Policy Manifold Search for Improving Diversity-based Neuroevolution

Nemanja Rakicevic
Imperial College London
n.rakicevic@imperial.ac.uk

Antoine Cully
Imperial College London
a.cully@imperial.ac.uk

Petar Kormushev
Imperial College London
p.kormushev@imperial.ac.uk

Abstract

Diversity-based approaches have recently gained popularity as an alternative paradigm to performance-based policy search. A popular approach from this family, Quality-Diversity (QD), maintains a collection of high-performing policies separated in the diversity-metric space, defined based on policies' rollout behaviours. When policies are parameterised as neural networks, i.e. Neuroevolution, QD tends to not scale well with parameter space dimensionality. Our hypothesis is that there exists a low-dimensional manifold embedded in the policy parameter space, containing a high density of diverse and feasible policies. We propose a novel approach to diversity-based policy search via Neuroevolution, that leverages learned latent representations of the policy parameters which capture the local structure of the data. Our approach iteratively collects policies according to the QD framework, in order to (i) build a collection of diverse policies, (ii) use it to learn a latent representation of the policy parameters, (iii) perform policy search in the learned latent space. We use the Jacobian of the inverse transformation (i.e. reconstruction function) to guide the search in the latent space. This ensures that the generated samples remain in the high-density regions of the original space, after reconstruction. We evaluate our contributions on three continuous control tasks in simulated environments, and compare to diversity-based baselines. The findings suggest that our approach yields a more efficient and robust policy search process.

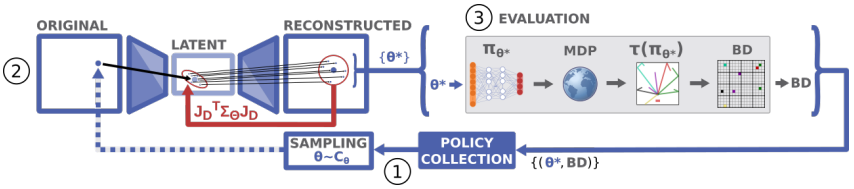


Figure 1: Components of the parameter search phase within the Policy Manifold Search approach: (1) sampling new policy parameters from the collection, (2) exploration in the learned latent space and covariance matrix scaling, (3) evaluation of the generated parameters and adding to the collection.

1 Introduction

In recent years, we have seen significant progress in tackling continuous control tasks, owing to Deep Reinforcement Learning (RL) approaches relying on gradient-based policy optimisation [15], as

Beyond Backpropagation Workshop 34th Conference on Neural Information Processing Systems (NeurIPS 2020), Vancouver, Canada.

well as the gradient-free alternative, Neurevolution [14, 16]. Optimising for task performance leads to a unique, locally-optimal solution, which corresponds to a unique set of policy NN parameters. In contrast to this, a diversity of solutions is sometimes necessary in cases where one environment can accommodate multiple tasks, or when different controllers are needed to solve and adapt to a dramatically changing environment or recover from damage [4]. Diversity can be implemented either as multiple distinct solutions [11, 3], or one solution modifiable during environment interactions [6, 13, 5, 9]. Quality-Diversity (QD) [11, 3] framework is an example of the former and introduces collections of solutions. Each different solution is assigned to the collection based on their *behaviour descriptor*, i.e. low-dimensional representation of the corresponding policy's behaviour in the environment [3]. New solutions are then generated by modifying existing ones via *mutation operators*. However, QD tends not to scale well with high-dimensional parameterisations [17, 7]. Inspired by the *manifold hypothesis* [2, 1] and prior successes in representation learning [9, 10], we hypothesise that there exists a lower-dimensional, non-linear manifold, embedded in the high-dimensional policy parameter space, which contains a high density of solutions for a particular task.

In this paper, we propose a new approach, Policy Manifold Search, that learns a manifold in the policy parameter space which is used for policy search. The goal is to find and collect a diversity of policies that work in a given environment. Focusing the search in the high solution density manifold improves the sample complexity and the coverage of diverse behaviours. What distinguishes our approach from standard representation learning approaches that learn latent embeddings, is that we do not use only point-wise embeddings but also the transformation function information. This leads to a consistent parameter search regardless of the local structure of the latent representation, by accounting for the distortions induced by the transformation function. Moreover, representation learning approaches usually focus on unique data e.g. states, images etc. Conversely, NN weights can have multiple configurations which lead to the same behaviour, thus adding complexity.

The main question we are addressing in this paper: *is the learned parameter space representation rich enough to improve policy search?* In order to answer this, we conduct ablation studies of the algorithm components, as well as comparisons to diversity-based baselines.

2 Method

We present Policy Manifold Search (PoMS), an iterative algorithm, where each iteration consists of the *parameter manifold learning* and *parameter search* phases. The former consists of obtaining a latent representation and its corresponding transformations, while the latter runs MAP-Elites (presented in Appendix A) in the learned latent parameter space to generate new diverse policies, as shown in Fig. 1. Each iteration can add new policies in the collection, which are then used to refine the latent representation in the next iteration. The PoMS pseudocode is presented in Algorithm 1 of Appendix A.

Preliminaries We consider a typical RL setting, with a deterministic environment defined as a Markov Decision Process. We define a deterministic policy π_θ, parameterised as a deep neural network, that maps the current state to the action to be taken at that state $a_t = \pi_\theta(s_t)$. The policy parameters θ are a P-dimensional set of network weights and biases, $\theta \in \mathbb{R}^P$, such that each point in the policy parameter space defines a unique policy. We use the term *original parameter space* to refer to this space of policy parameters. During an episode of length T, an agent interacts with the environment using the policy π_θ, thus generating a trajectory $\tau = \{s_i, a_i\}_1^T$. We want to distinguish how a certain deterministic policy π_{θ_k} interacts with the environment in a quantifiable way. To this end, we use the concept of a *Behaviour Descriptor* (BD) from the QD literature [3] which aims at uniquely describing the episode rollout. The BD is formalized as a mapping from a state-trajectory τ space to b-dimensional behaviour space $\mathcal{BD} : \mathrm{T} \rightarrow \mathcal{B}$. Different policies can produce the same BD, but a specific deterministic policy will map to a unique behaviour (surjective mapping).

2.1 Parameter Manifold Learning Phase

The main insight of the PoMS is learning a lower-dimensional manifold $\mathbb{R}^M$, embedded in $\mathbb{R}^P$ where $M << P$, around which a high-density of interesting (i.e. nondegenerate) policies are located. This manifold can then serve as smaller search space for a more efficient exploration. In order to obtain this manifold, we start by generating a uniformly sampled initial set of parameters, $\theta_i \sim \mathcal{U}(-1, 1)$, which are added to the the MAP-Elites collection. This collection of parameters is then used to

train an invertible dimensionality reduction model, like a deep AutoEncoder (AE). Each point in the original parameter space θ_i can be directly mapped into the corresponding point on the manifold $z_i \in \mathbb{R}^M$ using the encoder (f_E), and reconstructed back using the decoder (f_D). We use a fully-connected, symmetrical AE with a reconstruction loss: $\mathcal{L}_{AE} = \left\| \theta_i - \hat{\theta}_i \right\|$ where $\hat{\theta}_i = f_D \circ f_E(\theta_i)$. The bottleneck layer of the AE defines the latent parameter representation space. We do not apply any specific regularisation on the latent space, but simply train the AE in an unsupervised manner to achieve good reconstruction. As opposed to common training strategy on static datasets, in the case of PoMS it is not beneficial to normalise the training data before fitting the AE, as periodic additions to the collection lead to instability in the training. At each iteration of PoMS, the *parameter manifold learning* phase uses all the policy parameters from the collection and continues the training of the AE to refine the latent representation. Further details on the training procedure are given in Appendix A.

2.2 Parameter Search Phase

One of the strengths of MAP-Elites is that it constantly applies small mutations to the "elites" (the solutions contained in the grid). It is crucial to preserve this property in PoMS while performing search in the latent space. However, a small perturbation in the latent space can lead to a very large perturbation in the parameter space due to the complexity of the learned decoder. Therefore, there is a significant risk that applying mutations directly in the latent space (e.g. via Gaussian noise) will lead to an uncontrolled mutation, similar to random search.

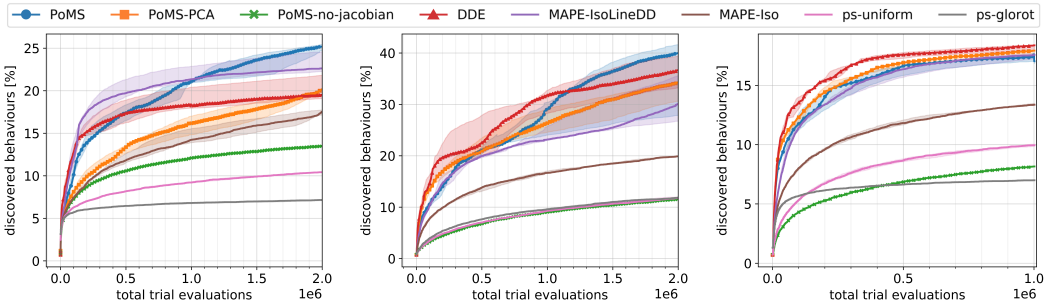
Considering the decoder Jacobian To address this issue, we propose to make the latent parameter space search heteroscedastic, as a function of the local structure. We use the Jacobian of the decoder, which gives us a linear approximation (first order Taylor expansion) of the transformation around a specific point in the latent space z, denoted as $\mathbf{J}_D(z)$. The Jacobian allows us to impose that each mutated point in the latent space $z' \sim \mathcal{N}(z_k, \Sigma_Z)$, when reconstructed, lands within a unit hyper-sphere centered aroung the reconstructed sample, in the original parameter space, i.e. $f_D(z') \sim \mathcal{N}(f_D(z_k), \sigma_\Theta \mathbb{I})$, as shown in Fig. 1. Let us define the desired covariance matrix of an isotropic unit Gaussian in the parameter space as $\Sigma_\Theta = \sigma_\Theta \mathbb{I}$, where σ_Θ is the desired radius of a hyper-sphere and $\mathbb{I} \in \mathbb{R}^{P \times P}$ is the identity matrix. The objective is to estimate the appropriate Gaussian noise covariance matrix Σ_Z applied in the latent space, as a function of Σ_Θ and the Jacobian, as $\Sigma_Z = \mathbf{J}_D^T \Sigma_\Theta \mathbf{J}_D$. For the full derivation using Taylor expansion, refer to Appendix C.

Mixing strategies for improving representation stability and diversity Performing the parameter search in the latent space has the advantage of offering a smaller search space with a high-density of different and interesting policies. However, like in most autoregressive algorithms, the AE is unable to generalise far beyond the training set data support. To overcome this problem we employ a region-based exploration strategy. If the reconstruction error of a latent point is below a threshold value ϵ_r, we perform the mutation in the latent space as explained above. Otherwise, the mutation is applied directly in the parameter space using $\mathcal{N}(\theta, \Sigma_\Theta)$. The reconstruction error threshold ϵ_r is determined heuristically based on the average reconstruction error of all the points in the collection, after the policy manifold learning phase. The pseudocode for region-based policy search is given in Algorithm 2 of Appendix A. Periodically adding solutions obtained via parameter space search helps reduce overfitting, thus helping the model generalise better. This can be regarded as a type of active learning based on model uncertainty [12].

3 Evaluation

We evaluate the algorithms on three tasks (Fig. 1 in Appendix D) in deterministic simulated environments. In each task, a robot is controlled by the policy which takes the full observation vector as input, and outputs desired actions (velocities and joint torques). We briefly describe the environments and the tasks, and defer further details to Appendix D.

Striker [observation 14D, action 3D, total behaviours 15300] is a bounded air-hockey-like environment, with the goal of controlling the striker to hit the puck so it lands on as many diverse positions as possible. **Bipedal-Walker [observation 26D, action 4D, total behaviours 12500]** is an OpenAI gym environment with the absolute location of the agent added to the observation, as it is used for the behaviour descriptors. The goal is to have the agent discover as many diverse gaits as possible. **Bipedal-Kicker [observation 26+4D, action 4D, total behaviours 10000]** extends the Bipedal-Walker task by adding a ball. The goal is for the agent to kick the ball so it travels in diverse



(a) Bipedal-Walker mixed scale (b) Bipedal-Kicker mixed scale (c) Striker mixed scale

Figure 2: Behaviour coverage plots of the compared methods, achieved in three continuous control environments. Additional results for 'normal' environments are in Fig. 2 of Appendix E.

trajectories. In addition to each environment, we introduce their ***-mix-scale** counterparts. The goal of this is to analyse the phenomenon when certain elements of the observation vector have different scales, which typically occurs in unbounded tasks. We use *behaviour coverage*, as a metric to assess the search performance, as it quantifies distinct behaviours discovered by an algorithm.

3.1 Results

The results of 'mix-scale' experiments are shown in Fig. 2, while the rest or the experimental results are in Fig. 2 of Appendix E, together with additional comparison details and analysis of the results. Our experimental evaluation aims to answer the following questions:

Q1. Are there benefits of using the learned latent space, over the original space, for policy search? In order to answer Q1, we compare the performance of the proposed PoMS approach, to the standard MAP-Elites (MAPE-Iso) which performs search in the original parameter space. Although standard MAP-Elites shows competitiveness in high-dimensional parameter space problems, PoMS systematically achieves higher behaviour coverage across the tasks. The only exception is the Striker case, where methods converge to the same performance. As a 'sanity check' we compare with two random search approaches in the parameter space, uniformly sampling policies (ps-uniform), and initialising according to Xavier-Glorot initialiser [8] (ps-glorot).

Q2. What is the effect of using the Jacobian of the decoder? We perform an ablation study to demonstrate the importance of the Jacobian scaling of the latent sampling covariance matrix, by comparing PoMS to naive latent space search (PoMS-no-jacobian). The results show that not accounting for the Jacobian, achieves significantly worse behaviour, and leads to a search which is almost random.

Q3. Is the non-linear manifold learning necessary, or does linear projection suffice? We examine the importance of non-linear manifold learning as opposed to using a linear projection of the parameter space via PCA (PoMS-PCA). This comparison highlights the intrinsic complexity of a given control problem, as locomotion tasks usually have an inherent non-linearity in the mapping of the policy outputs to the observation vector used for determining BDs, which is accounted for by PoMS.

Q4. How does PoMS compare to state-of-the-art QD approaches? We consider two recently introduced MAP-Elites algorithms: MAP-Elites with Line mutations [17] (MAPE-IsoLineDD) and MAP-Elites with data-driven encoding [7] (MAPE-DDE), as well as Diversity is All You Need (DIAYN) [5] from the deep RL literature. However, preliminary experiments showed that DIAYN is unable to scale to several thousands of skills, and takes significantly more computational time as it is not parallelizable, so we decided to exclude this algorithm from our analysis. The top performing methods, PoMS, PoMS-PCA, MAPE-DDE and MAPE-IsoLineDD, share the characteristic of focusing the search on a smaller subset of the parameter space, either as the hyper-volume of elites or via the learned manifold. This further solidifies our claims from Q1. The proposed PoMS approach outperforms the next best state-of-the-art approach by up to 5%, except in the Striker tasks where it achieves on-par performance at convergence.

4 Conclusion

In this paper, we proposed the Policy Manifold Search algorithm, that aims to discover a collection of policies with diverse behaviours. Our work is inspired by the manifold hypothesis, which assumes that the useful policies tend to concentrate near a low-dimensional manifold embedded in the original high-dimensional parameter space. Experimental evaluations of PoMS validate the benefits of using a learned manifold coupled with the Jacobian of the decoder for policy search to discover larger collections of diverse policies compared to the baselines.

Broader Impact

The motivation behind Policy Manifold Search is to open new avenues for future research on investigating representations of the parameter space and using its properties. Investigating additional regularisation and manifold learning approaches, could provide a new hybrid optimisation paradigm when combined with gradient-based approaches. In the presented form, PoMS integrated in the QD framework is useful for generating collections of diverse policies. Having such a collection is important for real-world continuously operating robots, which run in dynamically changing environment and are susceptible to damage. If damaged, robots would still be able to perform their tasks by choosing more robust alternative policies from the collection, while waiting to be repaired.

Acknowledgments and Disclosure of Funding

Nemanja Rakicevic is funded by the President's PhD Scholarship from Imperial College London.

References

[1] Yoshua Bengio, Aaron Courville, and Pascal Vincent. "Representation learning: A review and new perspectives". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2013).

[2] Lawrence Cayton. "Algorithms for manifold learning". In: *Univ. of California at San Diego Tech. Rep* (2005).

[3] Antoine Cully and Yiannis Demiris. "Quality and diversity optimization: A unifying modular framework". In: *IEEE Transactions on Evolutionary Computation* (2017).

[4] Antoine Cully et al. "Robots that can adapt like animals". In: *Nature* (2015).

[5] Benjamin Eysenbach et al. "Diversity is All You Need: Learning Skills without a Reward Function". In: *International Conference on Learning Representations*. 2019.

[6] Chelsea Finn, Pieter Abbeel, and Sergey Levine. "Model-agnostic meta-learning for fast adaptation of deep networks". In: *International Conference on Machine Learning*. 2017.

[7] Adam Gaier, Alexander Asteroth, and Jean-Baptiste Mouret. "Discovering Representations for Black-box Optimization". In: *Genetic and Evolutionary Computation Conference*. 2020.

[8] Xavier Glorot and Yoshua Bengio. "Understanding the difficulty of training deep feedforward neural networks". In: *International Conference on Artificial Iintelligence and Statistics*. 2010.

[9] Karol Hausman et al. "Learning an Embedding Space for Transferable Robot Skills". In: *International Conference on Learning Representations*. 2018.

[10] Alexandre Péré et al. "Unsupervised Learning of Goal Spaces for Intrinsically Motivated Goal Exploration". In: *arXiv preprint arXiv:1803.00781* (2018).

[11] Justin K Pugh, Lisa B Soros, and Kenneth O Stanley. "Quality diversity: A new frontier for evolutionary computation". In: *Frontiers in Robotics and AI* (2016).

[12] Nemanja Rakicevic and Petar Kormushev. "Active learning via informed search in movement parameter space for efficient robot task learning and transfer". In: *Autonomous Robots* (2019).

[13] Andrei A. Rusu et al. "Meta-Learning with Latent Embedding Optimization". In: *International Conference on Learning Representations*. 2019.

[14] Tim Salimans et al. "Evolution strategies as a scalable alternative to reinforcement learning". In: *arXiv preprint arXiv:1703.03864* (2017).

[15] John Schulman et al. “Proximal policy optimization algorithms”. In: *arXiv preprint arXiv:1707.06347* (2017).

[16] Felipe Petroski Such et al. “Deep neuroevolution: Genetic algorithms are a competitive alternative for training deep neural networks for reinforcement learning”. In: *arXiv preprint arXiv:1712.06567* (2017).

[17] Vassilis Vassiliades and Jean-Baptiste Mouret. “Discovering the elite hypervolume by leveraging interspecies correlation”. In: *Genetic and Evolutionary Computation Conference*. 2018.

Appendix A Autoencoder training procedure and data initialisation

The Policy Manifold Search algorithm, as well as other MAP-Elites-based baselines with latent space used for comparison, run in loops. Each loop consists of a parameter search phase and parameter manifold learning phase (as shown in Algorithm 1). The parameter search phase consists of 100 MAP-Elites iterations, and each iteration has a budget of 200 policy samples.

Algorithm 1: Policy Manifold Search

for *nloop* ***in*** *PoMS_loops* **do**
  `// parameter search phase`
  for *niter* ***in*** *MAP-Elites_iterations* **do**
    if *nloop* == *0* ***and*** *niter* == *0* **then**
      $\boldsymbol{\theta}^{SEL} \sim \mathcal{U}(-1, 1)$
    else
      $\boldsymbol{\theta}^{SEL} \sim \mathrm{C}_{\boldsymbol{\theta}}$
      $\boldsymbol{\theta}^{MUT} = \text{mutation\_operator}(\boldsymbol{\theta}^{SEL}, f_E, f_D)$
    end
    $\boldsymbol{\tau}^{MUT} = \text{environment\_eval}(\boldsymbol{\theta}^{MUT})$
    $\boldsymbol{b}^{MUT} = \mathcal{BD}(\boldsymbol{\tau}^{MUT})$
    $\mathrm{C}_{\boldsymbol{\theta}}[\boldsymbol{b}^{MUT}] \leftarrow \boldsymbol{\theta}^{MUT}$
  end
  `// parameter manifold learning phase`
  for $\boldsymbol{\theta}_{batch}$ ***in*** $\mathrm{C}_{\boldsymbol{\theta}}$ **do**
    $\hat{\boldsymbol{\theta}}_{\text{batch}} = f_D \circ f_E(\boldsymbol{\theta}_{\text{batch}}; \xi)$
    $\underset{\xi}{\operatorname{argmin}}\, \mathcal{L}_{AE} = \left\| \boldsymbol{\theta}_{\text{batch}} - \hat{\boldsymbol{\theta}}_{\text{batch}} \right\|$
  end
end

Algorithm 2: Region-based search

Input: $\mathrm{C}_{\boldsymbol{\theta}}, \Sigma_\Theta$
Output: $\mathrm{C}_{\boldsymbol{\theta}}$
for *num_loops* **do**
  $\epsilon_{recn} = \frac{1}{|\mathrm{C}_{\boldsymbol{\theta}}|} \sum_i^{|\mathrm{C}_{\boldsymbol{\theta}}|} \left\| \theta_i - \hat{\theta}_i \right\|$
  for *map-elites_iterations* **do**
    $\boldsymbol{\theta}^{SEL} \sim \mathrm{C}_{\boldsymbol{\theta}}$
    $\boldsymbol{\theta}^{MUT} = \emptyset$
    for θ_i ***in*** θ^{SEL} **do**
      if $\left\| \theta_i - \hat{\theta}_i \right\| < \epsilon_{recn}$ **then**
        $z_i = f_E(\theta_i)$
        $\mathbf{J} = \text{Jacobian}(f_D; z_i)$
        $\Sigma_\Phi = \mathbf{J}^T \Sigma_\Theta \mathbf{J}$
        $z_i^{MUT} = z_i + \mathcal{N}(0, \Sigma_Z)$
        $\theta_i^{MUT} = f_D(z_i^{MUT})$
      else
        $\theta_i^{MUT} = \theta_i + \mathcal{N}(0, \Sigma_\Theta)$
      end
      $\boldsymbol{\theta}^{MUT} \leftarrow \theta_i^{MUT}$
    end
    $\mathrm{C}_{\boldsymbol{\theta}} \leftarrow \text{unique\_bd}(\mathrm{C}_{\boldsymbol{\theta}} \bigcup \boldsymbol{\theta}^{MUT})$
  end
end

A.1 Initialising the policy collection

In the first MAP-Elites iteration of the first loop, the policy collection is still empty $\mathrm{C}_{\boldsymbol{\theta}} = \emptyset$. Therefore, it is not possible to select a sample to be mutated from the collection, so we generate the initial

sample by drawing 2000 policy samples from a uniform distribution $\boldsymbol{\theta}^{SEL} \sim \mathcal{U}(-1, 1)$, as usually done in the MAP-Elites literature [11]. This is done for all MAP-Elites-based baselines.

A.2 Autoencoder training

In order to get the latent representation of the parameter space, PoMS uses a fully-connected symmetrical Autoencoder. The specific architecture varies based on the experiment, and these are presented in detail in Appendix D.4. During training, the reconstruction loss $\mathcal{L}_{AE}$ is minimised in order to find the optimal AE parameters ξ, using the Adam optimiser with parameters: $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\varepsilon = 10^{-8}$ and learning rate of 10^{-5}. The training is ran for $2 \cdot 10^4$ epochs with batch size of 64. These parameters are fixed for all the experiments.

To improve the robustness of the optimiser, we reset the momentum variables at every loop. Moreover, 30% of each batch is used as a test set for early stopping of the training. If the slope of the line fitted to the last 100 test set values is larger than 10^{-5}, the training is stopped. We found that this improves the generalisation of the AE and reduces training time.

A.3 MAP-Elites algorithm

The MAP-Elites framework maintains a collection of multiple policy parameters, in a multi-dimensional cell-grid $\mathrm{C}_{\boldsymbol{\theta}}$, which is indexed by the behaviour index $b \in \mathcal{B}$ obtained using the behaviour descriptor. To resolve the surjective mapping, usually each cell of the grid is populated by a single highest-performing policy based on some performance metric. The aim of MAP-Elites is to fill all the cells of $\mathrm{C}_{\boldsymbol{\theta}}$ with the best possible policies through an iterative process. Each iteration consists of (i) randomly selecting a batch of individuals from the collection, (ii) applying a mutation and evaluating these modified individuals, (iii) based on the outcome of the evaluation, add the new individuals to the grid if the corresponding cell is vacant or if they outperform the currently occupying individual. In this study we focus purely on policy behaviour diversity, so the performance metric is replaced by random selection. The most common mutation operator adds an isotropic Gaussian noise $\mathcal{N}(0, \Sigma_\Theta)$, with a unit covariance matrix $\Sigma_\Theta = \sigma_\Theta \mathbb{I}$ where σ_Θ is a hyperparameter.

Appendix B Related Work

In this section we position our work within the diversity-based and representation learning literature.

Quality Diversity (QD) algorithms [29, 11] have been recently introduced as a framework that generates a collection of high-performing and diverse solutions. Most popular approaches are Multi-dimensional Archive of Phenotypic Elites (MAP-Elites) [12] and Novelty Search with Local Competition [23], which differ in how they select and maintain a collection of behaviours. In this work, we focus on MAP-Elites due to its simplicity of implementation and proven performance in various applications, such as video games [17], robotics [12], routing problems [37] etc. MAP-Elites usually uses simple low-dimensional parameterised controllers, such as periodic function generators, central pattern generators, small evolved networks or low-level controllers [12]. Recently, [9] proposed to scale MAP-Elites to Deep Neuroevolution and applied it to more complex environments. The policy search process is performed directly in the high-dimensional NN parameter space.

Diversity of policies in Deep RL is usually considered as a stepping stone for improving exploration, rather than for maintaining a collection of solutions. Such diversity is achieved via action-space noise [34], parameter-space noise [28, 15], or enforced via additional entropy [39, 19], intrinsic motivation [27, 2] or state-visitation [3, 36] terms in the reward function. Diversity is also essential in hierarchical RL, which uses a policy network conditioned on a task sampled from a discrete [20, 13] or continuous [35] distribution. Maximum Entropy (MaxEnt) RL framework [18, 1] focuses on maximising a state or action distribution entropy term added to the reward function, which incentivises the discovery novel state-trajectories, i.e. behaviours. The methods in [13, 35] focus purely on discovering diverse skills, i.e. behaviours, and do not consider extrinsic environment rewards. Discovery of diverse state-trajectories is encouraged by maximising the mutual information between states and skills, in addition to maximizing the state entropy.

While QD and MaxEnt RL frameworks both aim to promote behaviours diversity, they differ in how the diversity is maintained and in the definition of skills, i.e. behaviours. MaxEnt RL generates a

single, task-conditioned policy, that exhibits different behaviours depending on the task, while QD learns one policy for each of the behaviours. In QD, behaviours are enumerated and quantifiable via behaviour descriptors derived from the state-trajectory [11] or learned [25, 10], while MaxEnt RL methods quantify behaviour diversity by evaluating the entropy of the task context distribution [18, 13, 1, 20]. This allows QD to consider a significantly larger number of diverse behaviours.

Manifold Learning is the process of obtaining a lower-dimensional representation, i.e. manifold, embedded in the original high-dimensional input space. The *manifold hypothesis* assumes that a high density of datapoints is located in the vicinity of the manifold. This notion has been thoroughly examined as *representation learning* in machine learning [4] and RL [8, 13, 33], with important insights on how to exploit the structure of the manifold to improve robustness [32, 31]. These approaches deal with input spaces such as environment observations, images, graphs etc. which have different structural properties compared to NN parameter spaces. Recently, the concept of manifold learning, or *intrinsic dimension* of the parameter space, has been examined for NNs [24] and specifically NN-parameterised policies [30, 21, 7]. To the best of our knowledge, [7] is the only work besides ours that learns representations of NN policy parameters. They generate new policies via simple interpolation within the latent space without considering its structure.

Two recently introduced MAP-Elites based approaches present ideas which are close to the notion of parameter manifolds. In [38] the authors examine the idea of hypervolumes in the high-dimensional parameter space containing high-performing and diverse solutions, while [16] exploits the reconstruction error as the perturbation needed for policy search.

Appendix C Derivation of Jacobian scaling

Let us assume $\theta \in \Theta \subset \mathbb{R}^P$ and $z \in Z \subset \mathbb{R}^M$ to be Gaussian in the parameter and latent space respectively, such that $\theta \sim \mathcal{N}(\mu_\Theta, \Sigma_\Theta)$ and $z \sim \mathcal{N}(\mu_Z, \Sigma_Z)$.

We assume a non-linear, vector-valued function $f_D : \mathbb{R}^M \to \mathbb{R}^P$ which maps the latent space to the original parameter space.

We can get a linear approximation $\hat{f}_D$ in a point μ_Z by performing a first-order Taylor expansion:

$$\begin{aligned}
\hat{f}_D(z) &\approx f_D(\mu_z) + \sum_i^M \frac{\partial f_D(z)}{\partial z_i}\bigg|_{z=\mu_Z} (z_i - \mu_{Zi}) \\
&= f_D(\mu_Z) + \begin{bmatrix} \nabla f_D(z)_1 \\ \nabla f_D(z)_2 \\ \dots \\ \nabla f_D(z)_P \end{bmatrix} (z - \mu_Z) \\
&= \mu_\Theta + \mathbf{J}_D(z - \mu_Z)
\end{aligned} \tag{1}$$

where $\mathbf{J}_D$ is the Jacobian matrix of f_D at μ_Z:

$$\mathbf{J}_D(\mu_Z) = J_D(\mu_Z)_{ij} = \frac{\partial f_D(\mu_Z)_i}{\partial \mu_{Z_j}} \tag{2}$$

where indices i and j refer to the corresponding elements of the reconstructed or latent parameter vector, respectively.

Further, if $\hat{\theta} = \hat{f}_D(z)$ its expected value can be obtained as:

$$\begin{aligned}
\mathbb{E}[\hat{\theta}] &= \mathbb{E}\left[\mu_\Theta + \mathbf{J}_D(z - \mu_Z)\right] \\
&= \mathbb{E}[\mu_\Theta] + \mathbb{E}\left[\mathbf{J}_D(z - \mu_Z)\right] && \text{(expected values of a sum is the sum of expected values)} \\
&= \mu_\Theta + \mathbb{E}\left[\mathbf{J}_D z\right] - \mathbb{E}\left[\mathbf{J}_D \mu_Z\right] && \text{(expectation of a constant is a constant)} \\
&= \mu_\Theta + \mathbf{J}_D \mathbb{E}[z] - \mathbf{J}_D \mathbb{E}[\mu_Z] \\
&= \mu_\Theta + \mathbf{J}_D \mu_Z - \mathbf{J}_D \mu_Z \\
&= \mu_\Theta
\end{aligned} \tag{3}$$

We can obtain Σ_Θ based on Σ_Z. We start with the standard equation for covariance:

$$\begin{aligned}
\Sigma_\Theta &= \mathbb{E}\left[(\hat{\theta} - \mathbb{E}[\hat{\theta}])(\hat{\theta} - \mathbb{E}[\hat{\theta}])^T\right] \\
&= \mathbb{E}\left[(\mu_\Theta + \mathbf{J}_D(z - \mu_Z) - \mu_\Theta)(\mu_\Theta + \mathbf{J}_D(z - \mu_Z) - \mu_\Theta)^T\right] && \text{(using (1) and (2))} \\
&= \mathbb{E}\left[(\mathbf{J}_D(z - \mu_Z))(\mathbf{J}_D(z - \mu_Z))^T\right] \\
&= \mathbb{E}\left[\mathbf{J}_D(z - \mu_Z)(z - \mu_Z)^T\mathbf{J}_D^T\right] \\
&= \mathbf{J}_D\mathbb{E}\left[(z - \mu_Z)(z - \mu_Z)^T\right]\mathbf{J}_D^T && \text{(covariance definition for z)} \\
&= \mathbf{J}_D\Sigma_Z\mathbf{J}_D^T
\end{aligned} \tag{4}$$

By rearranging the previous equation we get:

$$\Sigma_Z = \mathbf{J}_D^T\Sigma_\Theta\mathbf{J}_D \tag{5}$$

Appendix D Experiment Details

D.1 Environement details

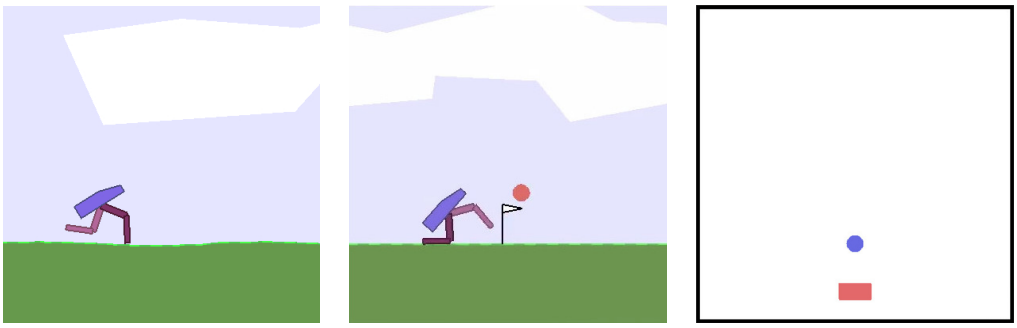
In this section we give implementation details of the environments used in the experiments, including the observation and action vectors used within the policy.

Striker (Fig. 3c) is an air-hockey-based environment implemented in Box2D [6]. The arena is bounded by four walls to the size of 100x100 units, created to be proportional to the striker size (5x2.5 units). The puck has a radius of 2.5 units. The input to the policy is a 14D observation vector, consisting of striker's x, y position and angle ϕ, the puck's x, y position, as well as their corresponding velocities, and puck-wall distances for each of the walls. The output of the policy is a 3D action vector that controls the striker's x, y and rotational velocities. The striker is allowed to move for 100 episodes and then stops to avoid continuous interaction with the puck, while the puck moves until it stops due to the damping effects. We define the behaviour descriptor as a 3D grid: D1-2: final x,y position of the puck, with 30 bins for each dimension; D3: wall(s) the puck bounced off during the trial, and has one of 17 possible values (no wall, south, east, north, west, and second order combinations).

Bipedal-Walker (Fig. 3a) is a standard OpenAI gym [5] Box2D environment. The original observation vector has 24 elements, which include the robot hull angle, horizontal, vertical and angular velocities, joints angles and angular velocities, legs-ground contact information, and 10 lidar rangefinder measurements. However, we also added the absolute coordinates of the robot hull, thus creating a 26D observation vector. The 4D action vector is unaltered and provides torques for each of the leg joints. At the start of the episode, the robot is placed in the middle of the terrain so it can walk either forward or backward in as many diverse ways as possible. The episode is limited to 500 steps. The 4D behaviour descriptor grid is based on the agent's absolute positions and leg-ground contacts during the episode: D1: average hull y-coordinate, 5 bins; D2: final hull x-coordinate, 100 bins; D3-4: proportion of time left and right legs spent in contact with ground, 5 bins each.

Bipedal-Kicker (Fig. 3b) extends the Bipedal-Walker task by adding a ball. Therefore, the observation vector is extended with the ball x, y position and velocities, making it 30D. The action output is the same. Since the goal is to have a diversity of ball ballistic trajectories, we make the terrain flat to avoid biasing the outcomes to local valleys. In order to facilitate kicking, as the agent does not have a foot, at the start of the episode the ball is dropped from a small height so the agent can hit it. Similarly to Striker, the agent is allowed to act 100 steps before the actions are set to 0, in order to have only one kick per episode. The behaviour descriptor is a 2D based grid, based on the ball trajectory, as a usual way of defining a 2D ballistic trajectory: D1: final ball x-coordinate, 200 bins; D2: maximum ball y-coordinate achieved during the episode, 50 bins.

In contrast to Bipedal-Walker, the Bipedal-Kicker and Striker tasks contain a ball, which is an external object manipulated by the agent. This adds complexity to the task as certain elements of the observation vector can vary independently of the agent's actions. Another distinction we note is between Striker which is bounded, and Bipedal-* environments which are unbounded. This causes



(a) Bipedal-Walker (b) Bipedal-Kicker (c) Striker

Figure 3: Screenshots of the three continuous control tasks used for experimental evaluation. The environments are implemented in Box2D [6].

certain elements of the observation vector to have different scale than the rest. Usually the difference in scale can be normalised, but in unbounded environments this is not straightforward. We introduce this distinction in the environments in order to evaluate its influence on the behaviour diversity.

To emphasise this phenomena, we implement two versions of each of the environments: We extend the Striker described above to create **Striker mix-scale** by scaling elements of the observation vector which are external to the agent (exteroceptive sensory input such as: ball position and velocity, and striker absolute position) by a factor of 100. To get **Bipedal-Walker mix-scale**, we multiply only the agents absolute position elements by 100, as there is no ball. This creates an interesting example that has only two scaled elements in the 14D observation vector, while this ratio is much larger in the other environments. Since the original Bipedal-Kicker has more unbounded elements than Bipedal-Walker and Striker, we consider this to be the **Bipedal-Kicker mix-scale** version. For the 'normal' version we normalise the external elements of the original version to the [0, 1] range by arbitrarily defining the limits in the environment. We hypothesise that having mixed scales of elements in the observation vector is propagated further to the agent policy parameters, and exploiting this structure in the parameter space affects the performance of the policy search algorithms.

D.2 Agent policy implementation

The policy of the agent is kept the same for all environments and is inspired by the Proximal Policy Optimisation [34] policy implementation for continuous control tasks. The policy is defined as a fully-connected neural network with two hidden layers of 32 neurons each, with tanh activation functions. The output layer has a linear activation function. The input and output size vary depending on the observation and action vector sizes, which are specific to each environment.

D.3 Metric calculation

As described in Section 3, we evaluate the algorithms on three tasks in a deterministic simulated environment. In order to achieve determinism in the metric, e.g. surjective mapping between the agent policy and the corresponding behaviour descriptor, we enable repeatability by fixing the environment initial state as customary in the QD literature [12, 26, 16].

Alternatives to a deterministic behaviour framework are presented in [14, 9, 22]. We will consider in future works how these approaches can be combined with PoMS.

For the experiments to have statistical significance, each run is executed with 5 different seeds for parameter initialisation. For each experiment, we show the median of the 5 seeds and 25th and 75th percentile. The x-axis of the behaviour coverage plots shows the total cumulative number of episode rollouts.

D.4 Algorithm hyperparameters

Here we describe the hyperparameters used in each of the algorithms, for different experiments. All algorithms based on MAP-Elites (PoMS versions, MAPE-Iso, MAPE-IsoLineDD and DDE) run

100 iterations of MAP-Elites with a budget of 200 samples during the parameter search phase. The ps-uniform and ps-glorot versions run for the same total amount of samples as other algorithms, while the progress is saved every 2000 samples.

PoMS
The proposed PoMS algorithm has 3 main hyperparameters that need to be tuned: AE architecture, latent space dimension (LD) and Σ_Θ. The AE is symmetric, i.e. both encoder and decoder have one hidden layer with ELU activation, and we vary the number of nodes (HD). The activation function of the bottleneck layer forming the latent space is linear, same as for the output layer of the decoder. We provide the values used for each of the experiments in the table below:

Environment	**HD**	**LD**	$\mathbf{\Sigma_\Theta}$
Striker	100	50	0.1
Striker mix-scale	100	20	0.01
Bipedal-Walker	100	50	0.1
Bipedal-Walker mix-scale	100	100	0.1
Bipedal-Kicker	100	100	0.01
Bipedal-Kicker mix-scale	100	50	0.01

PoMS-no-jacobian
The PoMS-no-jacobian has the same hyperparameters as PoMS for each of the experiments, in order to do the proper ablation study. Since Σ_Z is not obtained using the Jacobian scaling, we do not use a fixed matrix, rather, we dynamically update Σ_Z based on the current manifold representation. We consider the ranges of the latent space parameters per latent dimension r_Z, in order to scale a unit covariance matrix $\Sigma_Z = r_Z^T \mathbb{I}$. This results in a search which scales the importance of each of the latent dimensions based on its range. As we can see from the results, two issues arise with this approach: (i) range does not equal importance (solution density), (ii) applying the inverse transformation applies an additional distortion which can lead to undesirable values, because $\theta' \neq f_D(z')$, where $z' \sim \mathcal{N}(\mu_Z, \Sigma_Z)$ and $\theta' \sim \mathcal{N}(f_D(\mu_Z), \Sigma_\Theta)$, even if $\Sigma_\Theta = \Sigma_Z = \mathbb{I}$.

PoMS-PCA
The PoMS-PCA needs the latent dimension (LD) and Σ_Θ hyperparameters. These values are kept the same as in the corresponding PoMS version, in order to perform a proper ablation study.

DDE
We separate DDE hyperparameters into AE architecture hyperparameters and mutation operator specific hyperparameters. The former are kept the same as architectures of PoMS for each of the experiments, while the latter are kept the same as in the original paper [16]. Instead of running a fixed window for the multi-armed bandit upper confidence bound operator selector, we maintain a moving average.

MAPE-IsoLineDD
This algorithm has two main hyperparameters related to the weighing of the isometric and directional components of the mutation operator, and they are kept the same as in the original paper [38].

MAPE-Iso
The standard MAPE-Iso algorithm has only Σ_Θ that needs tuning. We set this to Σ_Θ=0.1 as this achieved the best performance for MAPE-Iso across the experiments.

ps-uniform
The performance of ps-uniform changes based on the range from which the policy parameters are sampled. We keep this range to [-1, 1]. It is important to mention that we examined other symmetric ranges as well, such as: [-0.1, 0.1], [-10, 10], [-100, 100], [-1000, 1000]. The final performance was the best when using [-1, 1], as the values sampled in this region are diverse enough for the tanh activation in the policy network to propagate non-saturated values. Although this would be expected for [-0.1, 0.1] as well, the obtained parameters were not diverse enough.

ps-normal
There are no hyperparameters to tune for this baseline.

Appendix E Evaluation details and discussion

E.1 Evaluation

Here we provide further details on the experiment organisation and the methods used for comparison, based on the main four questions:

Q1. Are there benefits of using the learned latent space, over the original space, for policy search?
Q2. What is the effect of using the Jacobian of the decoder?
Q3. Is the non-linear manifold learning necessary, or does linear projection suffice?
Q4. How does PoMS compare to state-of-the-art QD approaches?

In order to answer Q1, we compare the performance of the proposed PoMS approach, to the standard MAP-Elites (MAPE-Iso) which perform search in the original parameter space.

We perform an ablation study to address Q2 and Q3. The first aim is to demonstrate the importance of the Jacobian scaling of the latent sampling covariance matrix. The alternative approach would be a naive latent space search where the latent sampling covariance matrix Σ_Θ is determined by the current ranges of the latent representations (PoMS-no-jacobian). The second aim is to examine the importance of non-linear manifold learning (AE) as opposed to using a linear projection of the parameter space via PCA (PoMS-PCA).

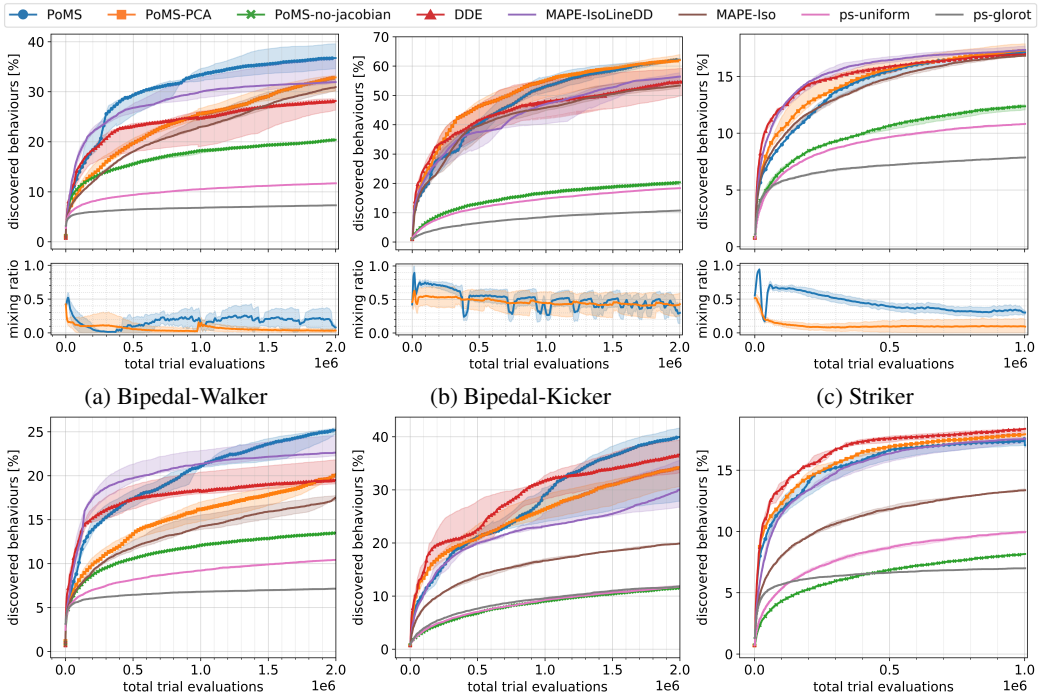
Regarding Q4, we consider two recently introduced MAP-Elites algorithms: MAP-Elites with Line mutations [38] (MAPE-IsoLineDD) that uses a mix of isotropic and directional Gaussian operators allowing for an adaptive search that implicitly explore the hyper-volume of the elites, which is similar to the learned manifold in PoMS, but in the original parameter space. Also, MAP-Elites with data-driven encoding [16] (MAPE-DDE) that combines the line mutation with a reconstruction-crossover operator based on an AE trained on the policies contained in the collection, like PoMS. The AE hyperparameters of MAPE-DDE are set equal to those of PoMS in the experiments. Additionally, we look at the Diversity is All You Need (DIAYN) algorithm [13] from the deep RL literature, as it aims to maximise the skill diversity of a single policy conditioned on a discrete latent distribution. However, preliminary experiments showed that it is unable to scale to several thousands of skills (like MAP-Elites and PoMS). In the time required by MAP-Elites and PoMS to perform several millions of iterations, we only managed to run DIAYN over 2000 iterations, without observing any promising results. Therefore, we decided to exclude this algorithm from our analysis.

E.2 Discussion

Below, we discuss the behaviour coverage results from Fig.4 achieved by the compared algorithms in more detail.

A1. Learned latent space vs original parameter space search. The first conclusion we note from the experiments, is that the standard MAP-Elites algorithm (MAPE-Iso) is competitive in high-dimensional parameter space problems, which has not been sufficiently investigated in previous work. Comparing the proposed PoMS approach with MAPE, PoMS systematically achieves higher behaviour coverage across the tasks. The only exception is the Striker case (Fig. 4c and 4f), where methods converge to the same performance. This can be attributed to the simplicity of the task, as the policy outputs can directly influence the planar movement of the striker and by extension the puck, while this connection is more complex within bipedal locomotion. Even though MAPE-IsoLineDD operates in the parameter space, it exploits the notion of a hyper-volume of elites which makes it more efficient than MAPE-Iso. However, besides the Striker tasks where it reaches equal asymptotic performance, it converges to a lower behaviour coverage compared to PoMS. By definition of the line mutation, the MAPE-IsoLineDD usually performs well when the hyper-volume is convex. When this assumption does not hold, a more involved transformation is needed which PoMS realises via manifold learning. These examples validate the benefits of using a learned latent representation of the parameter space for policy parameter search.

A2. Contribution of using the Jacobian of the decoder. By comparing PoMS and PoMS-no-jacobian, we can clearly see that generating new policies through random parameter perturbation in the latent space and reconstruction, not accounting for the Jacobian, achieves significantly worse behaviour diversity. As hypothesised, this leads to a search that is almost random, which explains the results where PoMS-no-jacobian shows similar performance as random search (Fig. 4b, 4e, 4f).



(a) Bipedal-Walker (b) Bipedal-Kicker (c) Striker

(d) Bipedal-Walker mixed scale (e) Bipedal-Kicker mixed scale (f) Striker mixed scale

Figure 4: Behaviour coverage and mixing-ratio plots of the compared approaches achieved in three continuous control environments. The markers on lines corresponding to approaches using a latent representation show points at which the latent representation is updated.

A3. Linear vs non-linear representations. The difference in performance between PoMS and PoMS-PCA, speaks mostly about the intrinsic complexity of the given task control problem. Tasks in which locomotion is involved have an intrinsic non-linearity in the mapping of the policy outputs and actual motions contained in the observation vector, which is used to determine behaviour descriptors. This explains the performance gap in tasks other than Striker. However, performance equal to PoMS seen in Bipedal-Kicker (Fig. 4b) is due to the observations being normalized which keep the parameters well behaved and easier to obtain a useful linear projection, while this does not hold in Striker-Kicker mix-scale (Fig. 4e) which explains the drop in performance

A4. State-of-the-art performance comparison. The top performing methods are PoMS, PoMS-PCA, DDE and MAPE-IsoLineDD, where the characteristic they have in common is focusing the search in the hyper-volume of elites or the learned manifold. This further solidifies claims from A1. Moreover, comparing 'normal' and mix-scale versions of the environments (top and bottom row of Fig. 4), we can see the robustness of the approaches using some form of a manifold for search. As we introduce a large variation of scale among the elements of the observation vector, this causes a reduction in the overall achieved behaviour diversity. This drop in performance is much higher in purely parameter space search methods such as MAPE-Iso, ps-uniform and ps-normal. The proposed PoMS approach outperforms the next best state-of-the-art approach by up to 5%, except in the Striker tasks where it achieves on-par performance at convergence, and Bipedal-Kicker where PoMS and PoMS-PCA have similar performance which is 5% better than the next best MAPE-IsoLineDD.

Mixing ratio Below the corresponding behaviour coverage plots, we show the mixing ratio plots. For the PoMS and PoMS-PCA, the plots represent the averaged ratio of samples generated in the latent versus the parameter space during the training. Mixing ratio of 1 means that all of the samples are taken in the parameter space, and vice versa. The first loop of the algorithm draws with a mixing ration of 0.5 and subsequently the ratio changes based on the mean reconstruction error as explained in the Method section. From mixing ratios we can see that in the beginning there is usually a spike to

take more parameter space samples. This is due to the fact that there are many sampled points with a high reconstruction error in the beginning, because initially the AE was fitted on the small amount of data and needs more data diversity - thus it 'explores'. The mixing ratio slowly decreases in favor of the latent space samples, with several salient 'dips' which slightly correlate to rises in behaviour discovery. This can be interpreted as the algorithm 'exploiting' the latent representation. This is not evident with PCA as its representations tend to be more rigid and do not change often with new data.

Appendix F Achieved behaviour diversity

In Tables 1-6, we show the achieved diversity of trajectories and policy collection coverage during the experiments, for each of the compared algorithms. We are showing the data corresponding to the median curves from the main results graph (Figure 3. in the main manuscript). The colouring of trajectories is based on a specific dimension of the policy collection cell-grid:

For Striker and Striker mix-scale experiments (Tables 1 and 2), each colour represents the index of D3, which describes the wall(s) with which the puck collided. The corresponding grid plot separates the x-y area based on this dimension.

For Bipedal-Walker and Bipedal-Walker mix-scale experiments (Tables 3 and 4), each colour represents the index of D3, which describes left leg's duty factor, i.e. proportion of time the left leg spent in contact with the ground.

For Bipedal-Kicker and Bipedal-Kicker mix-scale experiments (Tables 5 and 6), each colour represents the index of D2, which describes the max y-coordinate of the ball achieved during an episode.

Table 1: Puck trajectories and policy collections for Striker experiments.

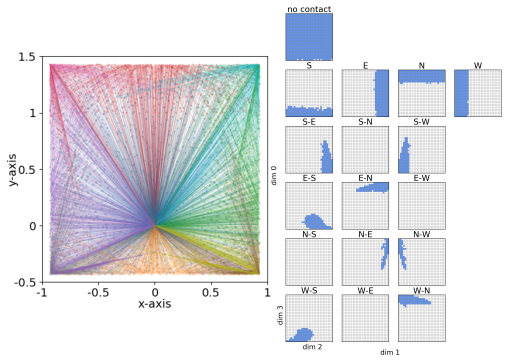
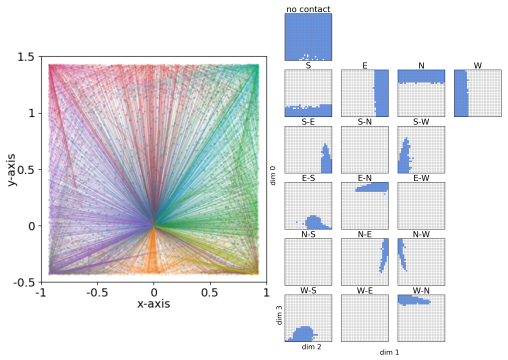
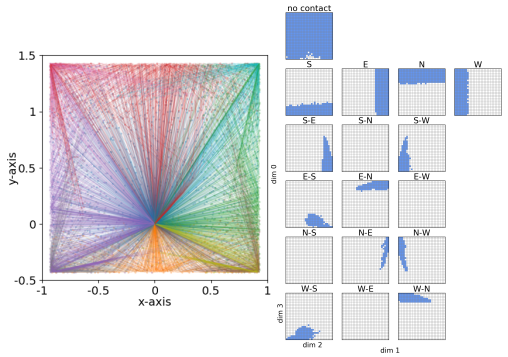
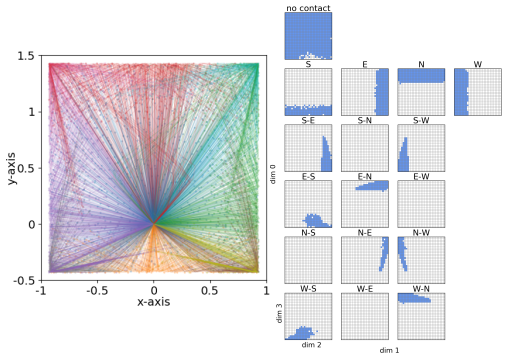
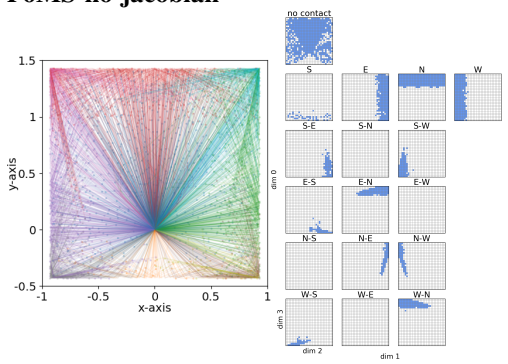
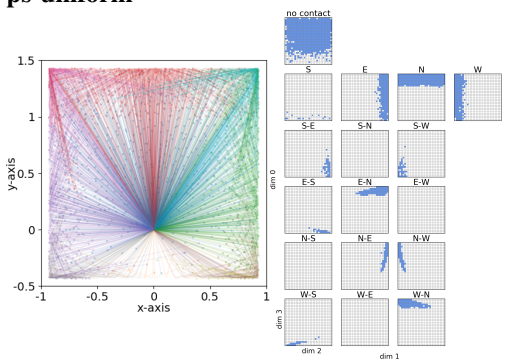
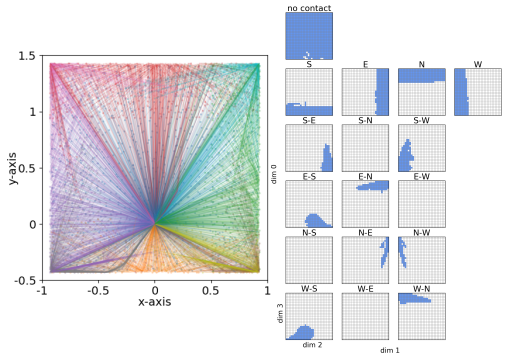
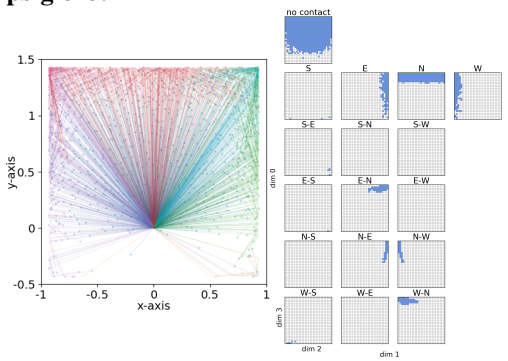
Algorithm	Striker
PoMS	
MAPE-IsoLineDD	
PoMS-PCA	
MAPE-Iso	
PoMS-no-jacobian	
ps-uniform	
DDE	
ps-glorot	

Table 2: Puck trajectories and policy collections for Striker mix-scale experiments.

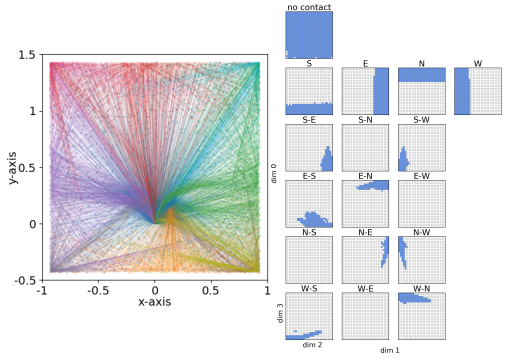
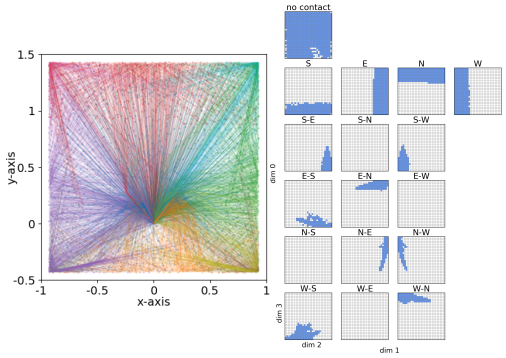
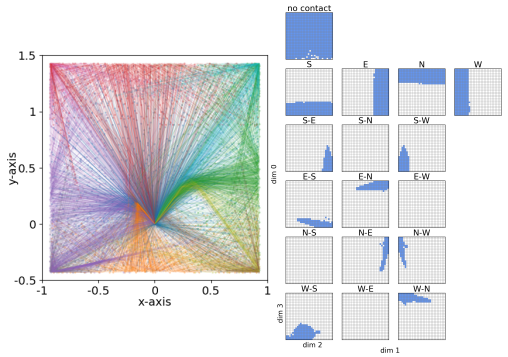
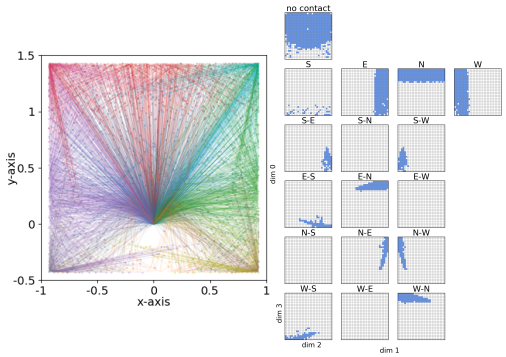
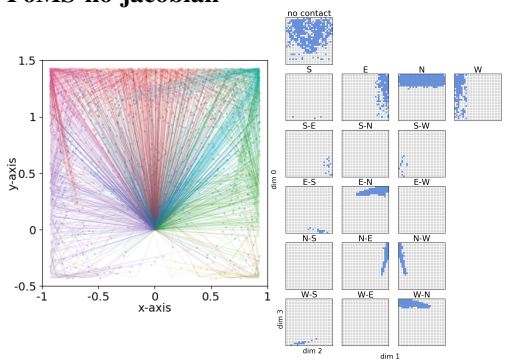
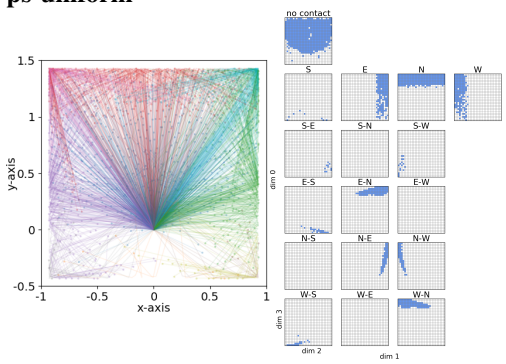
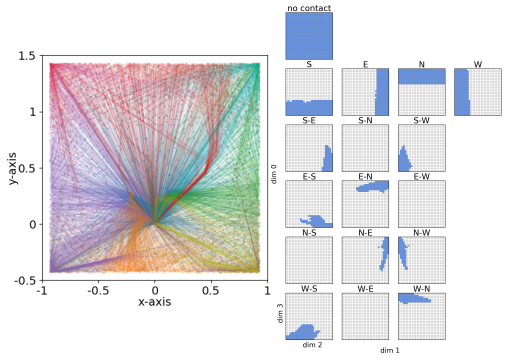
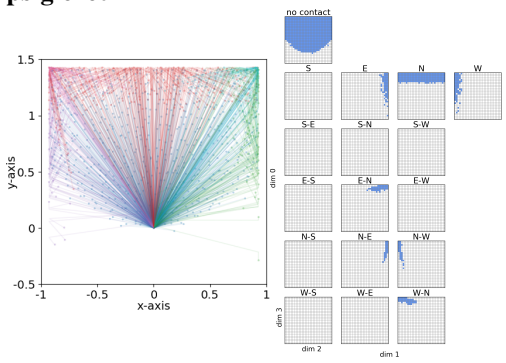
Algorithm	Striker mix-scale
PoMS	
MAPE-IsoLineDD	
PoMS-PCA	
MAPE-Iso	
PoMS-no-jacobian	
ps-uniform	
DDE	
ps-glorot	

Table 3: Robot trajectories and policy collections for Bipedal-Walker experiments.

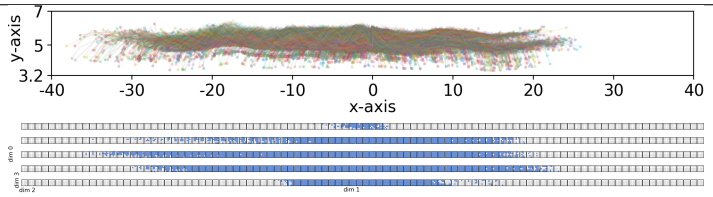
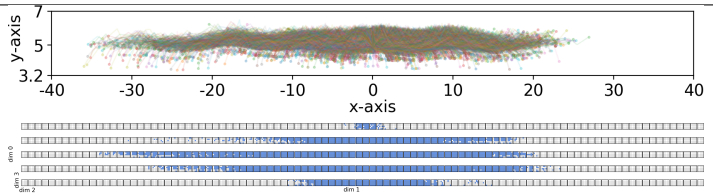
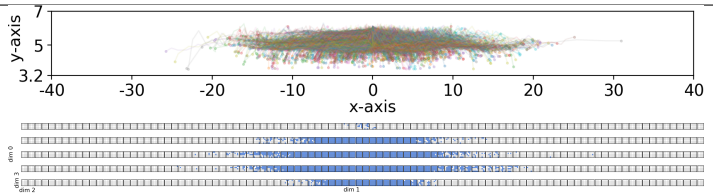
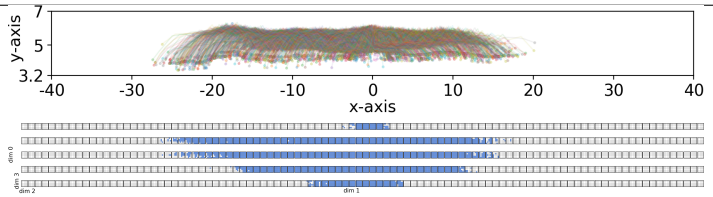
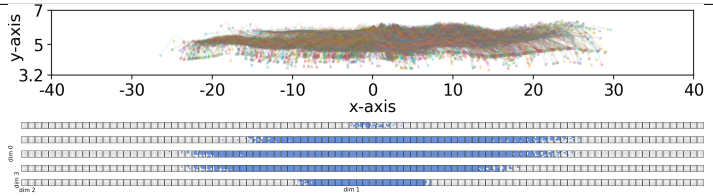
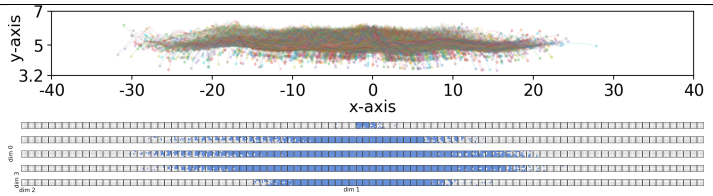
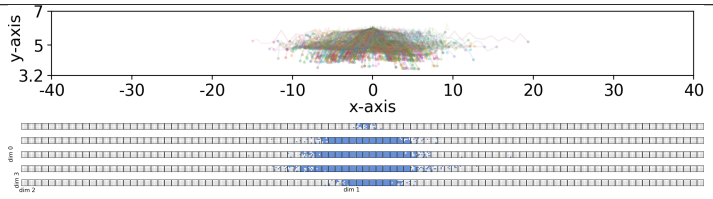
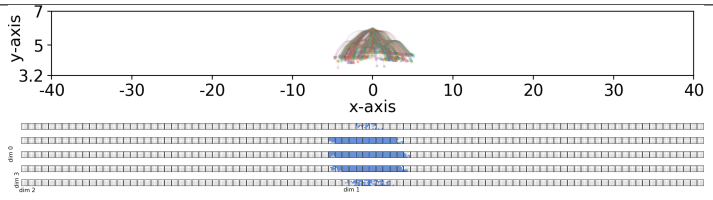
Algorithm	Bipedal-Walker
PoMS	
PoMS-PCA	
PoMS-no-jacobian	
DDE	
MAPE-IsoLineDD	
MAPE-Iso	
ps-uniform	
ps-glorot	

Table 4: Robot trajectories and policy collections for Bipedal-Walker mix-scale experiments.

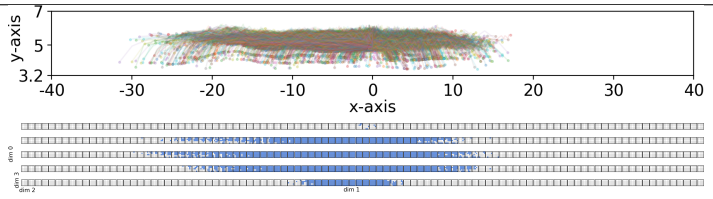
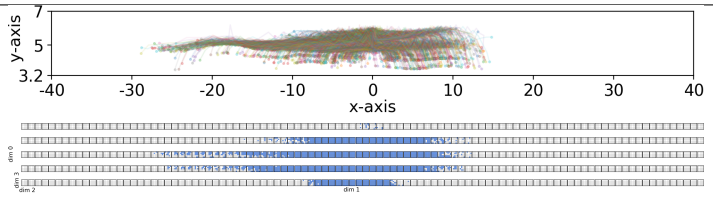
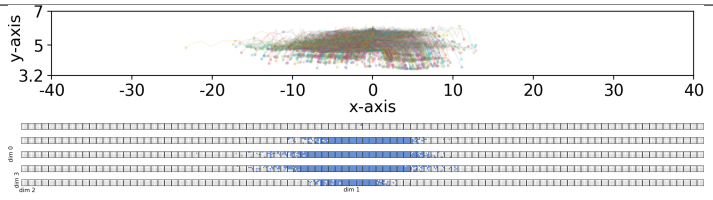
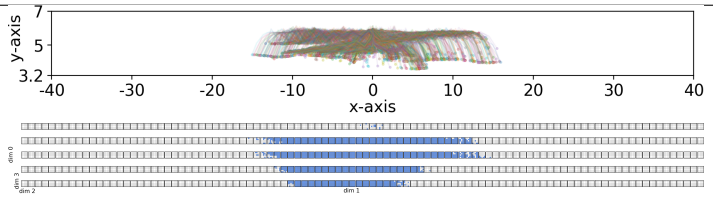
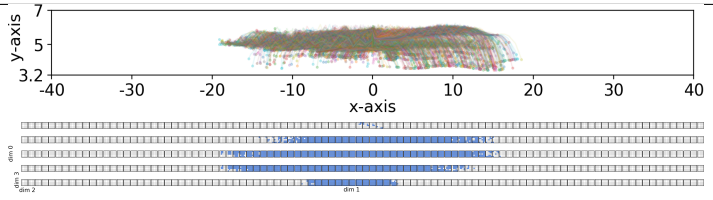
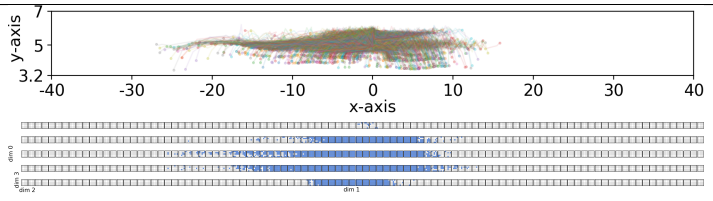
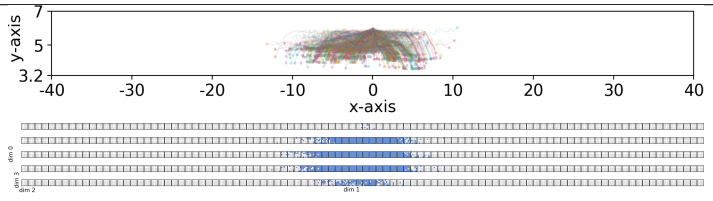
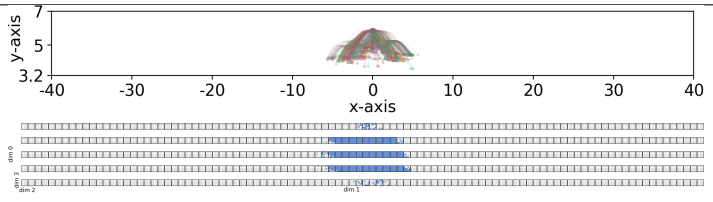
Algorithm	Bipedal-Walker mix-scale
PoMS	
PoMS-PCA	
PoMS-no-jacobian	
DDE	
MAPE-IsoLineDD	
MAPE-Iso	
ps-uniform	
ps-glorot	

Table 5: Ball trajectories and policy collections for Bipedal-Kicker experiments.

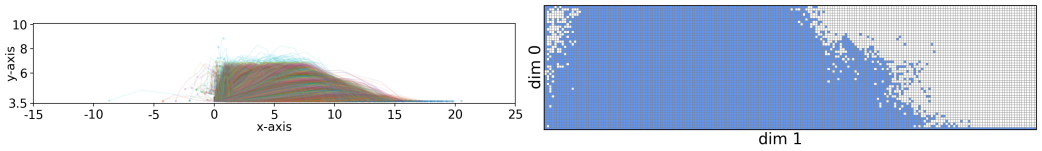
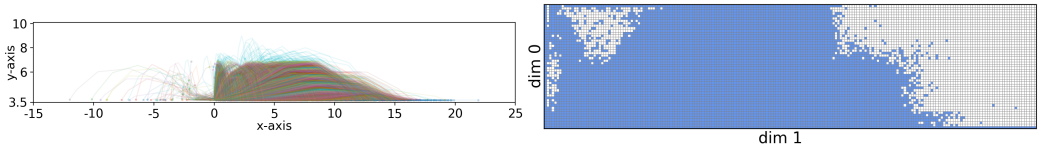
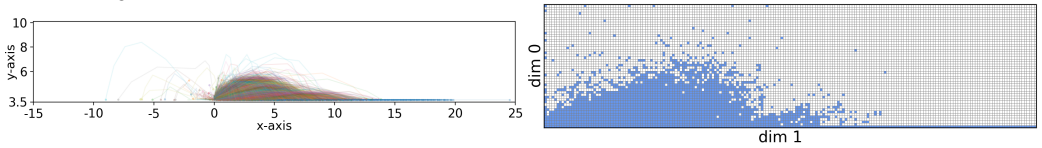
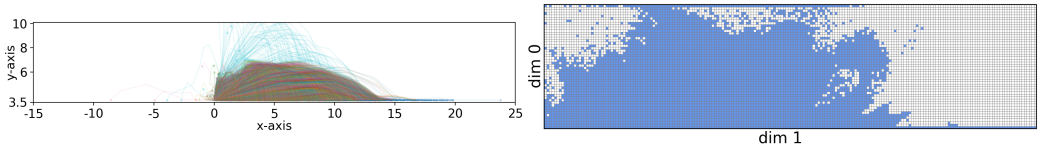
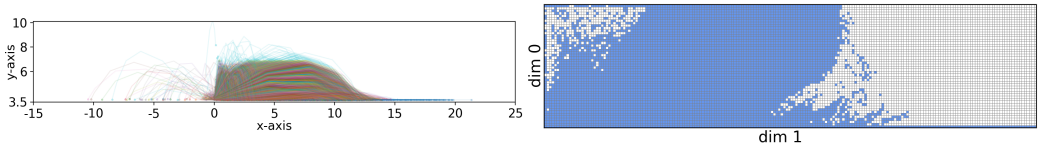
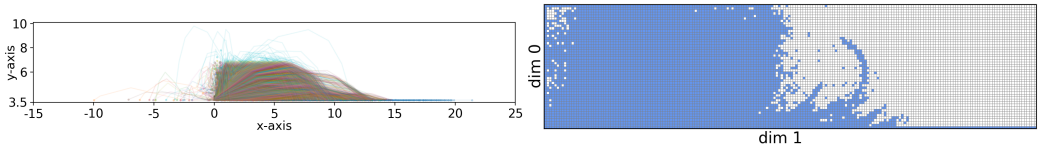
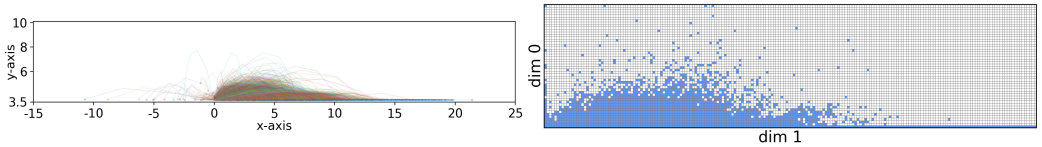
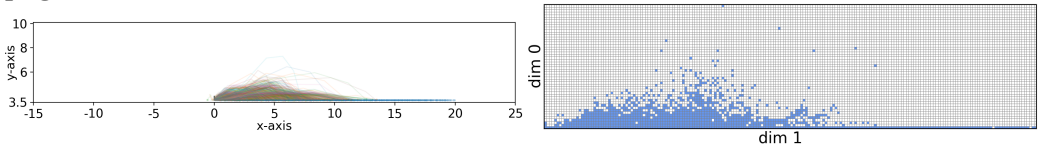
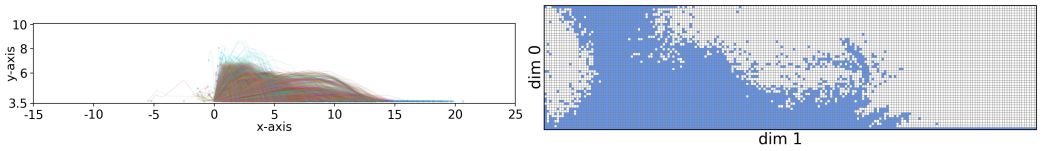
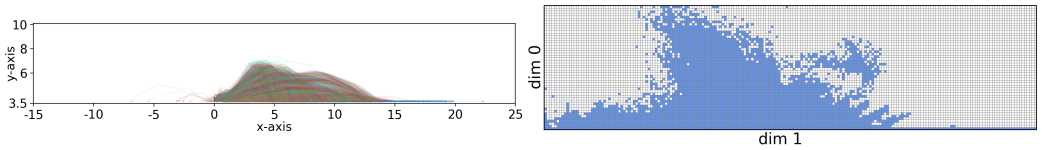
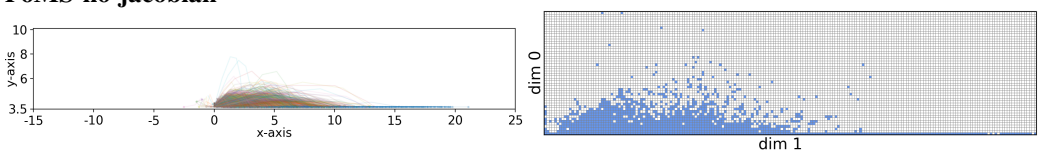
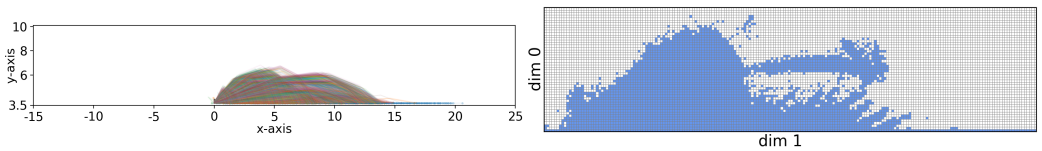
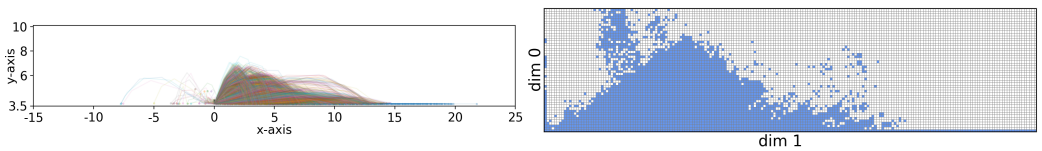
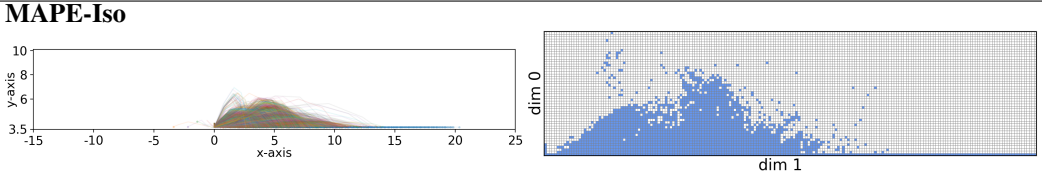
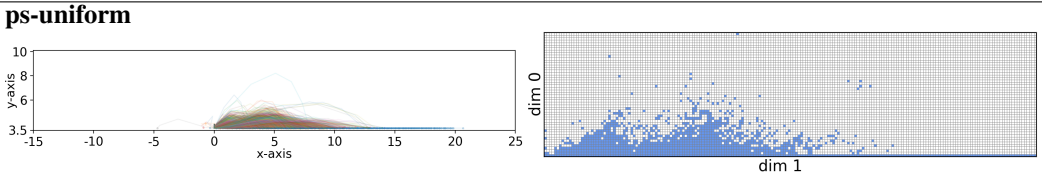
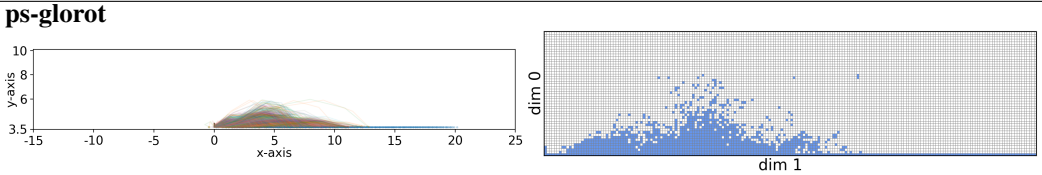
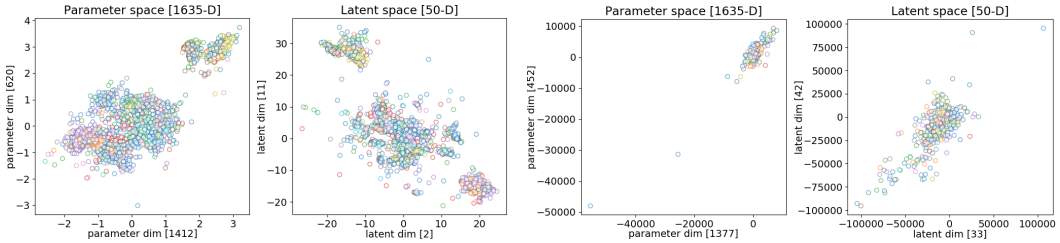
Algorithm	Bipedal-Kicker
PoMS	
PoMS-PCA	
PoMS-no-jacobian	
DDE	
MAPE-IsoLineDD	
MAPE-Iso	
ps-uniform	
ps-glorot	

Table 6: Ball trajectories and policy collections for Bipedal-Kicker mix-scale experiments.

Algorithm	Bipedal-Kicker mix-scale
PoMS	
PoMS-PCA	
PoMS-no-jacobian	
DDE	
MAPE-IsoLineDD	
MAPE-Iso	
ps-uniform	
ps-glorot	

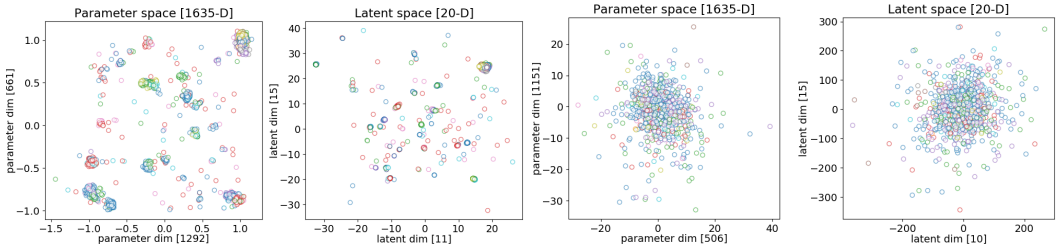
F.1 Parameter distributions and learned representations

We examine the effect that the Jacobian scaling has on the search process, and by extension the final policy collection. By visualising the parameter and latent spaces along dimensions of highest variance, we can see how PoMS-no-jacobian usually diverges to extremely large parameter values (Figures 5-10). This is due to the additional decoder transformation which is not accounted for. The colouring of the points is the same as in the trajectory plots from the previous section.



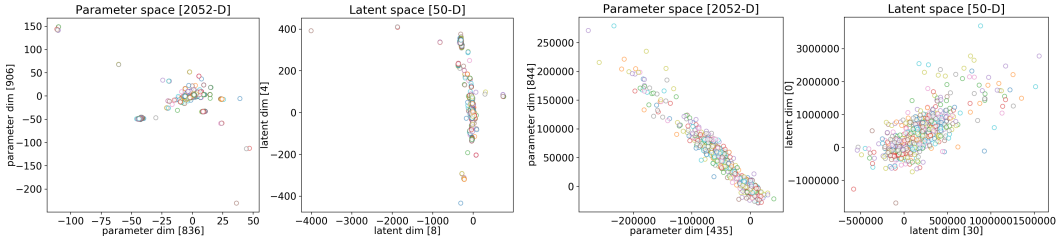
(a) PoMS (b) PoMS-no-jacobian

Figure 5: Comparison of parameter and latent spaces for Striker experiment.



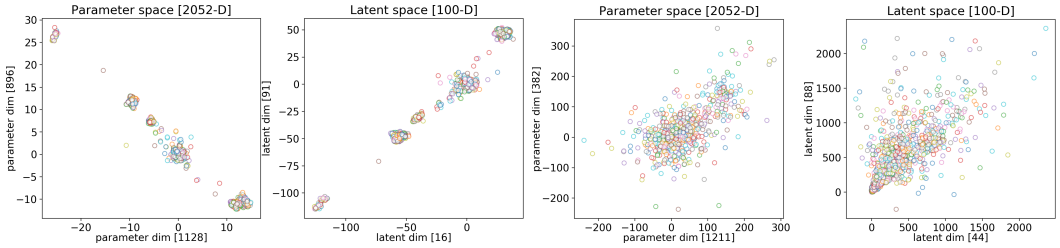
(a) PoMS (b) PoMS-no-jacobian

Figure 6: Comparison of parameter and latent spaces for Striker mix-scale experiment.



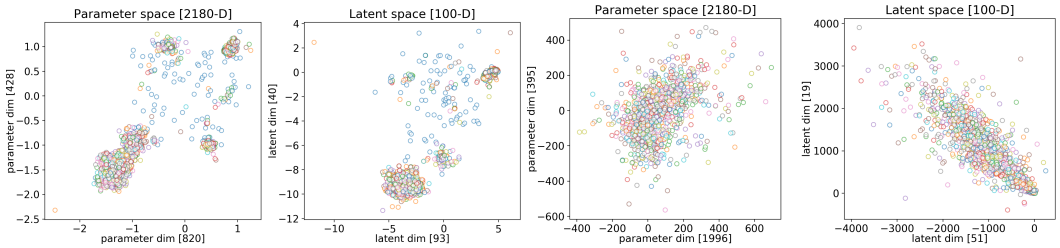
(a) PoMS (b) PoMS-no-jacobian

Figure 7: Comparison of parameter and latent spaces for Bipedal-Walker experiment.



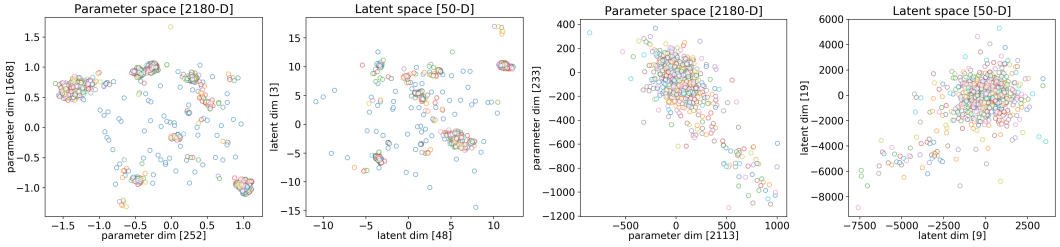
(a) PoMS (b) PoMS-no-jacobian

Figure 8: Comparison of parameter and latent spaces for Bipedal-Walker mix-scale experiment.



(a) PoMS (b) PoMS-no-jacobian

Figure 9: Comparison of parameter and latent spaces for Bipedal-Kicker experiment.



(a) PoMS (b) PoMS-no-jacobian

Figure 10: Comparison of parameter and latent spaces for Bipedal-Kicker mix-scale experiment.

References

[1] Joshua Achiam et al. "Variational option discovery algorithms". In: *arXiv preprint arXiv:1807.10299* (2018).

[2] Adrià Puigdomènech Badia et al. "Never Give Up: Learning Directed Exploration Strategies". In: *International Conference on Learning Representations*. 2020.

[3] Marc Bellemare et al. "Unifying count-based exploration and intrinsic motivation". In: *Advances in neural information processing systems*. 2016.

[4] Yoshua Bengio, Aaron Courville, and Pascal Vincent. "Representation learning: A review and new perspectives". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2013).

[5] Greg Brockman et al. "Openai gym". In: *arXiv preprint arXiv:1606.01540* (2016).

[6] Erin Catto. "Box2d: A 2d physics engine for games". In: *URL: http://www.box2d.org* (2011).

[7] Oscar Chang et al. "Agent Embeddings: A Latent Representation for Pole-Balancing Networks". In: *arXiv preprint arXiv:1811.04516* (2018).

[8] Nutan Chen et al. "Active learning based on data uncertainty and model sensitivity". In: *International Conference on Intelligent Robots and Systems*. 2018.

[9] Cédric Colas et al. "Scaling MAP-Elites to Deep Neuroevolution". In: *Genetic and Evolutionary Computation Conference*. Ed. by Carlos Artemio Coello Coello. 2020.

[10] Antoine Cully. "Autonomous skill discovery with quality-diversity and unsupervised descriptors". In: *Genetic and Evolutionary Computation Conference*. 2019.

[11] Antoine Cully and Yiannis Demiris. "Quality and diversity optimization: A unifying modular framework". In: *IEEE Transactions on Evolutionary Computation* (2017).

[12] Antoine Cully et al. "Robots that can adapt like animals". In: *Nature* (2015).

[13] Benjamin Eysenbach et al. "Diversity is All You Need: Learning Skills without a Reward Function". In: *International Conference on Learning Representations*. 2019.

[14] Manon Flageat and Antoine Cully. "Fast and stable MAP-Elites in noisy domains using deep grids". In: *Artificial Life Conference Proceedings*. MIT Press. 2020, pp. 273–282.

[15] Meire Fortunato et al. "Noisy Networks For Exploration". In: *International Conference on Learning Representations*. 2018.

[16] Adam Gaier, Alexander Asteroth, and Jean-Baptiste Mouret. "Discovering Representations for Black-box Optimization". In: *Genetic and Evolutionary Computation Conference*. 2020.

[17] Daniele Gravina et al. “Procedural content generation through quality diversity”. In: *arXiv preprint arXiv:1907.04053* (2019).

[18] Karol Gregor, Danilo Jimenez Rezende, and Daan Wierstra. “Variational Intrinsic Control”. In: *International Conference on Learning Representations*. 2017.

[19] Tuomas Haarnoja et al. “Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor”. In: *International Conference on Machine Learning*. 2018.

[20] Karol Hausman et al. “Learning an Embedding Space for Transferable Robot Skills”. In: *International Conference on Learning Representations*. 2018.

[21] Marija Jegorova, Stéphane Doncieux, and Timothy Hospedales. “Generative Adversarial Policy Networks for Behavioural Repertoire”. In: *arXiv preprint arXiv:1811.02945* (2018).

[22] Niels Justesen, Sebastian Risi, and Jean-Baptiste Mouret. “MAP-Elites for noisy domains by adaptive sampling”. In: *Proceedings of the Genetic and Evolutionary Computation Conference Companion*. 2019, pp. 121–122.

[23] Joel Lehman and Kenneth O Stanley. “Evolving a diversity of virtual creatures through novelty search and local competition”. In: *Genetic and Evolutionary Computation Conference*. 2011.

[24] Chunyuan Li et al. “Measuring the intrinsic dimension of objective landscapes”. In: *arXiv preprint arXiv:1804.08838* (2018).

[25] Elliot Meyerson, Joel Lehman, and Risto Miikkulainen. “Learning behavior characterizations for novelty search”. In: *Genetic and Evolutionary Computation Conference*. 2016.

[26] Jean-Baptiste Mouret and Jeff Clune. “Illuminating search spaces by mapping elites”. In: *arXiv preprint arXiv:1504.04909* (2015).

[27] Alexandre Péré et al. “Unsupervised Learning of Goal Spaces for Intrinsically Motivated Goal Exploration”. In: *arXiv preprint arXiv:1803.00781* (2018).

[28] Matthias Plappert et al. “Parameter Space Noise for Exploration”. In: *International Conference on Learning Representations*. 2018.

[29] Justin K Pugh, Lisa B Soros, and Kenneth O Stanley. “Quality diversity: A new frontier for evolutionary computation”. In: *Frontiers in Robotics and AI* (2016).

[30] Pierre H Richemond, Arinbjörn Kolbeinsson, and Yike Guo. “Biologically inspired architectures for sample-efficient deep reinforcement learning”. In: *Deep Reinforcement Learning Workshop, NeurIPS* (2019).

[31] Salah Rifai et al. “A generative process for sampling contractive auto-encoders”. In: *International Conference on Machine Learning*. 2012.

[32] Salah Rifai et al. “Contractive Auto-Encoders: Explicit Invariance During Feature Extraction”. In: *International Conference on Machine Learning*. 2011.

[33] Andrei A. Rusu et al. “Meta-Learning with Latent Embedding Optimization”. In: *International Conference on Learning Representations*. 2019.

[34] John Schulman et al. “Proximal policy optimization algorithms”. In: *arXiv preprint arXiv:1707.06347* (2017).

[35] Archit Sharma et al. “Dynamics-aware unsupervised discovery of skills”. In: *arXiv preprint arXiv:1907.01657* (2019).

[36] Haoran Tang et al. “# exploration: A study of count-based exploration for deep reinforcement learning”. In: *Advances in neural information processing systems*. 2017.

[37] Neil Urquhart, Silke Höhl, and Emma Hart. “An illumination algorithm approach to solving the micro-depot routing problem”. In: *Genetic and Evolutionary Computation Conference*. 2019.

[38] Vassilis Vassiliades and Jean-Baptiste Mouret. “Discovering the elite hypervolume by leveraging interspecies correlation”. In: *Genetic and Evolutionary Computation Conference*. 2018.

[39] Brian D Ziebart et al. “Maximum entropy inverse reinforcement learning.” In: *Conference on Artificial Intelligence*. 2008.