On the importance of data collection for training general goal-reaching policies

Alexis Jacq
Google DeepMind

Manu Orsini
Google DeepMind

Gabriel Dulca-Arnold
Google DeepMind

Olivier Pietquin
Google DeepMind

Matthieu Geist
Google DeepMind

Olivier Bachem
Google DeepMind

Abstract

Recent advances in ML suggest that the quantity of data available to a model is one of the primary bottlenecks to high performance. Although for language-based tasks there exist almost unlimited amounts of reasonably coherent data to train from, this is generally not the case for Reinforcement Learning, especially when dealing with a novel environment. In effect, even a relatively trivial continuous environment has an almost limitless number of states, but simply sampling random states and actions will likely not provide transitions that are interesting or useful for any potential downstream task. *How should one generate massive amounts of useful data given only an MDP with no indication of downstream tasks? Are the quantity and quality of data truly transformative to the performance of a general controller?* We propose to answer both of these questions. First, we introduce a principled unsupervised exploration method, ChronoGEM, which aims to achieve uniform coverage over the manifold of achievable states, which we believe is the most reasonable goal given no prior task information. Secondly, we investigate the effects of both data quantity and data quality on the training of a downstream goal-achievement policy, and show that both large quantities and high-quality of data are essential to train a general controller: a high-precision pose-achievement policy capable of attaining a large number of poses over numerous continuous control embodiments including humanoid.

1 Introduction

Recent work in large language models (Chowdhery et al., 2022), as well as general conclusions such as though proposed by (Sutton, 2019) suggest that one of the main bottlenecks of current machine-learning methods is access to large amounts of informative data. This is particularly true for the domain of Reinforcement Learning (RL) (Sutton and Barto, 2018), where data acquisition can be costly, and coherent datasets (Gulcehre et al., 2020; Fu et al., 2020) are less prominent and smaller than text datasets pulled from the internet. This brings us to the following two core questions:
•*How should one generate massive amounts of useful data given only an MDP with no indication of downstream tasks?*
•*What are the effects of quantity and quality of data on the training of a general policy?*

To answer the first question, we present Chronological Greedy Entropy Maximisation (ChronoGEM), a principled, highly scalable exploration method whose goal is to achieve uniform coverage over the manifold of achievable states. We demonstrate both theoretically that ChronoGEM approximates uniform sampling, and empirically that states achieved by ChronoGEM are diverse and well-distributed.

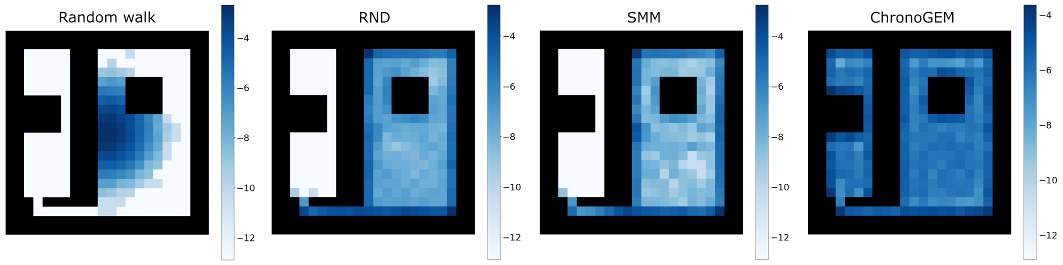


Figure 1: Log-frequencies of the discretized states visitation when taking the last states from 4000 episodes, sampled according to (left to right) a random walk, SMM, RND and ChronoGEM, averaged over 10 seeds. Only ChronoGEM managed to visit states in the top-left room.

Regarding the second question, we investigate the effects of data quality and quantity using Entropy-Based Conditioned Continuous Control Policy Optimization (C3PO), a goal-conditioned policy training pipeline that leverages pre-defined datasets of uniform goal states to learn a general pose-attainment policy on various high-DoF control tasks. Data quality is analyzed by comparing performance of our policy trained with goals sampled from ChronoGEM vs other contemporary unsupervised exploration methods. The question of quantity is particularly interesting if we can analyze the asymptotic effects of data, perhaps far beyond what is generally realized in RL training. The main blocker to this type of experiment in the past has simply been the time such experiments would take to run, but given observations that the long-tail of training can still provide significant performance benefits (Silver et al., 2017), it is important to understand how much is being left off the table by not pushing this limit. Recent advances in accelerator-specific parallelizable physics simulators such as Brax (Freeman et al., 2021) can allow us to train our policies into the billions of steps in a matter of hours, thus allowing us both to tune said approaches and observe their asymptotic behavior when pushing training orders of magnitude beyond what is usually performed.

We structure our paper by first introducing our two methods in Section 2: We look at both the theoretical foundations of ChronoGEM, as well as its independent building blocks. We also introduce the C3PO training setup, and discuss how it can be used to evaluate the effects of quantity and quality of data for learning a generalist policy. In Section 3 we perform a series of experiments to attempt to answer our core questions. We begin by comparing the performance of ChronoGEM relative to existing unsupervised exploration methods such as RND Burda et al. (2018) and SMM (Lee et al., 2019) on a series of environments, ranging from an illustrative 2-D maze task, to high-DoF continuous control environments through various methods. We then investigate the effects of quantity and quality of data on the training of a general controller policy. We do this by comparing goal achievement across our different goal datasets, generated by either ChronoGEM, RND or SMM, and show that policies learnt using ChronoGEM data can achieve a much more diverse set of goals than those learnt on other data sources. Finally, we push the training regime to multiple billions of steps to examine the asymptotic performance of our policy architecture, and achieve high-precision pose-attainment even on a complex system such as Humanoid. Finally, we briefly demonstrate applicability of a general controller in a zero-shot imitation task. Section 4 situates our proposed methods relative to current research.

2 Methods: ChronoGEM & C3PO

We introduce our two contributions: Chronological Greedy Entropy Maximisation (ChronoGEM) & Entropy-Based Conditioned Continuous Control Policy Optimization (C3PO). ChronoGEM is our proposed solution for principle scalable unsupervised exploration of high-DoF environments which we use to investigate how to generate massive amounts of useful data for downstream RL tasks. C3PO is our proposed pose-attainement policy training method to investigate the effects of quantity and quality of data on the performance of a generalist policy. To clarify notations we will quickly introduce the Markov Decision Process (MDP), which can be defined by a transition function T which maps states $s \in \mathcal{S}$ and actions $a \in \mathcal{A}$ to a corresponding state $s_{t+1} = T(s_t, a_t)$, while potentially providing a scalar reward $r(s_t, a_t) \in \mathbb{R}$ if there is an associated task.

2.1 Chronological Greedy Entropy Maximisation (ChronoGEM)

Given an arbitrary Markov Decision Process (MDP), and the goal of generating massive amounts of useful data for arbitrary downstream tasks, and no prior to guide our exploration of the MDP, we argue that the ideal set of explored states should be uniformly sampled from the manifold of reachable states at a given horizon T. The shape of the manifold of reachable states is cannot be known *a priori*, and may be arbitrarily complex, thus rendering any form of direct sampling impossible. Given acces to the MDP, this sampling can however be approximated by an iterative algorithm.

Let us assume we have a sample of N states that are approximately uniform for a horizon of $T-1$. We can perform K uniform actions from each of these states, and result with NK states, whose distribution will be biased by the MDPs natural dynamics instead of being uniform across state values. Nevertheless, with a sufficiently large N there is likely some subset of these NK states which would can be sampled to approximate a uniform distribution over states at horizon T. Let ρ_T be the distribution induced by these NK next states, and let us assume we have a method to estimate ρ_T. Since the set of achievable states in T steps is generally bounded, and given that we are able to closely estimate ρ_T, we can sub-sample the NK states with probability $\frac{1}{\rho_T}$, which will approximate uniform sampling according to the state statistics maintained in ρ_T. We prove in Appendix A that such sub-sampling approximates uniform sampling when the number of sampled states is sufficiently large.

Given this recursive definition, we can begin with states sampled from the environment's initial starting-state distribution ρ_0, perform K uniform actions and generate NK states. We can then sub-sample the NK states back to N states that approximate the uniform distribution using inverse probability weighting according to our density estimator ρ. By iterating this process T times, we can obtain a set of N states that are uniform for a given horizon T. We call this process ChronoGEM (for Chronological Greedy Entropy Maximization) since at a given step, it only focuses on maximizing the entropy by directly approximating a uniform distribution over the next step, without further planning. ChronoGEM is described more formally in Algorithm 1.

Algorithm 1 ChronoGEM

```
1: Sample $N$ states $S_0 = \{s_0^i\}_{i=1}^N \sim \rho_0$.
2: for $t = 1$ to $T$ do
3:     Sample $K$ uniform actions for each state of $S_{t-1}$.
4:     Obtain $KN$ next states.
5:     Estimate $\rho_t$ using a density model fitted on the distribution of these $KN$ states.
6:     Sample $N$ states with $p(s) \propto \frac{1}{\rho_t(s)}$ to get $S_t$.
7: end for
8: Return $S_T$.
```

ChronoGEM requires exactly KNT interactions with the environment, which makes it easily controllable in term of sample complexity. Due to the N sampled states being independent, ChronoGEM can be parallelized with N jobs consuming KT interactions each, significantly shortening the time complexity. As implemented, ChronoGEM requires re-settable states, although this could potentially be relaxed with a return-to-state policies similar to 'First Return then Explore' (Ecoffet et al., 2021), refer to Appendix C.0.1 for more discussion.

Density estimation. As we saw above, ChronoGEM requires a density estimator at each iteration to estimate ρ_t. Many choices of density estimation in high dimensional space exist, from simple Gaussian estimators to neural network-based methods such as autoregressive models or normalizing flows. The performance of these models may vary given the type of data: Some models are more suited for images, while others are better for text or lower dimensions. We implemented 7 candidate models, including Gaussian models (Multivariate, Mixture), autoregressive networks (RNade (Uria et al., 2013), Made (Germain et al., 2015)), and normalizing flows (real-NVP (Dinh et al., 2016), Maf (Papamakarios et al., 2017), NSF (Durkan et al., 2019)). After comparing performance of the various models through a state modeling task decribed in Appendix B, we concluded that the NSF variant of normalizing flows worked best for the state modeling task.

2.2 Entropy-Based Conditioned Continuous Control Policy Optimization (C3PO)

C3PO's objective is to learn a generalist control policy that can attain any reachable state in a given environment. Our intent is to investigate the effects of data quantity and quality on C3PO's training regime, and better understand the importance of both diverse data, and the asymptotic effects of massive amounts of experience on policy quality.

C3PO is a training regime that wraps an arbitrary policy learning algorithm with a curriculum and success threshold annealement. It requires a dataset of goal states $s_g \in \mathcal{G}$, as well as a goal-achievement criteria $S : (\mathcal{S}, \mathcal{S}, \mathbb{R}) \to \{0, 1\}$. The overall training regime is defined in Algorithm 2. In practice this training loop is run in a highly-parallelized manner on a set of accelerators, allowing for significantly longer episodes than is usually performed in RL experiments. The success criteria's threshold is initialized to a relatively tolerant value, and conditionally annealed every time the learnt policy achieves 90% success rate according to the goal-achievement criteria.

As we will see in Section 3.3, by using various sources of goal states, such as ChronoGEM, RND, SMM, and random walk for which we will have already compared the state space coverage and relative entropy, we can understand the downstream effects of data quality on training a general policy. With regards to data quantity, we will see in Section 3.3 that by pushing the number of training steps into the billions, we can significantly improve goal achievement rates of the generalist policy.

Algorithm 2 C3PO

```
Require: Goals $\mathcal{G}$, Achievement Criteria $S$, Threshold $\epsilon$
 1: success_rate $= 0$, $\mathcal{D} = \emptyset$
 2: while True do
 3:     Draw goal $s_g \in \mathcal{G}$
 4:     Rollout $\pi(\cdot, s_g)$, until $S(s_t, s_g, \epsilon) == 1$ or $t == T$
 5:     Add rollout to $\mathcal{D}$
 6:     Update success_rate average with $S(s_t, s_g, \epsilon)$
 7:     IF (success_rate $> 0.9$) : $\epsilon = 0.99 \times \epsilon$
 8:     Run policy training algorithm on $\mathcal{D}$.
 9: end while
10: return $\pi$
```

3 Experiments

We will now detail the experiments performed to investigate our two core questions. First we will look at experiments on *how one should generate massive amounts of useful data*, by investigating the performance of ChronoGEM relative to other unsupervised exploration algorithms. We will then look at the effects of *data quantity and quality* by looking at C3PO performance as these two factors are varied.

3.1 Environments

2D Maze. As a tool scenario to test ChronoGEM, we implemented a two-dimensional continuous maze in which actions are d_x, d_y steps bounded by the $[-1, 1]^2$ square, and the whole state space is bounded by the $[-100, 100]^2$ square. This environment, illustrated in Fig? 1 is similar to the maze used by Kamienny et al. (2021), except it adds a significant difficulty induced by the presence of two narrow corridors that needs to be crossed in order to reach the top-left room.

Continuous control tasks. We use control tasks from Brax (Freeman et al., 2021) as high dimensional environments. We chose four environments to cover varying complexities of task: Hopper, Walker2d, Halfcheetah and Ant. Environment observations were modified to contain (x, y, z) positions of all body parts to be better aligned with pose-achievement goals. All measures (cross-entropy in Section 3.2 and reaching distances in Section 3.3) are based on this observation space. To attenuate energy accumulation corner cases in the simulator, we maintain the environments in a low energy regime by reducing the action amplitude by a factor of 0.1 for Hopper and Walker, 0.01 for HalfCheetah. Actions for Ant are unmodified. In the two following subsections, we considered episodes of length $T = 128$. To compensate for varying control frequencies while keeping the same

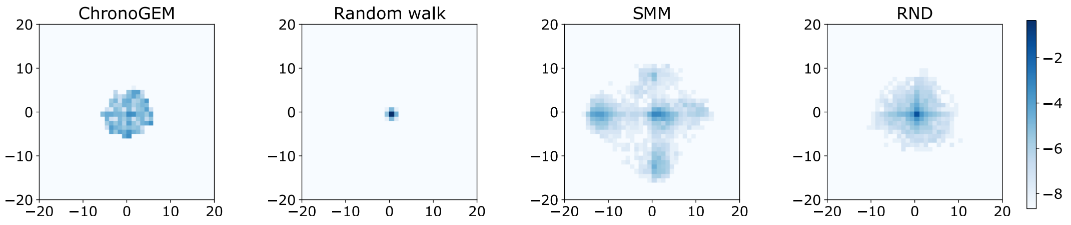


Figure 2: Sky-view of the discretised spatial log-frequency covered by Ant with the different exploration methods. SMM has the largest scope but contains empty zones even close to the origin. Both RND and ChronoGEM share similar range covering all directions, and ChronoGEM is visibly much more uniform, while other methods are concentrated on the origin. Note that this only represents the spatial positions, while both poses and positions are being explored.

wall clock episode length, there is an action repeat of 6 for Hopper and Walker. All default episode end conditions (primarily involving falling) have been removed.

3.2 Generating Useful Data With ChronoGEM

In our first set of experiments, we look at the performance of ChronoGEM relative to three other unsupervised exploration algorithm: RND (Burda et al., 2018), SMM (Lee et al., 2019) and a random walk policy. In Section 3.2 We will begin by looking at an easily interpretable environment, a 2D maze. We will then look at more complex environments and quantify data coverage in Section 3.2. ChronoGEM was run with $N = 2^{17}$ parallel environments and branching factor $K = 4$ in all experiments, except for Humanoid where $N = 2^{15}$ and $K = 64$ (Humanoid required a larger branching factor to avoid absorbing states). ChronoGEM-related hyperparameters are in Table 3 (Appx.).

Chronogem Visualization: 2D Maze. The main goal of this experiment is to verify that ChronoGEM manages to induce a uniform distribution over the whole state space of a simple yet challenging toy environment. In order to emphasize the relative difficulty of the exploration of this maze, we also run SMM, RND and a random walk to compare the resulting state coverage. In this setup, we know that if T is large enough, all achievable states are just every point in the maze, so ChronoGEM should be uniform on the maze given sufficient time (for instance, $T = 1000$). We can see the final state distribution in Figure 1, and observe that while ChronoGEM achieves fairly uniform state coverage, both RND and SMM fail at exploring beyond the first corridor, and a random walk did not even explore the whole first room. This suggests that ChronoGEM is performing as expected, and that RND and SMM struggle with bottleneck states, whereas random walk has trouble with diffusing far from the starting state distribution.

State Coverage Quantification with Entropy. In this section we will look at a more quantified method for evaluating state coverage of an exploration method using our learnt entropy estimator. Given a set of points $x_1 \dots x_N$ sampled from a distribution with an unknown density ρ, one can estimate an upper-bound of the entropy of ρ given by the cross-entropy $H(\rho, \hat{\rho})$ where $\hat{\rho}$ is an estimation of ρ: $H(\rho, \hat{\rho}) = -\mathbb{E}_{x\sim\rho}[\log \hat{\rho}(x)] = H(\rho) + \mathrm{KL}(\rho||\hat{\rho}) \geq H(\rho)$. The estimate $\hat{\rho}$ being learned by maximum likelihood specifically on the set of points, it directly minimises the cross entropy and closely approximates the true entropy. The KL term becomes negligible and only depends on the accuracy of the trained model on the observed set of points, which supposedly does not differ given the different exploration method that generated the points. Consequently, comparing the cross-entropy is similar to comparing the entropy of the distribution induced by the exploration. In this experiment, we used this upper-bound to study the efficiency of ChronoGEM compared to RND, SMM and a random walks. Figure 3 displays box plots over 10 seeds of the resulting cross entropy measured on the sets of states induced by different algorithms, on the 4 continuous control tasks. As expected, the random walk has the lowest entropy, and SMM has a better entropy than RND on Hopper and Walker2d (which makes sense since it is optimizing for the maximization of the entropy). ChronoGEM has the highest entropy on all environments, especially on HalfCheetah, where it was the only method to manage exploration while the actions were drastically reduced by the low multiplier (see Section 3.1). In order to illustrate the fact that ChronoGEM induces a state distribution

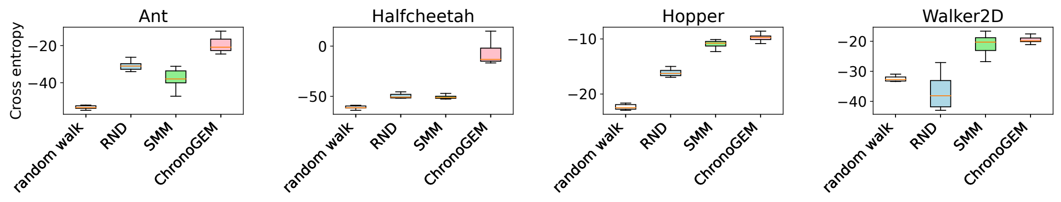


Figure 3: Distribution over 10 seeds of the cross entropies of the state visitation induced by ChronoGEM, RND, SMM and a random walk, on different continuous control tasks.

that is close to uniform, we measure the spatial coverage based on a discrete grid of the x-y plan: if the distribution is uniform over both the possible poses and positions, it should be in particular uniform over the positions. Figure 2 shows the resulting log-frequency on the x-y grid visitations and although ChronoGEM does not have the biggest exploration horizon, it nevertheless provides the most uniform coverage. Grid visitation for Hopper, Walker2d and Halfcheetah are available in Figure 8 in Appendix C. Given both the entropy coverage and the various qualitative visualizations, we believe ChronoGEM provides consistently uniform and exhaustive coverage of the state space for a given horizon T. Additionally, it is an efficient algorithm, which can quickly generate large amounts of data. For these reasons we believe it is a good candidate for generating massive amounts of useful data for downstream tasks. We will now look at this in practice by comparing its utility relative to RND, SMM and randomw-walk datasets as goal states for training a general goal-achievement policy.

3.3 Effects of Data Quality and Quantity on General Policy Training with C3PO

In this section we investigate the effects of data quality and quantity on the training of a general policy using C3PO. We start by looking at data quality, by evaluating C3PO policies trained on varied datasets of goal states. We hope to better understand the importance of data coverage for the performance of training a general goal-attainment policy. Indeed, the ideal goal distribution would be uniform across all achievable states, so if the data was generated in a way that it best approximates this distribution, we would hope this would significantly impact the quality of our trained policy. For each environment, we run the four exploration methods (ChronoGEM, Random Walk, SMM and RND) with 3 seeds each, which we split into training & evaluation goal sets. Training goal sets have 4096 goals and evaluation goal sets have 128 goals. For the task of goal-achievement, we use the following reward: $r(s_t, g) = -\|s_t - g\|_\infty^2 = -\max_{\mathbf{b_i} \in \mathbf{s_t}} \|\mathbf{b_i} - \mathbf{g_i}\|^2$, where $\mathbf{b_i}$ is a set of coordinates corresponding to a body component of the embodiment (leg, torso, foot). This reward effectively penalizes according to the distance of the most distant body part relative to the desired goal pose. The success criteria is $S(s_t, g, \epsilon) = |r(s_t, g)| < \epsilon$, thus effectively ending the episode once a certain value of reward is attained. The underlying training algorithm used is SAC (Haarnoja et al., 2018) with the hypers detailed in Table 4. Goal-states are simply appended to the observation vector of the policy network.

Effects of Data Quality. We evaluate policy performance with two quantitative and one qualitative approach. We perform cross-validation across datasets, by evaluating each policy trained with one dataset on evaluation goals from all other datasets. We observe that C3PO trained on ChronoGEM data is the most robust across datasets and environments, matching or often beating policies evaluated on their own datasets, especially for low tolerance values of ϵ. This can be explained by the fact that C3PO learns to reach a high variety of poses, since being able to achieve poses with high fidelity is what matters for low distance threshold regime. We believe that slightly lower performance on some environment/dataset pairs can be explained by goals being generally closer to the origin with ChronoGEM than SMM or RND (c.f. Figure 2). Full results of the cross-validation with regards to varying ϵ are visualized in the Appendix, Figure 9.

We can better quantify the global results by collecting all the areas under the curve (AUC), and weighting them proportionally to the exponential of the evaluation goals' entropy. In effect, if a goal-set is very diverse, goals therein are more diverse and therefore more interesting to achieve. Conversely, if a goal-set has low entropy, it may simply contain the same couple of goals that are hard to achieve, but have low value in terms of learning a generalist controller. The exponential of the entropy quantifies the number of states in the distribution. We call this metric Entropy Weighted Goal Achievement (EWGA): $\sum_{s \in \text{eval sets}} e^{\text{entropy}(s)} * \text{AUC}(\text{method on } s) / \sum_{s \in \text{eval sets}} e^{\text{entropy}(s)}$.

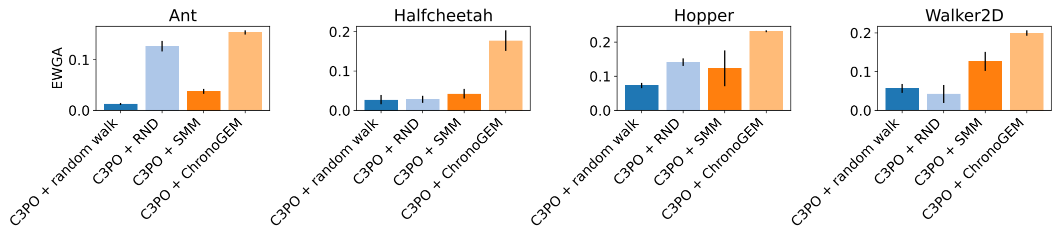


Figure 4: Entropy Weighted Goal-Achievement (EWGA). This estimates the ability of a policy to achieve goal sets that better covers the space (for example, a policy like C3PO that reaches a larger variety of states has an higher EWGA than a policy like SAC trained on the Random Walk, that only reaches states that are close to the origin).

The performance of each C3PO variant is detailed in Figure 4, where we can see that C3PO trained on ChronoGEM data is significantly better across all four environments regarding entropy-weighted AUC. To answer our original question on the importance of data quality, we can see that on datasets with higher entropy as per Figure 3, downstream C3PO policy performance is higher both in cross-validation (Figure 9) and in entropy-weighted AUC (Figure 4). In particular, ChronoGEM-trained C3PO is generally the most robust on un-normalized cross-validaiton, and is significantly superior on entropy-reweighted AUC.

Effects of Data Quantity. We now want to investigate the importance of large amounts of experiential data on the performance of a general goal-achievement policy. To do this, we will let C3PO train for many orders of magnitude more than what is generally used during continuous-control training regimes. For questions of resource-efficiency, we restrict these experiments to the ChronoGEM datasets, as they seem to perform best for goal-achievement. For environments where we do not converge to a good controller earlier, we let them run for up to 30×10^12 steps. Thanks to Brax's high parallelization and efficient infrastructure, it is possible to run such an experiment in a couple of days. Although we did not use it in our earlier experiments for resource reasons, we also add the much more difficult Humanoid task to our set environments for this analysis, with a couple modifications described in Appendix E.1. In Figure XX we show the performance of each policy for a fixed ϵ. We can see that for certain environments such as Hopper and Walker2d train in tens of millions of steps. One could be tempted to conclude that halfcheetah and ant have also hit their maximum performance at this stage. However, we show that by continuing training for billions of steps, we can also achieve very good performance on these environments. We illustrate some of the goal-achivement abilities of the policies able to achieve 90% success at .25 tolerance are represented in Figure 6 and in supplementary material[1]. We believe this shows that not only is data quality important, but also providing significant amounts of experience is necessary for general goal-achievemnet policies to attain high levels of performance.

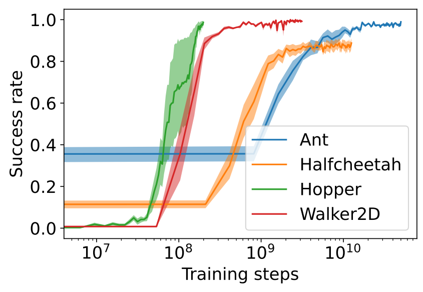


Figure 5: This plots illustrates C3PO-ChronoGEM training curves for each of the four environments used. We can observe that although training can converge quickly on some environments, other require significant amounts of interaction. Final convergence on Ant for example takes upwards of 10^{10} steps.

3.4 Zero-shot imitation

As an example task to illustrate the utility of a general goal-achievement policy we look at zero-shot imitation. We can easily perform this by telling the C3PO policy to aim for successive target states from a pre-generated expert policy trajectory.

we used a pre-trained C3PO policy to imitate an expert demonstration. Additional details on the implementation are in Appendix E.2. We define two types of task, one where a policy is trained

[1]Please refer to https://sites.google.com/view/chronogemc3po/home

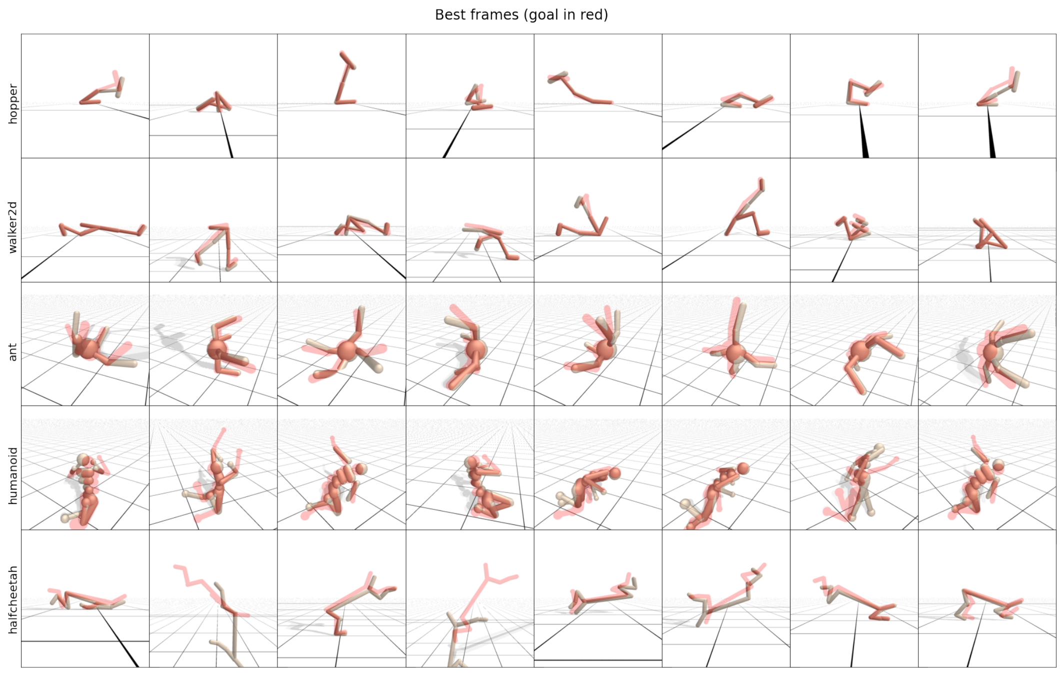


Figure 6: For each environment, we drew 8 goals from the ChronoGEM distribution and ran the policy trained on ChronoGEM goals. This figure represent the frame that is the closest from the goal for each episode, overlayed with the goal in red.

from scratch on the base environment reward, which involves moving forward at a certain speed, and another where we take trajectories induced by ChronoGEM exploration (ChronoGEM has the side effect of generating original and energetic behaviours like cartwheels or backflips). For imitation of the base environment reward policies, we can evaluate the effective reward of the imitation policy. Results suggest that zero-shot imitation based on C3PO reaching demonstrator's states can achieve around 50% of the expert score. For Ant, Halfcheetah, Hopper and Walker2d, the imitating agent successfully walked in the rewarding direction. Zero-shot imitation failed for humanoid, that could only manage to maintain a standing position but did not walk in the right direction. Full results are reported in in Table 1.

Imitation of ChronoGEM trajectories was easier in the sense that all the states where by construction in-domain for the policy. Because it is difficult to quantitatively measure this performance in an interpretable way, we joined the resulting videos (both demonstration and imitation) in the supplementary material[2]. These videos show that Hopper and Walker almost perfectly imitate the expert, while Ant and Halfcheetah only "approximate" the demonstration (eg, halfcheetah realizing one front flip while the expert did two).

Environment	Expert	Zero-shot imitation
Ant	2281.45	1083.48 ± 317.42
HalfCheetah	1092.65	984.32 ± 112.49
Hopper	676.58	214.95 ± 138.88
Walker2d	965.53	562.26 ± 51.01
Humanoid	2670.59	588.93 ± 505.45

Table 1: Zero-shot imitation of downstream tasks, based on a single expert demonstration of length 300. Imitation results are averaged over 9 different values (3 ChronoGEM seeds for C3PO $\times$ 3 environment seeds).

4 Related works

This work is situated between various fields. Although effectively a goal-conditioned policy optimization algorithm, C3PO is enabled by the ChronoGEM exploration algorithm. We will first look at similar exploration methods and then consider various goal-conditioned learning setups.

Bonus-based exploration. Although generally not concerned with goal-conditioned RL, there is a large family of exploration methods that are manifest as reward bonuses, with the intent of training a policy faster, or to be more efficient. One family of approaches uses state-visitation counts that can be

[2]Please refer to `https://sites.google.com/view/chronogemc3po/home`

approximate to create an associated bonus for rarely-visited states (Bellemare et al., 2016; Ostrovski et al., 2017). Prediction-error bonuses use the residual error on predictions of future states as a reward signal to approximate novel states, this includes methods such as RND (Burda et al., 2018) which leverages the prediction error of random projections of future states, or SPR (Schwarzer et al., 2020) and BYOL-Explore (Guo et al., 2022), which make use of the self-prediction error of the network with a frozen version of itself. Model-based methods often optimise for next-state novelty, either by looking at the KL divergence between sampled states and likely states, such as in VIME (Houthooft et al., 2016) or by explicitly driving towards states with high model ensemble disagreement such as in Plan2Explore (Sekar et al., 2020). RIDE (Raileanu and Rocktäschel, 2020) and NGU (Badia et al., 2020) use episodic memory in which the bonus reflects the variety of different states covered in a single trajectory.

Diffusion-based exploration ChronoGEM is based on a tree-structured diffusion, that makes a selection of states, randomly explores from these states and then reselect states, and so on. Go-Explore (Ecoffet et al., 2019) uses a similar approach, by running a random policy for some steps, selecting a set of 'interesting' states, and then branching from these. ChronoGEM does not require returning to said states, and only requires one step of random actions at each iteration. An idealized ChronoGEM with perfect density estimation is additionally provably approximating a uniform distribution over achievable goals as the number of sampled states is large enough, and it does not require any additive prior regarding state importance. Another close work also using a diffusion approach is UPSIDE (Kamienny et al., 2021). It finds a set of nodes along with a set of policies that connect any node to the closest ones, looks for new nodes by random exploration from the existing ones, and removes unnecessary nodes that are reached by the less discriminant policies. UPSIDE converges to a network of nodes that efficiently covers the state space. However, by construction UPSIDE does not produce uniform coverage, but a set of policies that reach different regions of the space. Using UPSIDE to obtain a similar quantity of states uniformly distributed than in ChronoGEM, we would have to train UPSIDE with 2^{17} policies, which is not computationally feasible.

Entropy maximisation. Some exploration algorithms, such as ChronoGEM, are constructed in order to maximize the entropy of the state visitation distribution. Most of them, however, focus on the distribution induced by the whole history buffer (instead of the just T-th states of episodes in ChronoGEM), generally based on the behavior of a trained policy. This is the case of MaxEnt (Hazan et al., 2019), GEM (Guo et al., 2021), SMM (Lee et al., 2019) and CURL (Geist et al., 2021). In APT (Liu and Abbeel, 2021b), instead of using a density model to estimate the entropy, they use a non-parametric approach based on the distance with the K nearest neighbors in a latent representation of the state space. APS (Liu and Abbeel, 2021a) combines APT's entropy bonus with an estimation of the cross-entropy based on successor features to maximize the mutual information $I(w; s)$ between a latent skill representations w and states.

Goal-Conditioned Reinforcement Learning. Goal-conditioned RL (Kaelbling, 1993; Schaul et al., 2015) is the general setup of learning a goal-conditioned policy instead of a specialized policy. We are particularly interested in goal-based setups where there is no *a-priori* reward function. Although well known works such as HER (Andrychowicz et al., 2017) demonstrate methods for learning goal-conditioned policies with minimal explicit exploration, more recent works (Pitis et al., 2020; OpenAI et al., 2021; Mendonca et al., 2021) demonstrate the importance of having a good curriculum of goals to train from. MEGA (Pitis et al., 2020) extends HER-style relabeling and answers the exploration problem by iteratively sampling goals according to a learnt density model of previous goals. ABC (OpenAI et al., 2021) demonstrates the importance of an adversarial curriculum for learning more complex goal-conditioned tasks, but is concentrated on specific tasks instead of arbitrary goal achievement. LEXA (Mendonca et al., 2021) builds on Plan2Explore (Sekar et al., 2020), and demonstrates the importance both of a good exploration mechanism, as well as the use of significant amounts of (imagined) data for learning an arbitrary goal-achievement policy. DIAYN (Eysenbach et al., 2018) uses a two-part mechanism that encourages the agent to explore novel areas for a given latent goal, while at the same time learning a goal embeddings for different areas of the state space. While some of the above methods consider notions of density for exploration (Eysenbach et al., 2018), C3PO uses a more principled exploration mechanism, and is particularly interested in collecting a large and uniformly distributed dataset of reachable states to train a goal-conditioned policy.

5 Conclusion

In this paper we looked at generating massive amounts of data on MDPs with no prior task information, as well as the effects of data quantity and quality on training a general policy. When compared to similar methods, we show that our proposed exploration mechanism ChronoGEM is capable of generating massive amounts of useful data. We also show that when training a downstream general controller policy with C3PO, the quality of ChronoGEM data for goal states is superior to similar methods, and that letting the policy consume massive amounts of experience data is fundamental to achieving high-quality general controllers. We believe these insights suggest that general controllers are achievable, but will require both performant exploration such as that generated by ChronoGEM and potentially massive amounts of data to be able to achieve high performance.

References

Andrychowicz, M., Wolski, F., Ray, A., Schneider, J., Fong, R., Welinder, P., McGrew, B., Tobin, J., Pieter Abbeel, O., and Zaremba, W. (2017). Hindsight experience replay. *Advances in neural information processing systems*, 30.

Badia, A. P., Sprechmann, P., Vitvitskyi, A., Guo, D., Piot, B., Kapturowski, S., Tieleman, O., Arjovsky, M., Pritzel, A., Bolt, A., et al. (2020). Never give up: Learning directed exploration strategies. *arXiv preprint arXiv:2002.06038*.

Bellemare, M., Srinivasan, S., Ostrovski, G., Schaul, T., Saxton, D., and Munos, R. (2016). Unifying count-based exploration and intrinsic motivation. *Advances in neural information processing systems*, 29.

Berner, C., Brockman, G., Chan, B., Cheung, V., Dębiak, P., Dennison, C., Farhi, D., Fischer, Q., Hashme, S., Hesse, C., et al. (2019). Dota 2 with large scale deep reinforcement learning. *arXiv preprint arXiv:1912.06680*.

Böttinger, K., Godefroid, P., and Singh, R. (2018). Deep reinforcement fuzzing. In *2018 IEEE Security and Privacy Workshops (SPW)*, pages 116–122. IEEE.

Burda, Y., Edwards, H., Storkey, A., and Klimov, O. (2018). Exploration by random network distillation. *arXiv preprint arXiv:1810.12894*.

Chowdhery, A., Narang, S., Devlin, J., Bosma, M., Mishra, G., Roberts, A., Barham, P., Chung, H. W., Sutton, C., Gehrmann, S., Schuh, P., Shi, K., Tsvyashchenko, S., Maynez, J., Rao, A., Barnes, P., Tay, Y., Shazeer, N., Prabhakaran, V., Reif, E., Du, N., Hutchinson, B., Pope, R., Bradbury, J., Austin, J., Isard, M., Gur-Ari, G., Yin, P., Duke, T., Levskaya, A., Ghemawat, S., Dev, S., Michalewski, H., Garcia, X., Misra, V., Robinson, K., Fedus, L., Zhou, D., Ippolito, D., Luan, D., Lim, H., Zoph, B., Spiridonov, A., Sepassi, R., Dohan, D., Agrawal, S., Omernick, M., Dai, A. M., Pillai, T. S., Pellat, M., Lewkowycz, A., Moreira, E., Child, R., Polozov, O., Lee, K., Zhou, Z., Wang, X., Saeta, B., Diaz, M., Firat, O., Catasta, M., Wei, J., Meier-Hellstern, K., Eck, D., Dean, J., Petrov, S., and Fiedel, N. (2022). Palm: Scaling language modeling with pathways.

Dinh, L., Sohl-Dickstein, J., and Bengio, S. (2016). Density estimation using real nvp. *arXiv preprint arXiv:1605.08803*.

Durkan, C., Bekasov, A., Murray, I., and Papamakarios, G. (2019). Neural spline flows. *Advances in neural information processing systems*, 32.

Ecoffet, A., Huizinga, J., Lehman, J., Stanley, K. O., and Clune, J. (2019). Go-explore: a new approach for hard-exploration problems. *arXiv preprint arXiv:1901.10995*.

Ecoffet, A., Huizinga, J., Lehman, J., Stanley, K. O., and Clune, J. (2021). First return, then explore. *Nature*, 590(7847):580–586.

Eysenbach, B., Gupta, A., Ibarz, J., and Levine, S. (2018). Diversity is all you need: Learning skills without a reward function. *arXiv preprint arXiv:1802.06070*.

Freeman, C. D., Frey, E., Raichuk, A., Girgin, S., Mordatch, I., and Bachem, O. (2021). Brax - a differentiable physics engine for large scale rigid body simulation.

Fu, J., Kumar, A., Nachum, O., Tucker, G., and Levine, S. (2020). D4rl: Datasets for deep data-driven reinforcement learning. *arXiv preprint arXiv:2004.07219*.

Geist, M., Pérolat, J., Laurière, M., Elie, R., Perrin, S., Bachem, O., Munos, R., and Pietquin, O. (2021). Concave utility reinforcement learning: the mean-field game viewpoint. *arXiv preprint arXiv:2106.03787*.

Germain, M., Gregor, K., Murray, I., and Larochelle, H. (2015). Made: Masked autoencoder for distribution estimation. In *International conference on machine learning*, pages 881–889. PMLR.

Gulcehre, C., Wang, Z., Novikov, A., Paine, T., Gómez, S., Zolna, K., Agarwal, R., Merel, J. S., Mankowitz, D. J., Paduraru, C., et al. (2020). Rl unplugged: A suite of benchmarks for offline reinforcement learning. *Advances in Neural Information Processing Systems*, 33:7248–7259.

Guo, Z. D., Azar, M. G., Saade, A., Thakoor, S., Piot, B., Pires, B. A., Valko, M., Mesnard, T., Lattimore, T., and Munos, R. (2021). Geometric entropic exploration. *arXiv preprint arXiv:2101.02055*.

Guo, Z. D., Thakoor, S., Pîslar, M., Pires, B. A., Altché, F., Tallec, C., Saade, A., Calandriello, D., Grill, J.-B., Tang, Y., et al. (2022). Byol-explore: Exploration by bootstrapped prediction. *arXiv preprint arXiv:2206.08332*.

Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S. (2018). Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pages 1861–1870. PMLR.

Hazan, E., Kakade, S., Singh, K., and Van Soest, A. (2019). Provably efficient maximum entropy exploration. In *International Conference on Machine Learning*, pages 2681–2691. PMLR.

Houthooft, R., Chen, X., Duan, Y., Schulman, J., De Turck, F., and Abbeel, P. (2016). Vime: Variational information maximizing exploration. *Advances in neural information processing systems*, 29.

Jumper, J., Evans, R., Pritzel, A., Green, T., Figurnov, M., Ronneberger, O., Tunyasuvunakool, K., Bates, R., Žídek, A., Potapenko, A., et al. (2021). Highly accurate protein structure prediction with alphafold. *Nature*, 596(7873):583–589.

Kaelbling, L. P. (1993). Learning to achieve goals. In *IJCAI*, volume 2, pages 1094–8. Citeseer.

Kamienny, P.-A., Tarbouriech, J., Lazaric, A., and Denoyer, L. (2021). Direct then diffuse: Incremental unsupervised skill discovery for state covering and goal reaching. *arXiv preprint arXiv:2110.14457*.

Lee, L., Eysenbach, B., Parisotto, E., Xing, E., Levine, S., and Salakhutdinov, R. (2019). Efficient exploration via state marginal matching. *arXiv preprint arXiv:1906.05274*.

Liu, H. and Abbeel, P. (2021a). Aps: Active pretraining with successor features. In *International Conference on Machine Learning*, pages 6736–6747. PMLR.

Liu, H. and Abbeel, P. (2021b). Behavior from the void: Unsupervised active pre-training. *Advances in Neural Information Processing Systems*, 34:18459–18473.

Mendonca, R., Rybkin, O., Daniilidis, K., Hafner, D., and Pathak, D. (2021). Discovering and achieving goals via world models. *Advances in Neural Information Processing Systems*, 34:24379–24391.

Mirhoseini, A., Goldie, A., Yazgan, M., Jiang, J. W., Songhori, E., Wang, S., Lee, Y.-J., Johnson, E., Pathak, O., Nazi, A., et al. (2021). A graph placement methodology for fast chip design. *Nature*, 594(7862):207–212.

OpenAI, O., Plappert, M., Sampedro, R., Xu, T., Akkaya, I., Kosaraju, V., Welinder, P., D'Sa, R., Petron, A., Pinto, H. P. d. O., et al. (2021). Asymmetric self-play for automatic goal discovery in robotic manipulation. *arXiv preprint arXiv:2101.04882*.

Ostrovski, G., Bellemare, M. G., Oord, A., and Munos, R. (2017). Count-based exploration with neural density models. In *International conference on machine learning*, pages 2721–2730. PMLR.

Papamakarios, G., Pavlakou, T., and Murray, I. (2017). Masked autoregressive flow for density estimation. *Advances in neural information processing systems*, 30.

Pitis, S., Chan, H., Zhao, S., Stadie, B., and Ba, J. (2020). Maximum entropy gain exploration for long horizon multi-goal reinforcement learning. In *International Conference on Machine Learning*, pages 7750–7761. PMLR.

Raileanu, R. and Rocktäschel, T. (2020). Ride: Rewarding impact-driven exploration for procedurally-generated environments. *arXiv preprint arXiv:2002.12292*.

Roy, J., Girgis, R., Romoff, J., Bacon, P.-L., and Pal, C. J. (2022). Direct behavior specification via constrained reinforcement learning. In Chaudhuri, K., Jegelka, S., Song, L., Szepesvari, C., Niu, G., and Sabato, S., editors, *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 18828–18843. PMLR.

Schaul, T., Horgan, D., Gregor, K., and Silver, D. (2015). Universal value function approximators. In *International conference on machine learning*, pages 1312–1320. PMLR.

Schmidt, J., Marques, M. R., Botti, S., and Marques, M. A. (2019). Recent advances and applications of machine learning in solid-state materials science. *npj Computational Materials*, 5(1):1–36.

Schwartz, J. and Kurniawati, H. (2019). Autonomous penetration testing using reinforcement learning. *arXiv preprint arXiv:1905.05965*.

Schwarzer, M., Anand, A., Goel, R., Hjelm, R. D., Courville, A., and Bachman, P. (2020). Data-efficient reinforcement learning with self-predictive representations. *arXiv preprint arXiv:2007.05929*.

Sekar, R., Rybkin, O., Daniilidis, K., Abbeel, P., Hafner, D., and Pathak, D. (2020). Planning to explore via self-supervised world models. In *International Conference on Machine Learning*, pages 8583–8592. PMLR.

Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A., et al. (2017). Mastering the game of go without human knowledge. *nature*, 550(7676):354–359.

Smith, L., Kew, J. C., Peng, X. B., Ha, S., Tan, J., and Levine, S. (2022). Legged robots that keep on learning: Fine-tuning locomotion policies in the real world. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 1593–1599. IEEE.

Sutton, R. (2019). The bitter lesson. *Incomplete Ideas (blog)*, 13:12.

Sutton, R. S. and Barto, A. G. (2018). *Reinforcement learning: An introduction*. MIT press.

Tan, J., Zhang, T., Coumans, E., Iscen, A., Bai, Y., Hafner, D., Bohez, S., and Vanhoucke, V. (2018). Sim-to-real: Learning agile locomotion for quadruped robots. *arXiv preprint arXiv:1804.10332*.

Uria, B., Murray, I., and Larochelle, H. (2013). Rnade: The real-valued neural autoregressive density-estimator. *Advances in Neural Information Processing Systems*, 26.

A Uniform sub-sampling

Let f_X be the density of a distribution with domain $S \subset \mathbb{R}^d$ ($x \in S \Leftrightarrow f_X(x) > 0$), and $X_1 \dots X_n \sim f_X$ iid. We assume that S is bounded. We define the sub-sampling Y_n such that $P(Y_n = X_i | X_1 \dots X_n) \propto \frac{1}{f_X(X_i)}$, and $Y \sim \mathcal{U}_S$ a random vector following the uniform distribution over S.

Theorem 1. *Y_n conditioned to $X_1 \dots X_n \sim f_X$ iid converges in distribution to Y when n goes to infinity.*

Proof. let A be any subset of S, and $\mu(.)$ the Lebesgue measure over S. Given any sampling $X_1 \dots X_n \sim f_X$ iid, we have:

$$P(Y_n \in A | X_1 \dots X_n) = \frac{\sum_{X_i \in A} \frac{1}{f_X(X_i)}}{\sum_{j=1}^n \frac{1}{f_X(X_j)}} = \frac{\frac{1}{n}\sum_{X_i \in A} \frac{1}{f_X(X_i)}}{\frac{1}{n}\sum_{j=1}^n \frac{1}{f_X(X_j)}} \to \frac{\mathbb{E}_X[\mathbb{1}_{X \in A} \frac{1}{f_X(X)}]}{\mathbb{E}_X[\frac{1}{f_X(X)}]} = \frac{\mu(A)}{\mu(S)}.$$

□

B Density Estimator Model Selection

We implemented 7 candidate models, including Gaussian models (Multivariate, Mixture), autoregressive networks (RNade (Uria et al., 2013), Made (Germain et al., 2015)), and normalizing flows (real-NVP (Dinh et al., 2016), Maf (Papamakarios et al., 2017), NSF (Durkan et al., 2019)). We decided to compare them on two continuous control task from the Brax environments (Freeman et al., 2021): Ant and HalfCheetah.

We proceeded in 5 steps.

B.1 Policy pre-training

We first pre-trained policies to solve all the different tasks, using 4 different RL algorithms: a uniform random walk, PPO, SAC and Evolution Strategies.

B.2 Density model training

For each pre-trained policy, we trained each density model with various hyperparameters configurations to maximize the log-density of the states visitation. Table 2 reports, for each model, all the values we tried for each hyperparameter of the model, and the different configurations we used are all combinations of these values. For each model, the training consisted in 100 epochs, each one containing a training phase interacting with 128×1000 environment transitions and an evaluation phase based on the log-likelihood score averaged over 128×1000 visited states. Every training was run over 5 different seeds and resulting scores were averaged across these seeds.

B.3 Configuration selection

For each model, we selected the best configuration across all environments and all RL algorithm, based on AUC. For each model, to compare the performance of configuration across various environment and algorithm, we normalized the score w.r.t. the max and min performances for each environment and RL algorithm couple. This procedure is described in Algo 3. Bold values in Table 2 represent the hyper from the best configuration we found for each model.

B.4 Hyperparameters effects

We studied the effect of hyper parameters, based on two approaches. In the first one, given each value of each single hyper we selected the best configuration for the rest of hyper parameters and looked at the resulting score. In the second one, given a selected best configuration (from the step described above in B.3) we replaced the value of one hyper by another one and looked how it deteriorated the score. However, we found no surprising effect: globally, the larger the models, the higher the score. Parameters that did not affect the size of the model (for ex the batch size) had no significant effect, especially on normalizing flows.

Gaussian		**Made**	
Learning rate	5e-4, **1e-4**, 1e-3	Learning rate	**5e-4**, 1e-4, 5e-4
Batch size	32, 64, **128**	Batch size	**32**, 64, 128
Mixture of Gaussian		Num. masks	1, **2**, 4
Learning rate	5e-4, 1e-4, **1e-3**	Num. Mixtures	1, 4, **8**
Batch size	32, 64, **128**	Num. hidden layers	**2**
Num. mixtures	10, 20, **30**	Hidden layers dim.	5, 10, **50**
RNade		**real-NVP, Maf, NSF**	
Learning rate	5e-4, **1e-4**, 5e-4	Learning rate	**5e-4**, 1e-4, 5e-4
Batch size	**32**, 64, 128	Batch size	32, 64, **128** (Real-NVP: **64**)
Num. mixtures	1, 10, **20**, 30	Num. layers	4, 6, **8** (MAF: **6**)
Hidden layer dim.	10, 20, **50**	hidden layers per layers	**500×500**

Table 2: Explored hyperparameters. Bolded values are correspond to the best configuration for each model, based on AUC criteria. For normalizing flows, we found the same best configurations, except a batch size of 64 for Real-NVP (128 for MAF and NSF) and 6 layers for MAF (8 for Real-NVP and NSF).

Algorithm 3 Hyper parameters selection for a given model.

```
Input: $\{\text{rl}_i\}$ a set of RL algorithms, $\{\text{env}_j\}$ a set of environment and $\{\text{conf}_k\}$ a set of hyperparams configuration. Score function $S(\text{rl}_i, \text{env}_j, \text{conf}_k) \in \mathbb{R}$, for ex AUC.

Normalize configurations scores for each RL algorithm and environment:
For $\text{rl}_i, \text{env}_j \in \{\text{rl}_i\} \times \{\text{env}_j\}$:
    $M(\text{rl}_i, \text{env}_j) = \max_k S(\text{rl}_i, \text{env}_j, \text{conf}_k)$
    $m(\text{rl}_i, \text{env}_j) = \min_k S(\text{rl}_i, \text{env}_j, \text{conf}_k)$
    For $\text{conf}_k \in \{\text{conf}_k\}$:
        $\bar{S}(\text{rl}_i, \text{env}_j, \text{conf}_k) = \frac{S(\text{rl}_i, \text{env}_j, \text{conf}_k) - m}{M - m}$

Sum normalized scores across different RL algorithms and environments:
For $\text{conf}_k \in \{\text{conf}_k\}$:    $\mathcal{F}(\text{conf}_k) = \sum_{i,j} \bar{S}\text{rl}_i, \text{env}_j, \text{conf}_k)$

Return best configuration according to sum of normalized scores:
Return: $\text{argmax}_k \mathcal{F}(\text{conf}_k)$
```

B.5 Model comparison

Finally, we compared the different models together on the different environments and RL algorithms, when run with the selected best configuration (from step B.3). Visually Gaussian and Mixture of Gaussians are less efficient than autoregressive models, which are less efficient than normalizing flows. Among normalizing flows, we found that NSF had the best average score across all RL algorithm and environments, while being the less sensible to the variations of hyper parameters. Figure 7 reports bar plots comparison of all model's AUC scores given their best hyper configurations on all the different RL algorithms and environments, averaged over 5 seeds.

C Space visitation of 2D environments

See Figure 8.

C.0.1 Resettable states assumption

Similarly to other diffusion-based algorithms (see related works **??**), ChronoGEM needs to explore many actions from a single given state. Although this is an unrealistic assumption for applications involving real-world systems, we argue that there are multiple scenarios where this is an acceptable assumption. To begin with, many tasks exist only as software: computer games (Berner et al., 2019; Roy et al., 2022), physics simulations (Jumper et al., 2021; Schmidt et al., 2019), software-aided design (Mirhoseini et al., 2021) or even arbitrary programs (Schwartz and Kurniawati, 2019;

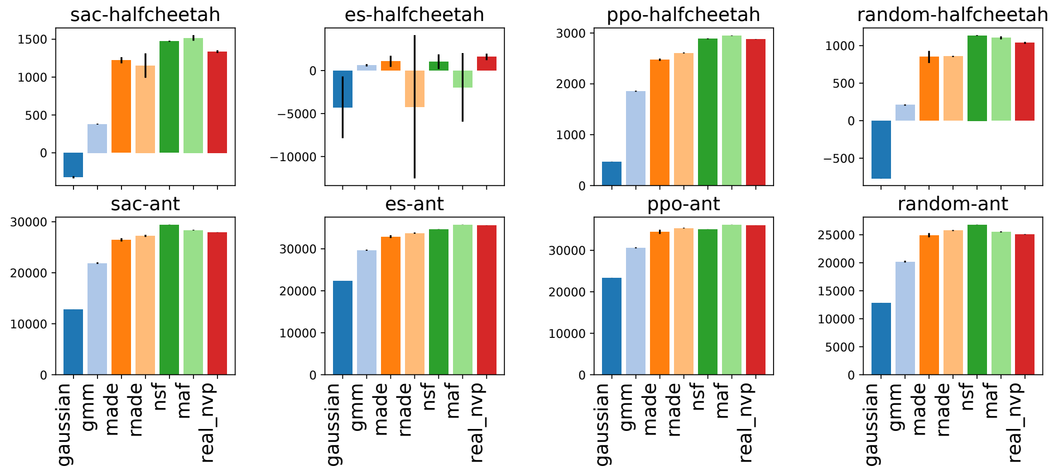


Figure 7: Model comparison based on AUC with selected best hyper configuration across all RL algorithms and environments, averaged over 5 seeds.

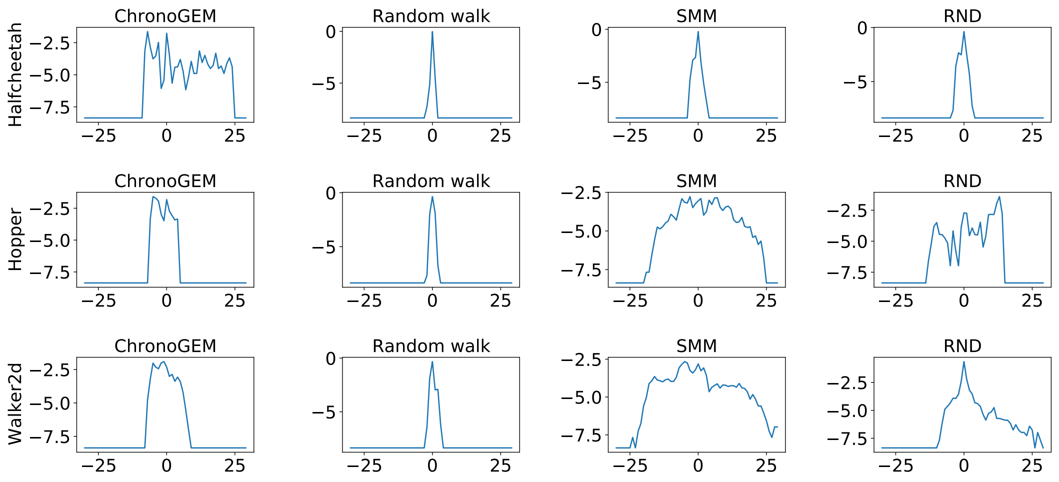


Figure 8: Log-frequencies of discretised X-axis visitations in the 2-dimensional environments (Hopper, Walker2d and Halfcheetah). In Hopper and Halfcheetah, SMM and RND visited a larger scope of spatial positions, but actually neglected to explore the possible poses, while ChronoGEM had a more uniform behaviour and well balanced both poses and positions exploration. In Halfcheetah, only ChronoGEM was able to run a decent exploration under the action reduction with a low multiplier.

Böttinger et al., 2018) are all important tasks that can be arbitrarily reset and parallelized. Secondly, sim2real (Tan et al., 2018; Smith et al., 2022) approaches allow for a policy trained on simulation to be transferred to real systems. In the case where a particular robot embodiment such as a quadruped walker might be used for a large number of task, it would make sense to invest the time in establishing a high-fidelity simulator and a sim2real pipeline to train generalist controllers such as those generated by C3PO.

D Exploration methods to reaching states from other exploration methods

See Figure 9.

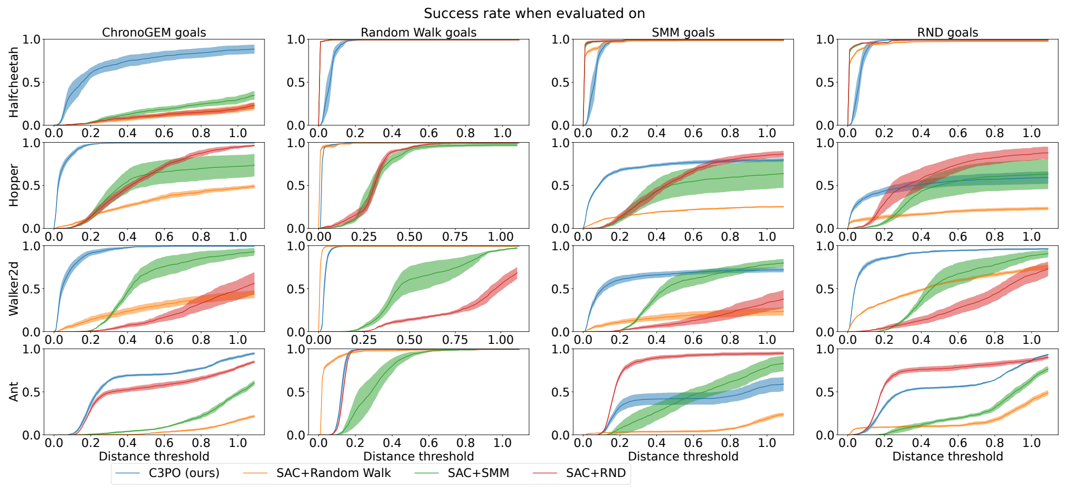


Figure 9: For each environment (lines) and each set of evaluation goals (columns), success rates as a function of distance thresholds obtained by SAC when trained on the different sets of training goals (ChronoGEM, Random Walk, SMM, RND). Each exploration algorithm was run over 3 seeds to collect evaluation and training goals, and each SAC training was also run over 3 seeds, so the resulting curves are actually averaging 9 different values.

E Method Hyper-Parameters

See Table 3 and Table 4 for C3PO hypers.

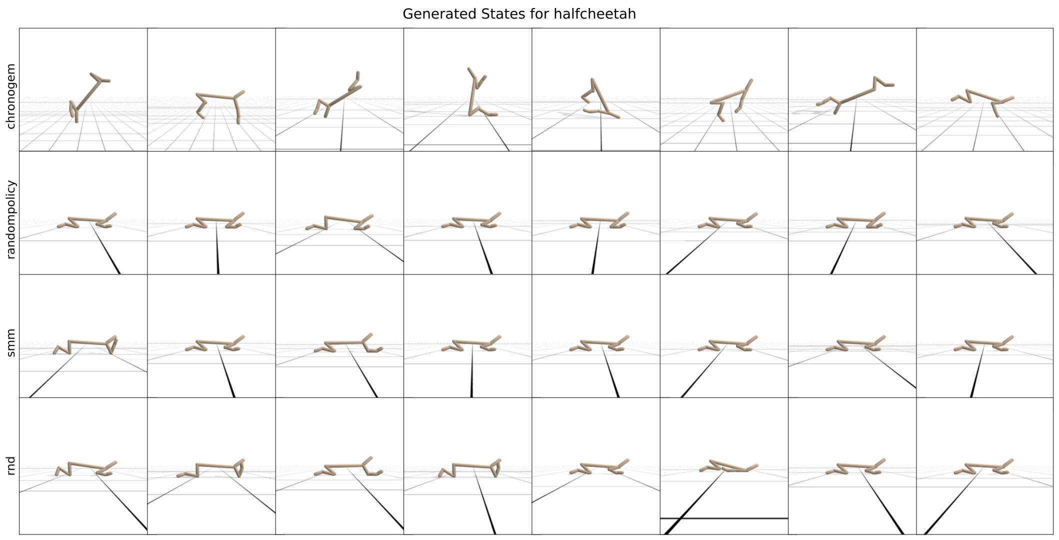
E.1 Specifics of ChronoGEM for Humanoid

By default, ChronoGEM would mostly explore positions where the humanoid is on the floor. However, it was simple to modulate the algorithm to only explore uniformly in the space of state where the humanoid is standing. For example, on can associate zero weight to undesired states during the re-sampling step. Thus, we avoid states in which the torso drops below an altitude of .8 (the default failure condition). ChronoGEM is modified to not draw states where the humanoid is too low. The goal-conditioned learner gets a high penalty for going too low as well. This demonstrates that when in posession of a prior, we can leverage it to steer the exploration and policy learning.

E.2 Specifics for Zero-Shot Imitation

Since reaching a target state is never immediate and requires at least a few action steps, we subsampled the expert trajectory to take one state every n states as a target for the imitating agent. Harder tasks would require higher values of n, making the imitation task less strict.
Since the agent was trained on ChronoGEM targets that are reachable in 128 steps, we considered relatively small episodes of 300 steps. Otherwise, as the imitator is moving slower than the expert, at some point the distance between the imitator's state and the target expert's states goes out of C3PO's domain of knowledge. For downstream tasks, we could directly evaluate the quality of imitations by looking at the environment's returns. We used a single expert demonstration and averaged results over 9 different values (3 ChronoGEM seeds for C3PO $\times$ 3 environment seeds). Full results of policy-imitation are available in Table 1.

F Example States Generated by Each Method



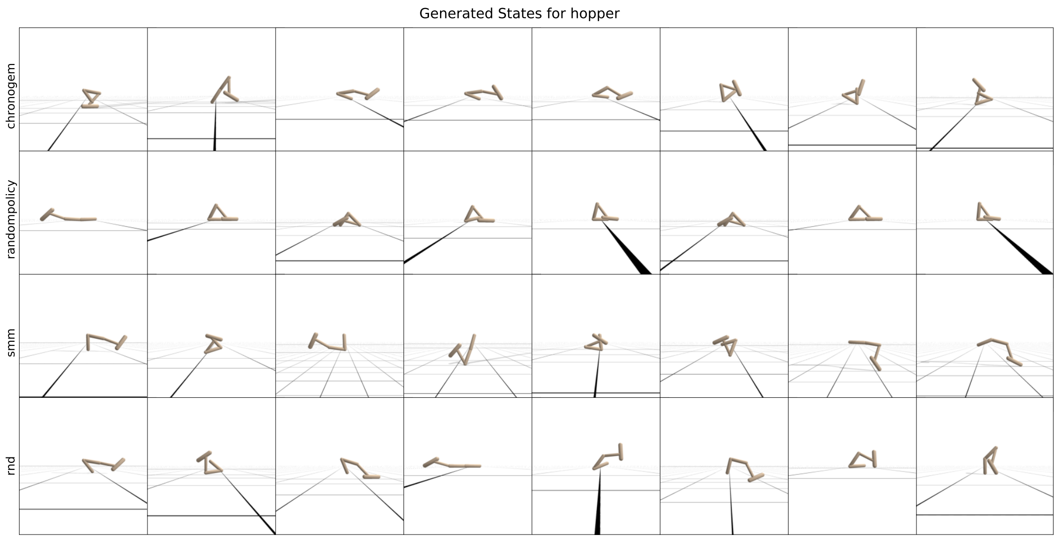
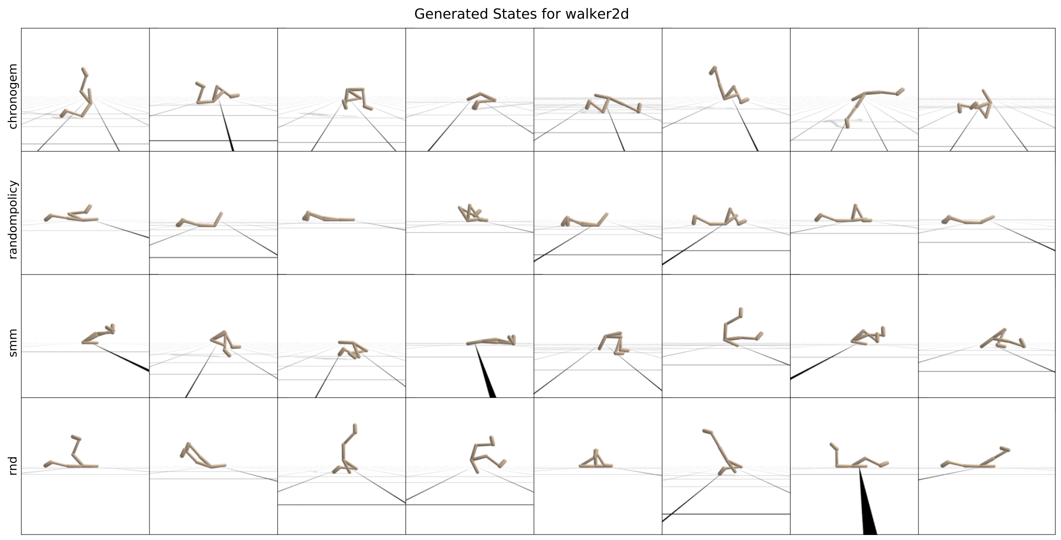


Figure 10: Random samples of generated states from the various exploration methods investigated for the HalfCheetah and Hopper environments. We can observe reduced variability for `randompolicy` states, as well as `smm` states in halfcheetah. These correspond to the reduces overall performance of policies trained on these datasets for HalfCheetah and Ant. Each visualized state is sampled directly from the dataset used for training goal-conditioned policies.



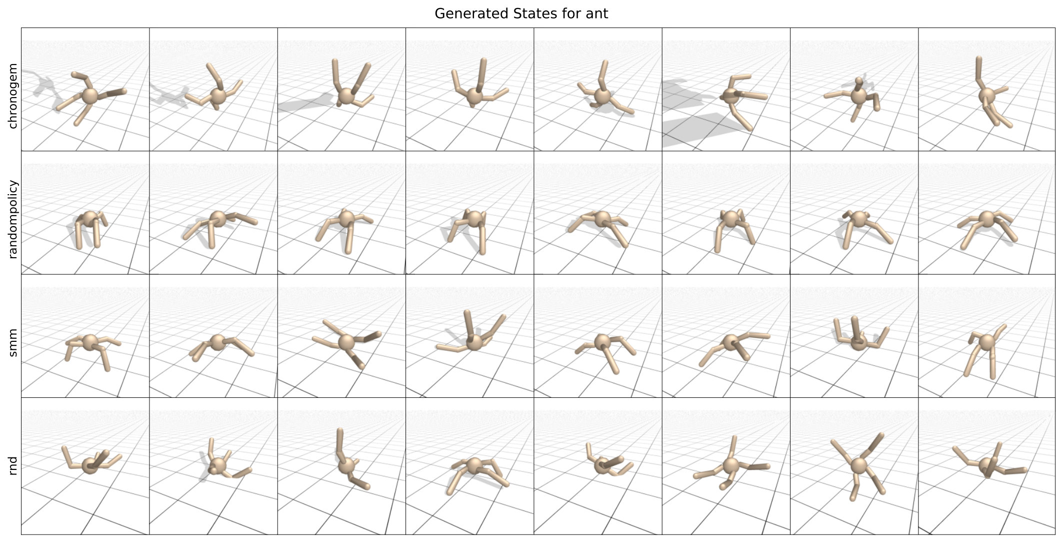


Figure 11: Random samples of generated states from the various exploration methods investigated for the Walker2D and Ant environments. As in the previous figure we can observe reduced variability for `randompolicy` states. Each visualized state is sampled directly from the dataset used for training goal-conditioned policies.

ChronoGEM Hypers	
buffer size	2^{17}
time horizon	128
branching factor	4
NSF batch size	2^{11}
NSF learning rate	3e-5
NSF hidden layers size	[512, 512]
NSF number of hidden layers	8
NSF training epochs per step	1

Table 3: ChronoGEM Hyperparameters

SAC Hypers	
normalize observations	True
reward scaling	1.
number of actors	2^{10}
batch size	1024
discounting	.98
learning rate	3e-5
min replay size	2^{13}
max replay size	2^{20}
epsilon update threshold	.9
epsilon update multiplier	.99
gradient updates per actor episode	32 * episode length
networks size	(1024) * 4

Table 4: C3PO-SAC Hyperparameters