer and used to compute z^* in closed form via Equation 2. Each parent is then interpolated toward z^* with step size α and projected back onto the latent sphere.

Table 11: Backward Inference (BI) operator hyperparameters.

Parameter	Value
Step size α	0.02
BI probability	0.5
Gaussian mutation σ	1.0
z-inference batch size	10 000

Interpolation (Iso) Operator. At each generation, half of the offspring are produced by Gaussian mutation and half by the interpolation operator. For the interpolation operator, two parents are sampled from the archive and linearly interpolated with a coefficient $\alpha \sim \mathcal{U}[0, 1]$, before projecting back onto the latent sphere.

Table 12: Interpolation (Iso) operator hyperparameters.

Parameter	Value
Interpolation probability	0.5
α distribution	$\mathcal{U}[0, 1]$
Gaussian mutation σ	1.0

Dataset collection for BFM-QD For `HalfCheetah_uni` and `Walker2D_uni`, we collect our own dataset using RND [7]. We use the official implementation from [21]. We keep the exact same hyperparameters.

Pseudo-algorithm of BFM-QD Algorithm 2 summarizes the BFM-QD framework. The key difference from standard QD (Algorithm 1) is that the archive stores latent codes z rather than policy weights θ, and all variation operators act directly in $\mathcal{Z}$. The BFM policy $\pi_{\text{BFM}}(\cdot \mid z)$ is called at evaluation time and remains frozen throughout.

Algorithm 2 BFM-QD

Require: Pretrained BFM policy $\pi_{\text{BFM}}(\cdot \mid z)$
Require: Latent space $\mathcal{Z}$, archive $\mathcal{C}$
Require: Number of iterations T, batch size N
Initialize archive $\mathcal{C} \leftarrow \emptyset$
for $t = 1$ to T **do**
  if $\mathcal{C}$ is empty **then**
    Sample N latent vectors $\{z_i\}_{i=1}^{N} \sim \mathcal{Z}$
  else
    Sample parent latent vectors $\{z_i\}_{i=1}^{N}$ from $\mathcal{C}$
  end if
  Generate offspring latents: $z_i' = \text{Perturbation}(z_i)$
  for each latent vector z_i' **do**
    Roll out policy $\pi_{\text{BFM}}(\cdot \mid z_i')$
    Evaluate fitness f_i and descriptor b_i
    Attempt insertion into archive $\mathcal{C}$ using (z_i', f_i, b_i)
  end for
end for
return archive $\mathcal{C}$

Algorithms 3 and 4 detail two variation operators: **(1)** The backward Inference defined in 4.2; **(2)** Interpolation defined in 5.3.2.

Algorithm 3 BFM-QD with Backward Inference (BI) Operator

Require: Pretrained BFM policy $\pi_{\text{BFM}}(\cdot \mid z)$, feature map ϕ
Require: Latent space $\mathcal{Z}$, archive $\mathcal{C}$, replay buffer $\mathcal{D}$
Require: Number of iterations T, batch size N, step size α, latent dimension d_z
Initialize archive $\mathcal{C} \leftarrow \emptyset$, replay buffer $\mathcal{D} \leftarrow \emptyset$
for $t = 1$ to T **do**
  if $\mathcal{C}$ is empty **then**
    Sample N latent vectors $\{z_i'\}_{i=1}^{N} \sim \mathcal{Z}$
  else
    Sample parent latent vectors $\{z_i\}_{i=1}^{N}$ from $\mathcal{C}$
    Sample $\mathcal{D}_r = \{(s_t, r(s_t))\}$ from $\mathcal{D}$
    $z^* \leftarrow \left(\mathbb{E}_{s\sim\mathcal{D}_r}[\phi(s)\phi(s)^\top]\right)^{-1} \mathbb{E}_{s\sim\mathcal{D}_r}[\phi(s)r(s)]$
    for each z_i **do**
      if with probability 0.5 **then**
        $z_i' \leftarrow z_i + \varepsilon, \quad \varepsilon \sim \mathcal{N}(0, \sigma^2 I)$ ▷ Gaussian mutation
      else
        $z_i' \leftarrow (1-\alpha)\cdot z_i + \alpha \cdot z^*$ ▷ Backward Inference
      end if
      $z_i' \leftarrow z_i'/\|z_i'\| \cdot \sqrt{d_z}$ ▷ project all offspring onto sphere
    end for
  end if
  $z_i' \leftarrow z_i'/\|z_i'\| \cdot \sqrt{d_z}$ ▷ project onto latent sphere
  for each latent vector z_i' **do**
    Roll out policy $\pi_{\text{BFM}}(\cdot \mid z_i')$, collecting trajectory $\tau_i = \{(s_t, r(s_t))\}$
    Evaluate fitness f_i and descriptor b_i
    Store trajectory τ_i in replay buffer $\mathcal{D}$
    Attempt insertion into archive $\mathcal{C}$ using (z_i', f_i, b_i)
  end for
end for
return archive $\mathcal{C}$

Algorithm 4 BFM-QD with Interpolation (Iso) Operator

Require: Pretrained BFM policy $\pi_{\text{BFM}}(\cdot \mid z)$
Require: Latent space $\mathcal{Z}$, archive $\mathcal{C}$
Require: Number of iterations T, batch size N, latent dimension d_z

Initialize archive $\mathcal{C} \leftarrow \emptyset$
for $t = 1$ to T **do**
  if $\mathcal{C}$ is empty **then**
    Sample N latent vectors $\{z'_i\}_{i=1}^{N} \sim \mathcal{Z}$
  else
    Sample parent latent vectors $\{z_i\}_{i=1}^{N}$ from $\mathcal{C}$
    for each z_i **do**
      if with probability 0.5 **then**
        $z'_i \leftarrow z_i + \varepsilon, \quad \varepsilon \sim \mathcal{N}(0, \sigma^2 I)$ ▷ Gaussian mutation
      else
        Sample second parent $z_j \sim \mathcal{C}$
        Sample $\alpha \sim \mathcal{U}[0, 1]$
        $z'_i \leftarrow (1 - \alpha) \cdot z_i + \alpha \cdot z_j$ ▷ Interpolation
      end if
      $z'_i \leftarrow z'_i / \|z'_i\| \cdot \sqrt{d_z}$ ▷ project all offspring onto sphere
    end for
  end if
  $z'_i \leftarrow z'_i / \|z'_i\| \cdot \sqrt{d_z}$ ▷ project onto latent sphere
  for each latent vector z'_i **do**
    Roll out policy $\pi_{\text{BFM}}(\cdot \mid z'_i)$
    Evaluate fitness f_i and descriptor b_i
    Attempt insertion into archive $\mathcal{C}$ using (z'_i, f_i, b_i)
  end for
end for
return archive $\mathcal{C}$

E Additional Results

E.1 Expanded main results

Figure 7 reports the per-task, unnormalized QD-score curves for all methods across all 18 tasks and four environments. The results confirm the aggregated findings in Figure 2: BFM-QD variants consistently lead across tasks, with the performance gap over parameter-space baselines widening sharply as reward density decreases and task complexity increases.

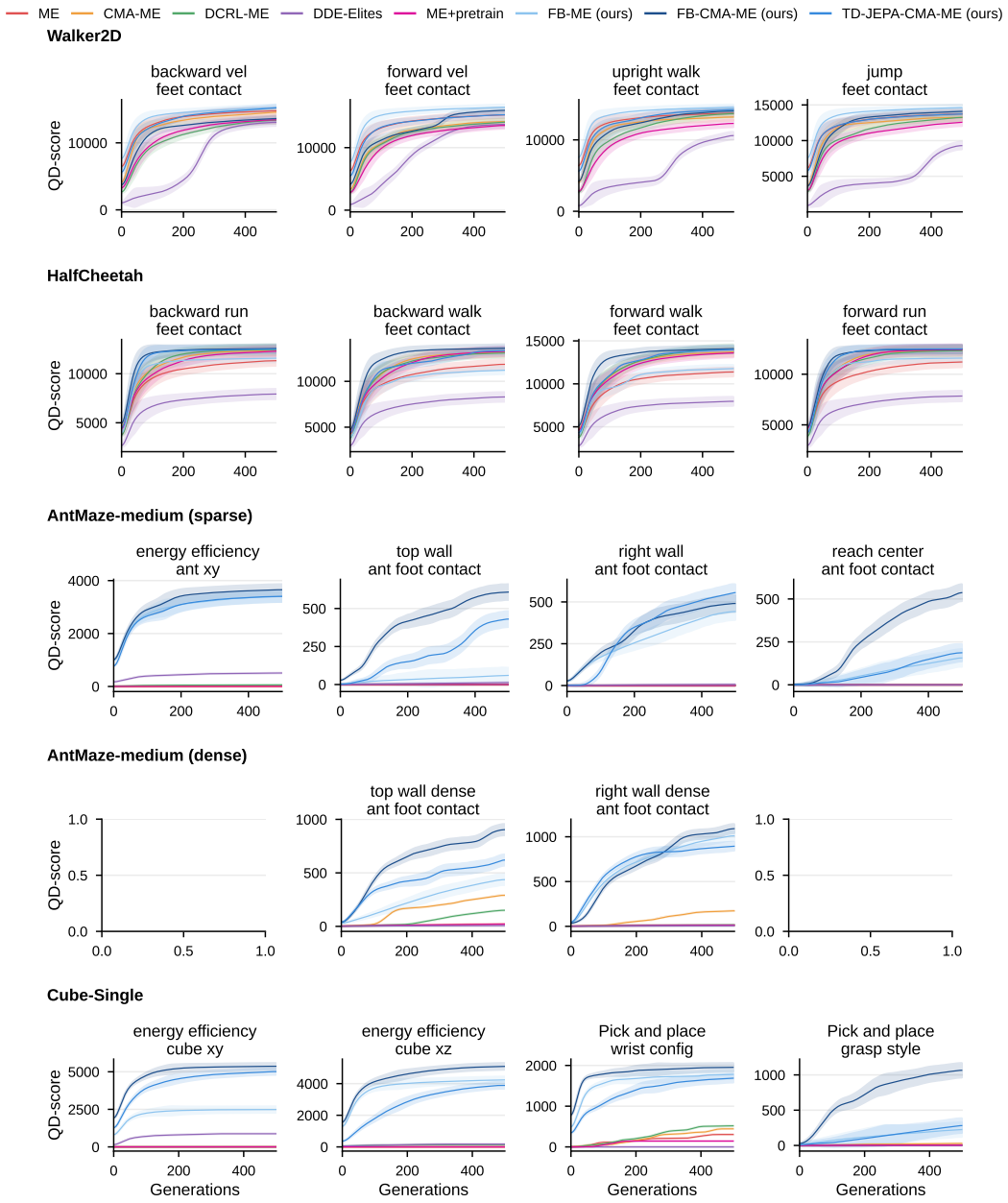


Figure 7: **QD-score across all environments and methods, averaged over 5 seeds.** BFM-QD variants (FB-ME, FB-CMA-ME, TD-JEPA-CMA-ME) consistently outperform parameter-space baselines (ME, CMA-ME, DCRL-ME, DDE-Elites), with margins that grow substantially in sparse and deceptive tasks (AntMaze-medium, Cube-single) where all parameter-space methods collapse to near-zero performance.

E.2 Archive Visualization

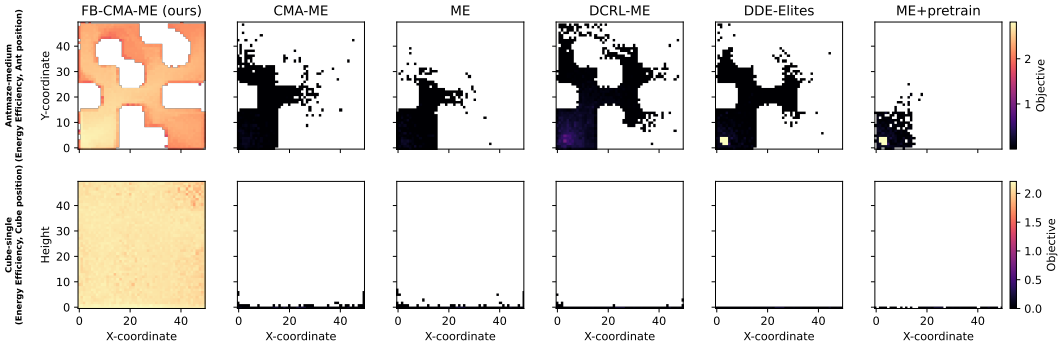


Figure 8: **Archive coverage.** Heatmaps of the descriptor space populated by FB-CMA-ME versus CMA-ME, ME, DCRL-ME, DDE-Elites, and ME+Pretrain, on AntMaze-medium (descriptor: xy-position, fitness: energy efficiency) and Cube-single (descriptor: cube $[x, \text{height}]$, fitness: energy efficiency). FB-CMA-ME achieves dense, broad coverage while parameter-space methods leave large regions entirely unpopulated.

Figure 8 visualizes the archive heatmaps for FB-CMA-ME, CMA-ME, ME, DCRL-ME, and DDE-Elites on AntMaze-medium and Cube-single, making the collapse of parameter-space methods concrete. On AntMaze-medium, CMA-ME populates only a small cluster near the origin. On Cube-single, the CMA-ME archive is almost entirely empty, except for moving the cube on the surface of the table, because successfully holding the cube requires a temporally extended sequence of approach, grasp, and transport that random parameter perturbations cannot discover. FB-CMA-ME sidesteps both failure modes entirely, achieving dense and uniform coverage across almost the full descriptor space in both environments.

E.3 The latent space is behaviorally rich by construction

Figure 9 shows the archive coverage heatmaps obtained by random sampling from the trained FB latent space and the MLP parameter space, across all four environments.

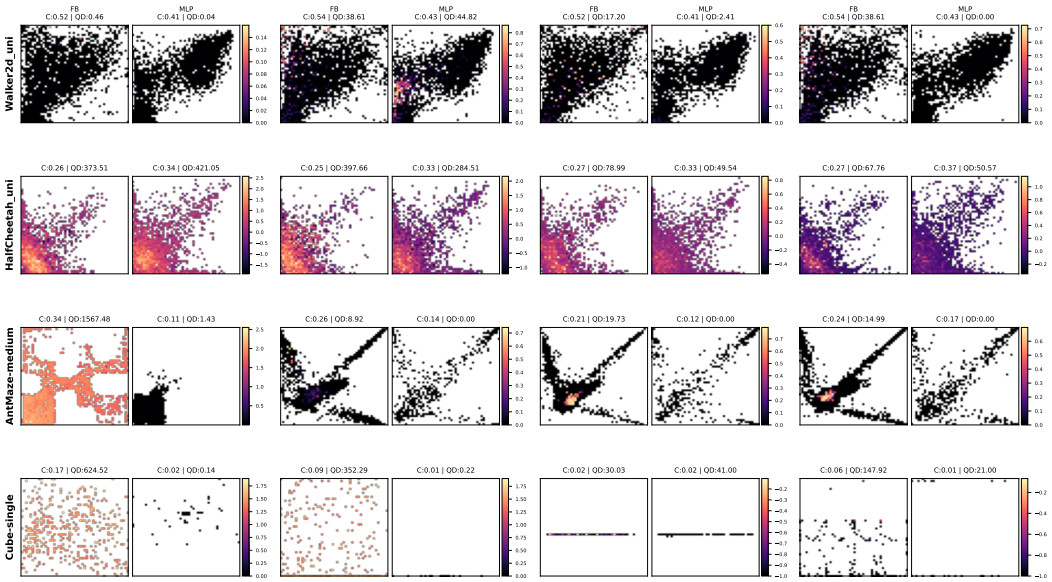


Figure 9: **Archive coverage from random sampling.**

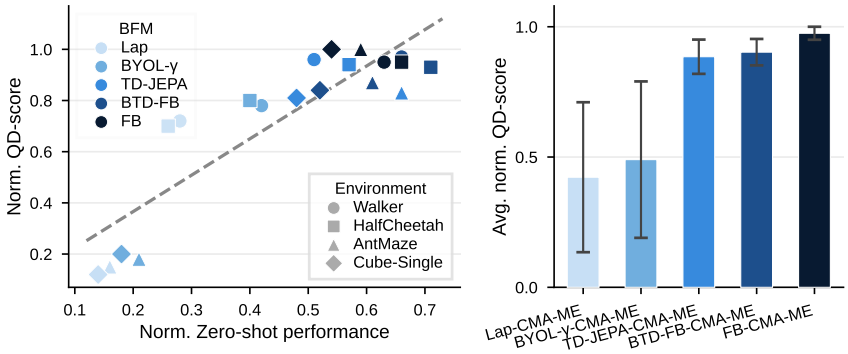


Figure 10: **BFM-QD does not require high zero-shot performance.** (Left) Methods with modest zero-shot returns still attain high QD-scores, suggesting latent space diversity matters more than zero-shot accuracy. (Right) FB-CMA-ME achieves the highest average QD-score, closely followed by BTD-FB-CMA-ME and TD-JEPA-CMA-ME.

E.4 Effect of the Behavioral Foundation Model

We investigate how the choice of BFM affects QD performance. Different BFMs learn the feature map ϕ and latent space $\mathcal{Z}$ through different objectives, which may yield latent spaces of varying quality and behavioral coverage. Understanding this sensitivity is important for practitioners choosing a BFM backbone for QD search.

Experiment We fix the QD optimizer to CMA-ME and vary the BFM across five methods: Laplacian (Lap) [37], BYOLγ [22], TD-JEPA, BTD-FB [4], and FB. For each BFM, we report its zero-shot performance on the task fitness function alongside the resulting QD-score to assess whether zero-shot performance is a reliable predictor of QD performance.

Results Results are shown in Figure 10. FB achieves the highest average QD-score, closely followed by BTD-FB and TD-JEPA, while Lap and BYOLγ perform worse, particularly on AntMaze-medium and Cube-single. Overall, zero-shot performance is a good predictor of QD-score, and the regression confirms a clear positive correlation. However, on easy tasks, even BFMs with low zero-shot performance can achieve strong QD-scores, suggesting that a highly accurate BFM is not strictly necessary when tasks are dense and simple.

Takeaway: Zero-shot performance is a reliable proxy for QD performance: stronger BFMs consistently yield higher QD-scores. While weaker BFMs can still perform well on easy tasks, the choice of the BFM becomes critical as task difficulty increases.

E.5 Diversity as an Optimization Strategy

We investigate whether diversity pressure in QD search provides benefits beyond single-objective optimization in the BFM latent space. A natural question is whether simply optimizing for fitness in $\mathcal{Z}$ is sufficient, or whether maintaining a diverse archive of solutions actively helps find better optima.

Experiment We compare three methods that all operate within the FB latent space. **FB Zero-shot** uses the closed-form backward inference (Equation 2) to obtain z^* in a single pass without any additional environment interaction. **FB-CMA-ES** runs a single CMA-ES [19] optimizer directly in $\mathcal{Z}$, optimizing for maximum fitness without diversity pressure. **FB-CMA-ME** adds QD search on top of the same latent space.

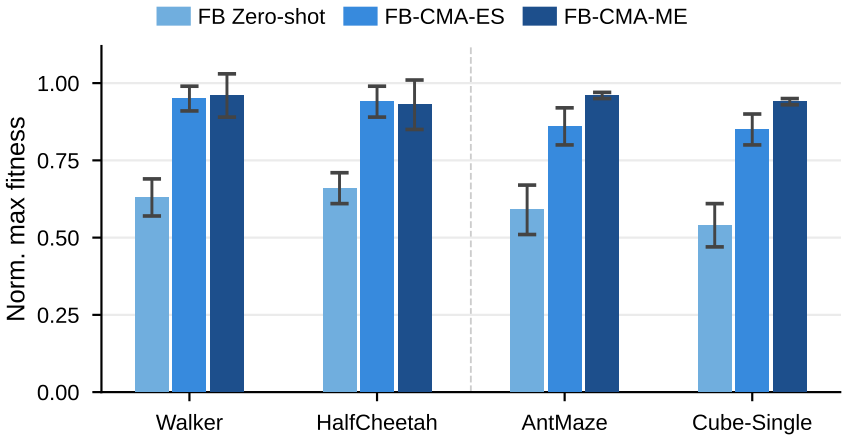


Figure 11: **Diversity as an optimization strategy.** Comparison of FB Zero-shot, FB-CMA-ES (single-objective optimization in $\mathcal{Z}$), and FB-CMA-ME (QD search in $\mathcal{Z}$). FB-CMA-ME outperforms FB-CMA-ES on maximum fitness, demonstrating that diversity pressure helps escape local optima during optimization.

Results Results are shown in Figure 11. FB-CMA-ME outperforms both FB-CMA-ES and FB Zero-shot consistently across all tasks. The gap between FB Zero-shot and FB-CMA-ES confirms that closed-form inference provides a useful initialization but not an optimal solution, due to approximation errors. More strikingly, FB-CMA-ME outperforms FB-CMA-ES even on maximum fitness: by maintaining solutions across diverse behavioral niches, diversity pressure helps escape local optima that a focused single-objective optimizer converges to.

Takeaway: Diversity pressure is not only beneficial for coverage: it also improves maximum fitness by preventing premature convergence to local optima in the latent space.

E.6 Sensitivity to the Offline Dataset Quality

We investigate how the quality and composition of the offline dataset used to pretrain the BFM affect QD performance. Since the diversity of the latent space is ultimately bounded by the diversity of the training data, dataset composition is a critical factor for BFM-QD practitioners.

Experiment We vary the dataset composition on AntMaze-medium across six conditions: a **Biased** dataset restricted to the lower-left region of the maze, and five mixtures interpolating between fully random and expert data (**Random**, **75R/25E**, **50R/50E**, **25R/75E**, **Expert**). All other hyperparameters are fixed, and the QD-score is averaged across all AntMaze-medium tasks.

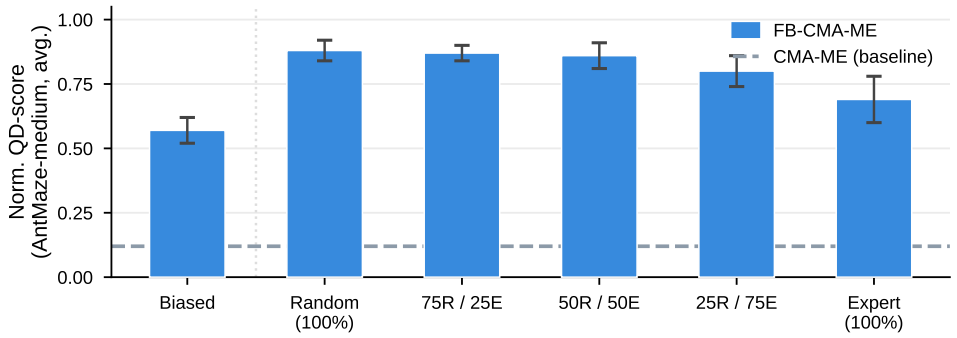


Figure 12: **Sensitivity to offline dataset quality.** Normalized QD-score on AntMaze-medium for FB-CMA-ME trained on datasets of varying composition: a spatially biased dataset (lower-left region only), and five mixtures of random and expert data. Performance degrades as expert data replaces random data, confirming that behavioral coverage matters more than trajectory quality for BFM-QD. Even the biased dataset substantially outperforms the CMA-ME baseline (dashed line).

Results Results are shown in Figure 12. Performance degrades monotonically as expert data replaces random data: expert trajectories cover only a narrow set of optimal paths, reducing the behavioral diversity of the latent space. This confirms that coverage matters more than trajectory quality for BFM-QD. Notably, even the spatially biased dataset does not collapse and substantially outperforms CMA-ME, suggesting that the BFM learns environment dynamics and motor primitives that generalize beyond the covered region.

Takeaway: Behavioral coverage in the offline dataset matters more than trajectory quality: random exploration datasets provide a richer and more diverse behavioral prior than expert ones. Even a spatially restricted random-exploration dataset yields a functional latent space that supports strong QD performance.

E.7 Additional Latent-Space Search Baselines

DDE-Elites searches a learned latent space but learns it online from the archive, and ME+Pretrain uses offline data but searches parameter space. We test baselines that combine both, and Policy Manifold Search (PoMS) [31], which runs ME in the latent space of an autoencoder over policy parameters.

Experiment We train three baselines:

- **DDE-Elites+Pretrain (TD3)** pretrains 10 independent TD3 policies offline on the BFM dataset, using the task fitness as reward (as in ME+Pretrain). A single converged TD3 run collapses to one behavior and gives the VAE nothing to encode diversity from, so we keep 10 intermediate checkpoints per run (100 policies). This population initializes the archive, and we then run DDE-Elites.
- **DDE-Elites+Pretrain (BC)** follows the same protocol, but policies are pretrained by behavioral cloning on dataset trajectories, since different trajectories represent different behaviors.
- **PoMS** is closely related to DDE-Elites, as both encode archive solutions with an autoencoder over policy parameters.

We evaluate on Walker and Cube-single over 5 seeds.

Results Table 13 reports normalized QD-score. FB-ME outperforms all baselines on both environments. On Walker, pretraining does not help: DDE-Elites+Pretrain reaches 0.61 (BC) and 0.76 (TD3), versus 0.74 for DDE-Elites, and none of the DDE-Elites or PoMS variants reaches ME. On Cube-single, pretraining raises DDE-Elites from 0.09 to 0.15 (BC) and 0.14 (TD3), which remains far below FB-ME (0.58). PoMS falls below DDE-Elites on both environments (0.63 and 0.07). These baselines imitate trajectories, optimize a single reward, or learn the latent space from the policy parameters in the archive, so their policies are bounded by the dataset, by one objective, or by the archive itself. A BFM instead learns the environment dynamics and trains policies to explore the resulting feature space.

Table 13: Normalized QD-score (mean $\pm$ std, 5 seeds).

Method	Walker	Cube
ME	0.93 ± 0.02	0.02 ± 0.01
DDE-Elites	0.74 ± 0.21	0.09 ± 0.03
ME+Pretrain	0.81 ± 0.11	0.02 ± 0.00
DDE-Elites+Pretrain (BC)	0.61 ± 0.09	0.15 ± 0.04
DDE-Elites+Pretrain (TD3)	0.76 ± 0.13	0.14 ± 0.03
PoMS	0.63 ± 0.13	0.07 ± 0.03
FB-ME (ours)	$\mathbf{0.98 \pm 0.01}$	$\mathbf{0.58 \pm 0.19}$

Takeaway: Neither offline pretraining nor a latent space learned over policy parameters is sufficient. The gains come from the structure of the BFM latent space.

E.8 Backward Inference vs. Sampling Around the Inferred Optimum

BI moves each parent toward z^*. A simpler alternative is to sample offspring around z^*.

Experiment We sample $z' = z^* + \epsilon$ with $\epsilon \sim \mathcal{N}(0, \sigma^2 I)$ for $\sigma \in \{0.5, 1.0\}$, with z^* computed from the pretraining dataset to isolate the effect of data quality early in the search. We compare to FB-BI on Walker and Cube-single over 3 seeds.

Results Table 14 reports normalized QD-score. Sampling around a fixed point confines the search to a small region of the latent space with behaviors close to that of z^*, instead of exploring the whole behavioral space. FB-BI performs far better because each parent keeps its own behavior and moves only a small step toward z^*.

Table 14: Normalized QD-score (mean $\pm$ std, 3 seeds).

Method	Walker	Cube
$z^* + \epsilon$, $\sigma = 0.5$	0.16 ± 0.03	0.09 ± 0.01
$z^* + \epsilon$, $\sigma = 1.0$	0.18 ± 0.02	0.09 ± 0.02
FB-BI	$\mathbf{0.92 \pm 0.04}$	$\mathbf{0.96 \pm 0.05}$

Takeaway: BI works because it moves each solution individually toward z^* while preserving diversity. Sampling around z^* does not.

E.9 Pretraining Data Collected by QD

We use RND to collect pretraining data. QD could also collect it, but this commits the dataset to a descriptor, whereas RND is descriptor-agnostic.

Experiment We retrain FB on data collected by our own FB-QD search, and compare the resulting QD-score to FB trained on RND data. We evaluate on Walker (forward velocity) and AntMaze (top wall).

Results Table 15 shows that QD-collected data reduces performance. Once QD finds a high-fitness elite for a cell, it refines solutions near that elite instead of covering new transitions. RND instead visits states with high prediction error and covers the state space. BFM pretraining needs data that covers the transition dynamics broadly, not only along the descriptor axes.

Table 15: QD-score of FB-ME by pretraining data source (mean $\pm$ std).

Pretraining data	Walker	AntMaze
RND	$\mathbf{16471 \pm 175}$	$\mathbf{61 \pm 45}$
QD (FB-QD)	5654 ± 12	46 ± 9

Takeaway: Broad coverage of the dynamics matters more than diversity along the test descriptor. Exploration data beats goal-directed data.

E.10 Sensitivity to the Backward Inference Step Size

Experiment We run FB-BI on Walker with step size $\alpha \in \{0, 0.005, 0.02, 0.05, 0.1, 0.2, 0.5, 1\}$ over 3 seeds. $\alpha = 0$ recovers FB-ME, and $\alpha = 1$ replaces each solution with z^*. Our default is $\alpha = 0.02$.

Results Figure 13 reports normalized QD-score. Performance is stable for $\alpha \in [0.005, 0.1]$, and our default lies inside this plateau. Beyond $\alpha = 0.1$ it degrades, since larger α pulls offspring toward the single point z^* and collapses diversity (Section 4.2).

Takeaway: FB-BI is robust to α over a wide range. Large steps collapse diversity.

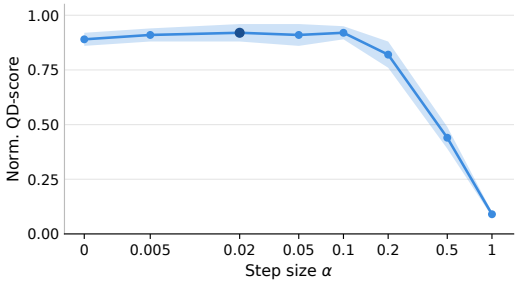


Figure 13: Normalized QD-score of FB-BI on Walker vs. step size α (mean $\pm$ std, 3 seeds).

E.11 Compute Resources

All experiments were run on NVIDIA H100 GPUs. Table 16 reports wall-clock times per run. BFM pretraining is a one-time cost shared across all tasks within a given environment, so the amortized cost per QD run is dominated by the search itself.

Table 16: Compute cost per run on a single NVIDIA H100 GPU. BFM pretraining is performed once per environment and reused across all tasks and seeds.

Phase	**Method**	**GPU-hours**
BFM Pretraining (one-time)	BYOLγ	2.67
	Lap	2.67
	TD-JEPA	2.50
	FB	2.32
	BTD-FB	3.17
QD Search (per run)	FB-ME	1.63
	FB-CMA-ME	1.83
	TD-JEPA-CMA-ME	1.87
	ME	1.28
	CMA-ME	1.47
	DCRL-ME	2.33
	DDE-Elites	2.25
	ME+Pretrain	1.28

E.12 Results Table

Tables 17–20 report the final unnormalized QD-scores (mean ± std over 5 seeds) for all methods across Walker2D, HalfCheetah, AntMaze-medium, and Cube-single.

Table 17: QD-scores on **Walker2D** (mean ± std over 5 seeds). **Bold**: best; underline: second best.

Method	Backward Vel	Forward Vel	Upright Walk	Jump
Baselines				
MAP-Elites	14801±514	15253±538	13746±493	13772±465
CMA-ME	14580±489	14177±507	13241±511	13262±481
PGA-ME	12410±491	13081±535	12124±474	12456±475
DCRL-ME	13410±413	14089±456	13714±436	13286±462
ME+Pretrain	13465±268	13576±264	12328±280	12595±268
DDE-Elites	13097±105	13723±132	10742±210	9388±165
BFM-QD (ours)				
FB-ME	15307±219	16471±175	14533±198	14641±228
FB-CMA-ME	13673±169	15986±169	14051±230	14115±188
TD-JEPA-CMA-ME	15302±272	15291±277	14268±225	13747±225
Lap-CMA-ME	9358±166	10391±154	9994±227	10043±192
BYOLγ-CMA-ME	9401±176	9932±187	10225±220	9473±204
BTD-FB-CMA-ME	16135±220	17546±183	13350±200	13307±221
FB-CMA-ME-Iso	13432±174	17087±175	15365±220	14291±194
FB-BI	**17807±227**	**18908±180**	15474±188	15328±225
FB-PGA-ME	16212±218	17906±186	**16368±187**	**16714±225**

Table 18: QD-scores on **HalfCheetah** (mean ± std over 5 seeds). **Bold**: best; underline: second best.

Method	Backward Run	Backward Walk	Forward Walk	Forward Run
Baselines				
MAP-Elites	11331±524	11874±534	11402±490	11230±520
CMA-ME	12381±507	13200±514	13724±469	12371±534
PGA-ME	10685±463	14326±473	12523±478	12636±498
DCRL-ME	12466±460	13103±479	14082±456	12324±472
ME+Pretrain	12302±291	13293±338	13618±315	12425±309
DDE-Elites	7931±120	8311±125	7981±216	7860±198
BFM-QD (ours)				
FB-ME	11589±228	11246±228	11781±224	11590±239
FB-CMA-ME	12574±165	13617±201	14096±203	12470±223
TD-JEPA-CMA-ME	12502±234	13387±229	14002±220	12495±282
Lap-CMA-ME	7341±156	8051±189	9281±184	7952±232
BYOLγ-CMA-ME	9055±159	9849±215	11214±195	8557±239
BTD-FB-CMA-ME	10578±220	10488±220	10573±217	11394±230
FB-CMA-ME-Iso	**14315±165**	**14839±200**	**14745±212**	13232±230
FB-BI	12890±233	12514±221	12360±224	13077±247
FB-PGA-ME	14105±233	12520±219	12960±223	**13745±240**

Table 19: QD-scores on **AntMaze-Medium** (mean ± std over 5 seeds). **Bold**: best; underline: second best. Superscript^{D} denotes the dense-reward variant.

Method	Energy Eff.	Top Wall	Right Wall	Reach Center	Top Wall^{D}	Right Wall^{D}
Baselines						
MAP-Elites	6±0	0±0	0±0	0±0	23±0	14±0
CMA-ME	6±0	0±0	0±0	0±0	293±0	175±0
PGA-ME	22±2	0±0	0±0	0±0	76±0	5±1
DCRL-ME	61±0	0±0	0±0	0±0	153±0	17±0
ME+Pretrain	5±0	0±0	0±0	0±0	20±0	13±0
DDE-Elites	509±0	14±3	7±4	0±0	8±1	7±4
BFM-QD (ours)						
FB-ME	3406±52	61±45	446±51	161±51	441±52	1018±57
FB-CMA-ME	3657±51	611±50	498±49	552±49	909±49	1099±49
TD-JEPA-CMA-ME	3407±88	436±10	565±56	192±21	638±66	900±50
Lap-CMA-ME	274±60	14±37	105±53	14±42	94±46	133±66
BYOLγ-CMA-ME	947±54	10±51	59±51	36±50	64±52	151±58
BTD-FB-CMA-ME	3637±55	528±48	493±42	470±39	783±57	1098±46
FB-CMA-ME-Iso	3603±44	**667±58**	**569±55**	**641±56**	**1040±43**	1208±40
FB-BI	**4074±60**	70±48	510±52	163±44	457±47	1185±54
FB-PGA-ME	3896±66	63±43	514±45	176±43	475±49	**1278±57**

Table 20: QD-scores on **Cube-Single** (mean ± std over 5 seeds). **Bold**: best; underline: second best.

Method	Energy Eff. (x, y)	Energy Eff. (x, z)	Wrist Config	Grasp Style
Baselines				
MAP-Elites	0±0	2±0	305±0	6±0
CMA-ME	4±0	2±0	443±0	29±0
PGA-ME	0±0	89±0	680±5	8±1
DCRL-ME	0±0	141±0	521±0	6±0
ME+Pretrain	3±0	1±0	152±0	4±0
DDE-Elites	875±4	174±0	0±0	0±0
BFM-QD (ours)				
FB-ME	2486±83	4240±89	1783±105	229±103
FB-CMA-ME	5368±106	5095±93	**1953±120**	1073±111
TD-JEPA-CMA-ME	5018±121	3888±106	1690±87	290±144
Lap-CMA-ME	613±73	407±93	320±114	17±100
BYOLγ-CMA-ME	657±77	1115±96	317±109	57±110
BTD-FB-CMA-ME	5192±107	4963±84	1906±113	919±110
FB-CMA-ME-Iso	**5965±114**	**5957±93**	1945±121	**1227±108**
FB-BI	2787±80	4892±80	1928±97	255±93
FB-PGA-ME	2835±74	4832±73	1887±94	255±101

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction accurately state all contributions. Claims are matched by the experiments in Section 5.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Section 6 explicitly discusses three limitations: (i) requirement of a diverse offline dataset; (ii) performance degradation with narrower datasets; (iii) the frozen BFM imposes a hard ceiling on behavioral expressivity.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [Yes]

Justification: See 4.2, and a complete proof is provided in A.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: Appendix D provides full experimental details: environments, datasets, archive configuration, evaluation budget, MLP architecture, latent dimension, all hyperparameters, number of seeds, and the variation operators.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [No]

Justification: All datasets used are publicly available, and Appendix D provides sufficient detail for reimplementation.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: All training and evaluation details are reported in Appendix D.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: All results are averaged over 5 (3 in two additional experiments) independent random seeds. Shaded regions in learning curves and error bars in bar plots denote standard deviation across seeds.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).

- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Compute resources mentioned in the Appendix E.11

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The research involves only simulated benchmarks. No human subjects, sensitive data, or dual-use concerns are present

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [N/A]

    Justification: This is foundational research on reinforcement learning algorithms using simulated environments. No involvement of real deployments, sensitive data, or high-risk technology.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.

- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: All datasets used are existing public benchmarks. No safeguards are required.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: All assets (libraries and baselines) are properly cited.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.

- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A]

    Justification: The contribution is a framework and algorithmic approach.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The paper involves no crowdsourcing and no research with human subjects. All experiments are conducted entirely in simulation.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: No human subjects are involved. IRB approval is not required.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.

- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: No LLMs needed in our research.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.