Transferable Reinforcement Learning via Generalized Occupancy Models

Chuning Zhu
University of Washington
Seattle, WA 98195
zchuning@cs.washington.edu

Xinqi Wang
University of Washington
Seattle, WA 98195
wxqkaxdd@cs.washington.edu

Tyler Han
University of Washington
Seattle, WA 98195
than123@cs.washington.edu

Simon Shaolei Du
University of Washington
Seattle, WA 98195
ssdu@cs.washington.edu

Abhishek Gupta
University of Washington
Seattle, WA 98195
abhgupta@cs.washington.edu

Abstract

Intelligent agents must be generalists, capable of quickly adapting to various tasks. In reinforcement learning (RL), model-based RL learns a dynamics model of the world, in principle enabling transfer to arbitrary reward functions through planning. However, autoregressive model rollouts suffer from compounding error, making model-based RL ineffective for long-horizon problems. Successor features offer an alternative by modeling a policy's long-term state occupancy, reducing policy evaluation under new tasks to linear reward regression. Yet, policy improvement with successor features can be challenging. This work proposes *a novel class of models*, i.e., generalized occupancy models (GOMs), that learn a distribution of successor features from a stationary dataset, along with a policy that acts to realize different successor features. These models can quickly select the optimal action for arbitrary new tasks. By directly modeling long-term outcomes in the dataset, GOMs avoid compounding error while enabling rapid transfer across reward functions. We present a practical instantiation of GOMs using diffusion models and show their efficacy as a new class of transferable models, both theoretically and empirically across various simulated robotics problems. Videos and code: https://weirdlabuw.github.io/gom/.

1 Introduction

Reinforcement learning (RL) agents are ubiquitous in a wide array of applications, from language modeling [7] to robotics [21, 27]. Traditionally, RL has focused on the single-task setting, learning behaviors that maximize a specific reward function. However, for practical deployment, RL agents must be able to generalize across different reward functions within an environment. For example, a robot deployed in a household setting should not be confined to a single task such as object relocation but should handle various tasks, objects, initial and target locations, and path preferences.

This work addresses the challenge of developing RL agents that can broadly generalize to *any* task in an environment specified by a reward function. To achieve this type of generalization, we consider

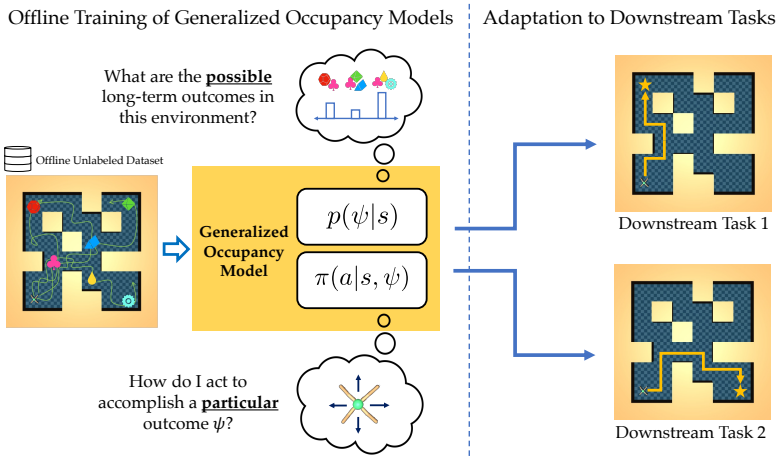

Figure 1: The transfer setting for generalized occupancy models. Given an unlabeled offline dataset, we learn a generalized occupancy model that models both "what can happen?" $p(\psi|s)$ and "how can we achieve a particular outcome?" $p(a|s,\psi)$. This is used for quick adaptation to new downstream tasks without test-time policy optimization.

the paradigm of pretraining on an offline dataset of transitions and inferring optimal policies for downstream tasks from observing task-specific rewards. Since the target task is not revealed during pretraining, the model must encode information about the environment dynamics without committing to a particular policy or reward. Moreover, once the task reward is observed, the model must provide a way to evaluate and improve the policy since different tasks require different optimal policies.

A natural approach to this problem is model-based reinforcement learning [60, 20, 61], which learns an approximate dynamics model of the environment. Given a downstream reward function, task-optimal behavior can be obtained by "planning" via model rollouts [49, 57, 38, 44]. Typically, model rollouts are generated in an autoregressive fashion, conditioning each step on generation from the previous steps. In practice, however, autoregressive generation notoriously suffers from *compounding error* [30, 1, 25], which arises when small, one-step approximation errors accumulate over long horizons. This leads to rollout trajectories that diverge from real trajectories, limiting many model-based RL methods to short-horizon, low-dimensional problems.

A alternative class of algorithms based on *successor features* (SFs) has emerged as a potential approach to transferable decision-making [3, 2]. Successor features represent the discounted sum of features for a given policy. Assuming a linear correspondence between features and rewards, policy evaluation under new rewards reduces to a simple linear regression problem. Notably, by directly predicting long-term outcomes, SFs avoid autoregressive rollouts and hence compounding error. However, the notion of successor features is deeply tied to the choice of a particular policy. This policy dependence poses a challenge when recovering the optimal policy for downstream tasks. Current approaches to circumvent policy dependence either maintain a set of policies and select the best one during inference [3] or randomly sample reward vectors and make conditional policy improvements [5, 53, 54]. Nevertheless, a turnkey solution to transfer remains a desirable goal.

In this work, we propose a new class of models—*generalized occupancy models (GOMs)*—that are rapidly transferable across reward functions while avoiding compounding errors. Rather than modeling the successor features under a particular policy, GOMs model the entire *distribution* of successor features under the *behavior policy*, effectively encoding all possible outcomes that appear in the dataset starting from a designated state. Crucially, by representing outcomes as successor features, we enjoy the benefit of zero-shot outcome evaluation after solving a linear reward regression problem, while avoiding compounding error. To enable transfer to downstream tasks, we jointly learn a readout policy that generates an action to accomplish a particular long-horizon outcome (represented as a successor feature) from a state. At test time, we plan by selecting the best in-distribution outcome from the current state for a particular reward and query the readout policy for an optimal action to realize it. *GOMs are a new class of world models as they essentially capture the dynamics of the world and can be used to plan for arbitrary rewards in new tasks.*

Since multiple outcomes can follow from a particular state, and multiple actions can be taken to achieve a particular outcome, both the outcome distribution and the policy require expressive model classes to represent. We provide a practical instantiation of GOMs using diffusion models [22, 46] and show that under this parameterization, planning can be cast as a simple variant of guided diffusion sampling [12]. We validate the transferability of GOMs across a suite of long-horizion simulated robotics domains and further provide analysis showing that GOMs provably converge to "best-in-data" policies. With GOMs, we hope to introduce a new way for the research community to envision transfer in reinforcement learning beyond model-based RL.

2 Related Work

Our work has connections to numerous prior work on model-based RL and successor features.

Model-Based RL To enable transfer across rewards, model-based RL learns one-step (or multi-step) dynamics models via supervised learning and use them for planning [10, 37, 38, 57] or policy optimization [11, 49, 25, 63, 19]. These methods typically suffer from compounding error, where autoregressive model rollouts lead to large prediction errors over time [1, 30]. Despite improvements to model architectures [19, 24, 1, 31, 64] and learning objectives [26], modeling over long horizons without compounding error remains an open problem. GOMs instead directly model cumulative long-term outcomes in an environment, avoiding autoregressive generation while remaining transferable.

Successor Features Successor features achieve generalization across rewards by modeling the accumulation of features (as opposed to rewards in model-free RL) [3, 2]. With the assumption that rewards are linear in features, policy evaluation under new rewards reduces to a linear regression problem. A key limitation of successor features is their inherent dependence on a single policy, as they are defined as the accumulated features when acting according to a particular policy. This makes extracting optimal policies for new tasks challenging.

To circumvent this policy dependence, generalized policy improvement [3, 2] maintains a discrete set of policies and selects the highest valued one to execute at test time, limiting the space of available policies for new tasks. Universal SF [6] and forward-backward representations [53, 54] randomly sample reward weights z and jointly learn successor features and policies conditioned on z. The challenge lies in achieving coverage over the space of all possible policies through sampling of z, resulting in potential distribution shifts for new problems. RaMP [8] learns a successor feature predictor conditioned on an initial state and a sequence of actions. Transfer requires planning by sampling actions sequences, which becomes quickly intractable over horizon. In contrast, GOMs avoid conditioning on any explicit policy representation by modeling the distribution of all possible outcomes represented in a dataset, and then selecting actions corresponding to the most desirable long term outcome.

Distributional Successor Measure (DSM) [58] is a concurrent work that learns a distribution over successor representations using tools from distributional RL [4]. Importantly, DSM models the distributional successor measure of a *particular* policy, where the stochasticity stems purely from the policy and the dynamics. This makes it suitable for robust policy evaluation but not for transferring to arbitrary downstream tasks. In contrast, GOMs model the distribution of successor feature outcomes in the dataset (i.e., the behavior policy). Here the distribution stems from range of meaningfully distinct long-term outcomes. This type of modeling allows GOMs to extract optimal behvavior for arbitrary downstream tasks, while DSMs suffer from the same policy dependence that standard successor feature-based methods do.

3 Preliminaries

We adopt the standard Markov Decision Process (MDP) notation and formalism [23] for an MDP $\mathcal{M} = (\mathcal{S}, \mathcal{A}, r, \gamma, \mathcal{T}, \rho_0)$, but restrict our consideration to the class of deterministic MDPs. While this does not encompass every environment, it does capture a significant set of problems of practical interest. Hereafter, we refer to a deterministic MDP and a *task* interchangeably. In our setting, we consider transfer across different tasks that always share the same action space $\mathcal{A}$, state space $\mathcal{S}$, and

transition dynamics $\mathcal{T} : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$[1] The difference between tasks only lies in having different state-dependent Markovian reward functions $r : \mathcal{S} \to [0, 1]$.

Value Functions and Successor Features Let $R = \sum_{t=1}^{\infty} \gamma^{t-1} r(s_t)$ denote the cumulative reward for a trajectory $\{s_i, a_i\}_{i=1}^{\infty}$. One can then define the state value function under policy π as $V^\pi(s) := \mathbb{E}_{\pi,\mathcal{T}}[R \mid s_1 = s]$, and the state-action value function as $Q^\pi(s, a) := \mathbb{E}_{\pi,\mathcal{T}}[R \mid s_1 = s, a_1 = a]$. The value function admits a temporal structure that allows it to be estimated using dynamic programming, which iteratively applies the Bellman operator until a fixed point is reached $V^\pi(s) := r(s) + \gamma \mathbb{E}_\pi [V^\pi(s_2) \mid s_1 = s]$. While these Bellman updates are in the tabular setting, equivalent function approximator variants (e.g., with neural networks) can be instantiated to minimize a Bellman "error" with stochastic optimization techniques [36, 18, 35].

Successor features [2] generalize the notion of a value function from task-specific rewards to task-agnostic features. Given a state feature function $\phi : S \to \mathbb{R}^d$, the successor feature of a policy is defined as $\psi^\pi(s) = \mathbb{E}_{\pi,\mathcal{T}}\left[\sum_{t=1}^{\infty} \gamma^{t-1} \phi(s_i) \mid s_1 = s\right]$. Suppose rewards can be linearly expressed by the features, i.e. there exists $w \in \mathbb{R}^n$ such that $R(s) = w^\top \phi(s)$, then the value function for the particular reward can be linearly expressed by the successor feature $V^\pi(s) = w^\top \psi^\pi(s)$. Hence, given the successor feature ψ^π of a policy π, we can immediately compute its value under any reward once the reward weights w are known. Analogous to value functions, successor features also admit a recursive Bellman identity $\psi^\pi(s) := \phi(s) + \gamma \mathbb{E}_\pi [\psi^\pi(s')]$, allowing them to be estimated using dynamic programming [3]. In this paper, with a slight abuse of terminology, we also refer to the discounted sum of features along a *trajectory* as a successor feature. In this sense, a successor feature represents an outcome that is feasible under the dynamics and can be achieved by some policy.

Diffusion Models GOMs rely on expressive generative models to represent the distribution of successor features. Diffusion models [22, 46] are a class of generative models where data generation is formulated as an iterative denoising process. Specifically, DDPM [22] consists of a forward process that iteratively adds Gaussian noise to the data, and a corresponding reverse process that iteratively denoises a unit Gaussian to generate samples from the data distribution. The reverse process leverages a neural network estimating the score function of each noised distribution, trained with a denoising score matching objective [48]. In addition, one can sample from the conditional distribution $p(x|y)$ by adding a guidance $\nabla_x \log p(y|x)$ to the score function in each sampling step [12]. As we show in Sec. 4.3, guided diffusion enables quick sampling of optimal outcomes from GOMs.

Problem setting We consider a transfer learning scenario with access to an offline dataset $\mathcal{D} = \{(s_i, a_i, s'_i)\}_{i=0}^{N}$ of transition tuples collected with some behavior policy π_β under dynamics $\mathcal{T}$. The goal is to quickly obtain the optimal policy π^* for some downstream task, specified in the form of a reward function or a number of (s, r) samples. While we cannot hope to extrapolate beyond the dataset (as is common across problems in offline RL [33]), we will aim to find the best policy *within* dataset coverage for the downstream task. This is defined more precisely in Section 4.2.

4 Generalized Occupancy Models

We introduce the framework of GOMs as a scalable approach to the transfer problem described in Section 3, with the goal of learning from an unlabeled dataset to quickly adapt to any downstream task specified by a reward function. We start by relating the technical details behind learning GOMs in Section 4.1, followed by explaining how GOMs can be used for efficient multi-task transfer in Section 4.2. Finally, we describe a practical instantiation of GOMs in Section 4.3.

4.1 Learning Generalized Occupancy Models

To transfer and obtain optimal policies across different reward functions, generalist decision-making agents must model the future in a way that permits the evaluation of new rewards *and* new policies. To this end, GOMs adopt a technique based on off-policy dynamic programming to directly model the distribution of cumulative future outcomes, without committing to a particular reward function $r(\cdot)$ or policy π. Fig. 2 illustrates the two components in GOMs, and we describe each below.

(1) Outcome model: for a particular a state feature function $\phi(s)$, GOMs model the distribution of successor features $p(\psi|s)$ over all paths that have coverage in the dataset. In deterministic MDPs,

[1] For simplicity, we also use $\mathcal{T}(s, a)$ to denote the next state.

each successor feature ψ (discounted sum of features $\psi = \sum_t \gamma^{t-1}\phi(s_t)$) can be regarded as an "outcome". When the state features are chosen such that reward for the desired downstream task is a linear function of features, i.e., there exists $w \in \mathbb{R}^n$ such that $r(s) = w^\top \psi(s)$ [41, 42, 65, 55, 8, 3], the value of each outcome can be evaluated as $w^\top \psi$. That is, knowing w effectively transforms the distribution of outcomes $p(\psi|s)$ into a distribution of task-specific values (sum of rewards) $p(R|s)$. Notably, since w can be estimated by regressing rewards from features, distributional evaluation on a new reward function boils down to a simple linear regression.

As in off-policy RL, the outcome distribution $p(\psi|s)$ in GOMs can be learned via an approximate dynamic programming update, which is similar to a distributional Bellman update [4]:

$$\begin{aligned} \max_{\theta} \quad & \mathbb{E}_{(s,a,s')\sim\mathcal{D}}\left[\log p_\theta(\phi(s) + \gamma\psi_{s'}|s)\right] \\ \text{s.t} \quad & \psi_{s'} \sim p_\theta(\cdot|s') \end{aligned} \tag{1}$$

Intuitively, this update suggests that the distribution of successor features $p_\theta(\psi|s)$ at state s maximizes likelihood over current state feature $\phi(s)$ added to sampled future outcomes $\psi_{s'}$. This instantiates a fixed-point procedure, much like a distributional Bellman update. An additional benefit of the dynamic programming procedure is trajectory stitching, where combinations of subtrajectories in the dataset will be represented in the outcome distribution.

(2) Readout policy: Modeling the distribution of future outcomes in an environment is useful only when it can be realized in terms of actions that accomplish particular outcomes. To do so, GOMs pair the outcome model with a readout policy $\pi(a|s,\psi)$ that actualizes a desired long-term outcome ψ into the action a to be taken at state s. Along with the outcome model $p_\theta(\psi|s)$, the readout policy $\pi_\rho(a|s,\psi)$ can be optimized via maximum-likelihood estimation:

$$\begin{aligned} \max_{\rho} \quad & \mathbb{E}_{(s,a,s')\sim\mathcal{D}}\left[\log \pi_\rho(a|s, \psi = \phi(s) + \gamma\psi_{s'})\right] \\ \text{s.t} \quad & \psi_{s'} \sim p_\theta(.|s') \end{aligned} \tag{2}$$

This update states that if an action a at a state s leads to a next state s', then a should be taken with high likelihood for outcomes ψ, which are a combination of the current state feature $\phi(s)$ and future outcomes $\psi_{s'} \sim p_\theta(\cdot|s')$.

The outcome distribution $p(\psi|s)$ can be understood as a natural analogue to a value function, but with two crucial differences: (1) it represents the accumulation of not just a single reward function but an arbitrary feature (with rewards being linear in this feature space), and (2) it is not specific to any particular policy but represents the distribution over all cumulative outcomes covered in the dataset. The first point enables transfer across rewards, while the second enables the selection of optimal actions for new rewards rather than being restricted to a particular (potentially suboptimal) policy. Together with the readout policy $\pi(a|s,\psi)$, these models satisfy our desiderata for transfer, i.e., that the value for new tasks can be estimated by simple linear regression without requiring autoregressive generation, and that optimal actions can be obtained without additional policy optimization.

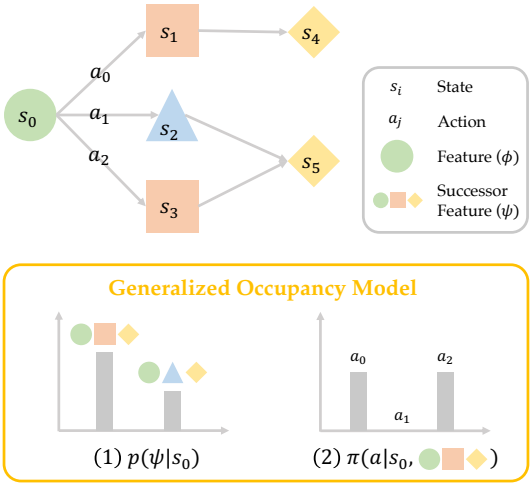

Figure 2: Generalized Occupancy Model for a simple environment. Given a state feature function ϕ, GOMs learn a distribution of all possible long-term outcomes (successor features ψ) in the dataset $p(\psi|s)$, along with a readout policy $\pi(a|s,\psi)$ that takes an action a to realise ψ starting at state s.

4.2 Transfer with Generalized Occupancy Models

To synthesize optimal policies for novel downstream reward functions using GOMs, two sub-problems must be solved: (1) inferring the suitable linear reward weights w_r for a particular reward function from a set of (s, r) tuples and (2) using the inferred w_r to select an optimal action a^* at a state s. We discuss each below.

Inferring task-specific weights with linear regression. As noted, for any reward function $r(s)$, once the linear reward weights w_r are known (i.e., $r(s) = w_r^T \phi(s)$), the distribution of returns in the dataset $p(R|s)$ is known through linearity. However, in most cases, rewards are not provided in functional form, making w_r unknown a priori. Instead, given a dataset of $\mathcal{D} = \{(s, r)\}$ tuples, w_r can be obtained by solving a simple linear regression problem $\min \frac{1}{|\mathcal{D}|} \sum_{(s,r)\in\mathcal{D}} \|w_r^T \psi(s) - r\|_2^2$. We emphasize that this is all the optimization required for transferring GOMs to new tasks.

Generating task-specific policies via distributional evaluation. Given the inferred w_r and the corresponding future return distribution $p(R|s)$ obtained through linear scaling of $p(\psi|s)$, the optimal action can be obtained by finding the ψ corresponding to the highest possible future return that has sufficient data-support:

$$\psi^* \leftarrow \arg\max_{\psi} \qquad w_r^T \psi, \qquad \text{s.t} \quad p(\psi|s) \geq \epsilon, \tag{3}$$

where $\epsilon > 0$ is a tuning parameter to ensure sufficient coverage for ψ. This suggests that the optimal outcome ψ^* is the one that provides the highest future sum of rewards $w_r^T \psi^*$ that is valid under the environment dynamics and dataset coverage.

This optimization problem can be solved in a number of ways. Perhaps the most straightforward is via a random shooting technique [52], which samples a set of ψ from $p(\psi|s)$ and chooses the one with the highest $w_r^T \psi$. Sec. 5 bases our theoretical analysis on this planning method. Sec. 4.3 shows that for outcome models instantiated with specific model classes such as diffusion models, the planning problem can be simplified to guided sampling.

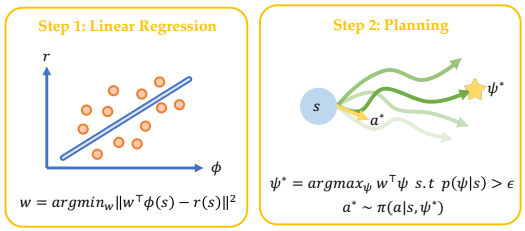

Figure 3: Planning with GOMs. Once a GOM is learned, the optimal action can be obtained by performing reward regression and searching for the optimal outcome under the dynamics to decode via the policy.

Once ψ^* has been obtained, the action to execute in the environment can be acquired via the readout policy $\pi_\rho(a|s, \psi^*)$. Fig. 3 shows the full the planning procedure, and we refer the reader to Appendix. F for the pseudocode. As described previously, GOMs enable rapid transfer to arbitrary new rewards in an environment without accumulating compounding error or requiring expensive test-time policy optimization. In this way, they can be considered a new class of models of transition dynamics that avoids the typical challenges in model-based RL and successor features.

4.3 Practical Instantiation

In this section, we provide a practical instantiation of GOMs that is used throughout our experimental evaluation. The first step to instantiate GOMs is to choose an expressive state feature that linearly expresses a broad class of rewards. We choose the state feature ϕ to be d-dimensional random Fourier features [51]. Next, the model class must account for the multimodal nature of outcomes and actions since multiple outcomes can follow from a state, and multiple actions can be taken to realize an outcome. To this end, we parametrize both the outcome model $p(\psi|s)$ and the readout policy $\pi(a|s, \psi)$ using a conditional diffusion model [22, 46]. We then train these models (optimize Equation 1, 2) by denoising score matching, a surrogate of maximum likelihood training [47].

Remarkably, when $p(\psi|s)$ is parameterized by a diffusion model, the special structure of the planning problem in Eq. 3 allows a simple variant of guided diffusion [13, 24] to be used for task-directed planning. In particular, taking the log of both sides of the constraint and recasting the constrained optimization via the penalty method, we get a penalized objective $\mathcal{L} = w_r^T \psi + \alpha(\log p(\psi|s) - \log \epsilon)$. Taking the gradient yields $\nabla_\psi \mathcal{L}(\psi, \alpha) = w_r + \alpha \nabla_\psi \log p(\psi|s)$. The expression for $\nabla_\psi \mathcal{L}(\psi, \alpha)$ is simply the score function $\nabla_\psi \log p(\psi|s)$ in standard diffusion training (Section 3), with the linear weights w_r added as a guidance term. Planning then becomes doing stochastic gradient Langevin dynamics [56] to obtain an optimal ψ^* sample, using $\nabla_\psi \mathcal{L}(\psi, \alpha)$ as the gradient. Guided diffusion removes the need for sampling a set of particles. As we show in Appendix E, it matches the performance of random shooting while taking significantly less inference time. In Appendix B we show that the guided diffusion procedure can alternatively be viewed as taking actions conditioned on a soft optimality variable.

5 Theoretical Analysis of GOMs

To provide a theoretical understanding of GOMs, we conduct an error analysis to connect the error in estimating the ground truth $p_0(\psi \mid s)$ to the suboptimality of the GOM policy, and then study when the GOM policy becomes optimal. We start our analysis conditioning on $\epsilon > 0$ estimation error in the ground truth outcome distribution $p_0(\psi \mid s)$.

Condition 5.1. We say the learnt outcome distribution $\hat{p}$ is an ϵ-good approximation if $\forall\ s \in \mathcal{S}$, $\|\hat{p}(\psi \mid s) - p_0(\psi \mid s)\|_\infty \leq \epsilon$.

Since GOMs capture the outcome distribution of the behavior policy π_β, we need a definition to evaluate a policy π with respect to π_β.

Definition 5.2. We say a state-action pair (s, a) is (δ, π_β)-good if over the randomness of π_β, $\mathbb{P}_{\pi_\beta}[Q^{\pi_\beta}(s, a) < \sum_{t=1}^{\infty} \gamma^{t-1} r(s_t) \mid s_1 = s] \leq \delta$. Furthermore, if for all state s, $(s, \pi(s))$ is (δ, π_β)-good, then we call π a (δ, π_β)-good policy.

We proceed to use Definition 5.2 to characterize the suboptimality of the GOM policy. Let τ denote the sampling optimality of the random shooting planner in Sec. 4.2. Specifically, we expect to sample a top τ outcome ψ from the behavior policy in $O(\frac{1}{\tau})$ samples, where $\mathbb{P}_{\pi_\beta}[w_r^T \psi \leq \sum_t \gamma^{t-1} r(s_t)] \leq \tau$. The following result characterizes the suboptimality of the GOM policy. The proof is deferred to Appendix. A.

Theorem 5.3 (main theorem)**.** *For any MDP $\mathcal{M}$ and ϵ-good outcome distribution $\hat{p}$, the policy $\widehat{\pi}$ given by the random shooting planner with sampling optimality τ is a $(\epsilon + \tau, \pi_\beta)$-good policy.*

From Theorem 5.3, we can obtain the following suboptimality guarantee in terms of the value function under the Lipschitzness condition. The corollary shows the estimation error in $p_0(\psi \mid s)$ will be amplified by an $O\left(\frac{1}{1-\gamma}\right)$ multiplicative factor.

Corollary 5.4. *If we have λ-Lipschitzness near the optimal policy, i.e., $Q^*(s, a^*) - Q^*(s, a) \leq \lambda\delta$ when (s, a) is (δ, β)-good, the suboptimality of output policy $\hat{\pi}$ is $V_0^*(s_0) - V_0^{\hat{\pi}}(s_0) \leq \frac{\lambda}{1-\gamma}(\tau + \epsilon)$.*

Lastly, we extend our main theoretical result to the standard full data coverage condition in the offline RL literature, where the dataset contains all transitions [50, 59, 43]. The following theorem states that GOMs can output the optimal policy in this case. The proof is deferred to Appendix A.

Theorem 5.5. *In deterministic MDPs, when $|\mathcal{A} \times \mathcal{S}| < \infty$, and $\forall (s, a) \in \mathcal{S} \times \mathcal{A}, N(s, a, \mathcal{T}(s, a)) \geq 1$, GOM is guaranteed to identify an optimal policy.*

6 Experimental Evaluation

In our experimental evaluation, we aim to answer the following research questions. (1) Can GOMs transfer across tasks without expensive test-time policy optimization? (2) Can GOMs avoid the challenge of compounding error present in MBRL? (3) Can GOMs solve tasks with arbitrary rewards beyond goal-reaching problems? (4) Can GOMs go beyond the offline dataset, and accomplish "trajectory-stitching" to actualize outcomes that combine different trajectories?

We answer these questions through a number of experimental results in simulated robotics problems. We defer detailed descriptions of domains and baselines to Appendix D and C, as well as detailed ablative analysis to Appendix E.

6.1 Problem Domains and Datasets

D4RL Antmaze D4RL Antmaze [14] is a navigation domain that involves controlling a quadruped to reach some designated goal location. Each task corresponds to reaching a different goal location. We use the D4RL dataset for pretraining and dense rewards described in Appendix D for adaptation.

Franka Kitchen Franka Kitchen [14] is a manipulation domain where the goal is to control a Franka arm to interact with appliances in the kitchen. Each task corresponds to interacting with a set of items. We use the D4RL dataset for pretraining and standard sparse rewards for adaptation.

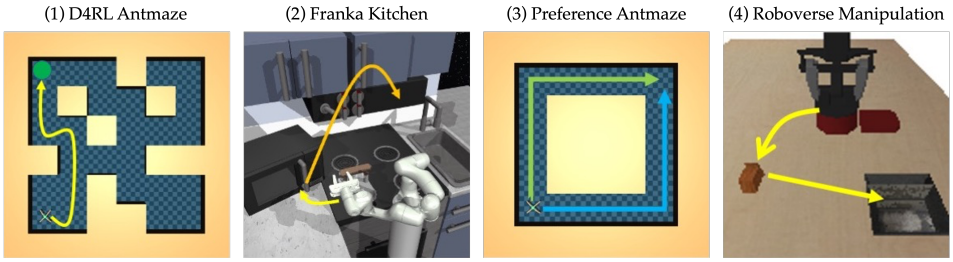

Figure 4: Evaluation domains: (1) D4RL Antmaze [14] (2) Franka Kitchen [14] (3) Preference-Based Antmaze with the goal of taking a particular path (4) Roboverse [45] robotic manipulation.

Table 1: Offline multitask RL on AntMaze and Kitchen. GOMs show superior transfer performance (in average episodic return) than successor features, model-based RL, and misspecified goal-conditioned baselines.

	GOM (Ours)	USF	FB	RaMP	MOPO	COMBO	GC-IQL
umaze	**593** ± 16	462 ± 4	469 ± 12	459 ± 3	451 ± 2	574 ± 10	571 ± 15
umaze-diverse	**568** ± 12	447 ± 3	474 ± 2	460 ± 7	467 ± 5	547 ± 11	577 ± 7
medium-diverse	**631** ± 67	394 ± 52	294 ± 61	266 ± 2	236 ± 4	418 ± 16	403 ± 10
medium-play	**624** ± 58	370 ± 31	264 ± 29	271 ± 5	232 ± 4	397 ± 12	390 ± 33
large-diverse	**359** ± 59	215 ± 20	181 ± 46	132 ± 1	128 ± 1	244 ± 19	226 ± 9
large-play	**306** ± 18	250 ± 41	165 ± 12	134 ± 3	128 ± 2	248 ± 4	229 ± 5
kitchen-partial	**43** ± 6	0 ± 0	4 ± 4	0 ± 0	8 ± 7	11 ± 9	-
kitchen-mixed	**46** ± 5	10 ± 10	5 ± 5	0 ± 0	0 ± 0	0 ± 0	-

Preference Antmaze Preference Antmaze is a variant of D4RL Antmaze [14] where the goal is to reach the top right corner starting from the bottom left corner. The two tasks in this environment correspond to the two paths to reach the goal, simulating human preferences. We collect a custom dataset and design reward functions for each preference.

Roboverse Robotic Manipulation Roboverse [45] is a tabletop manipulation environment with a robotic arm completing multi-step problems. Each task consists of two phases, and the offline dataset contains separate trajectories of each phase but not full task completion. A sparse reward is assigned to each time step of task completion.

6.2 Baseline Comparisons

Successor Features We compare with three methods from the successor feature line of work. **Universal SF (USF)** [6] overcomes the policy dependence of SF by randomly sampling reward weights z and jointly learning a successor feature predictor ψ_z and a policy π_z conditioned on z. ψ_z captures the successor feature of π_z, while π_z is trained to maximize the reward described by z. **Forward-Backward Representation (FB)** [53, 54] follows the same paradigm but jointly learns a feature network by parameterizing the successor measure as an inner product between a forward and a backward representation. **RaMP** [8] removes the policy dependence of SF by predicting cumulative features from an initial state and an open-loop sequence of actions, which can be used for planning.

Model-Based RL We compare with two variants of model-based reinforcement learning. **MOPO** [63] is a model-based offline RL method that learns an ensemble of dynamics models and performs actor-critic learning. **COMBO** [62] introduces pessimism into MOPO by training the policy using a conservative objective [29].

Goal-Conditioned RL Goal-conditioned RL enables adaptation to multiple downstream goals g. However, it is solving a more restricted class of problems than RL as goals are less expressive than rewards in the same state space. Moreover, standard GCRL is typically trained on the same set of goals as in evaluation, granting them privileged information. To account for this, we consider a goal-conditioned RL baseline **GC-IQL** [39, 28] and only train on goals from half the state space to show its fragility to goal distributions. We include the original method trained on test-time goals in Appendix E.

Table 2: Evaluation on non-goal-conditioned tasks. GOMs are able to solve non-goal-conditioned tasks, taking different paths in preference antmaze (Fig 4), while goal-conditioned RL cannot optimize for arbitrary rewards.

	GOM (Ours)	COMBO	GC-IQL
Up	139 ± 1	143 ± 9	72 ± 19
Right	**142** ± 2	136 ± 4	83 ± 25

Table 3: Evaluation of trajectory stitching ability of GOMs. GOMs outperform non-stitching baselines, demonstrating their abilities to recombine outcomes across trajectory segments

	GOM (Ours)	RaMP	DT
PickPlace	**49** ± 8	0 ± 0	0 ± 0
ClosedDrawer	**40** ± 5	0 ± 0	0 ± 0
BlockedDrawer	**66** ± 7	0 ± 0	0 ± 0

6.3 Do GOMs learn transferable behavior across tasks?

We evaluate GOMs on transfer problems, where the dynamics are shared, but the reward functions vary. We train GOMs using the data distributions provided with the standard D4RL datasets [14]. While GOMs in principle can identify the task reward from a small number of state-reward pairs, we relabel the offline dataset with test-time rewards to remove the confounding factor of exploration.

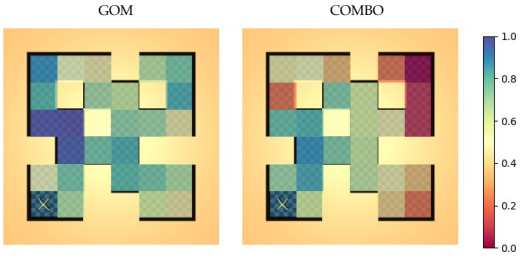

Figure 5: Transfer across tasks with GOMs and COMBO [62] in medium antmaze. Each tile corresponds to a different task, with color of the tile indicating the normalized return. GOMs successfully transfer across a majority of tasks, while MBRL [62] struggles on tasks that are further away from the initial location.

Table 1 reports the episodic return of transferring to the default D4RL tasks. GOMs are able to adapt to new tasks with a simple linear regression, showing significantly higher transfer performance than successor features (mismatch between training and evaluation policy sets), model-based RL (compounding error) and goal-conditioned RL (goal distribution misspecification). Notably, we show in Appendix E that GOMs are even competitive with goal-conditioned RL methods trained on test-time goals. The transferability of GOMs can also be seen in Fig 5, where we plot the performance of GOMs across various tasks (corresponding to different tiles in the maze). We see that GOMs have less degradation across tasks than model-based RL [62].

Although the GOM framework and theoretical results are derived under deterministic MDPs, we emphasize that the D4RL antmaze datasets are collected with action noise, emulating stochastic transitions. These results indicate that GOM are practically applicable to some range of stochastic settings, although we expect it to perform better in purely deterministic settings.

6.4 Can GOMs solve tasks with *arbitrary* rewards?

While methods like goal-conditioned RL [39, 16] are restricted to shortest path goal-reaching problems, GOMs are able to solve problems with *arbitrary* reward functions. This is crucial when the reward is not easily reduced to a particular "goal". To validate this, we evaluate GOMs on tasks that encode nontrivial human preferences in a reward function, such as particular path preferences in antmaze. In this case, we have different rewards that guide the agent specifically down the path to the left and the right, as shown in Fig 4. As we see in Table 2, GOMs and model-based RL obtain policies that respect human preferences and are performant for various rewards. Goal-conditioned algorithms are unable to disambiguate preferences and end up with some probability of taking each path.

6.5 Do GOMs perform trajectory stitching?

The ability to recover optimal behavior by combining suboptimal trajectories, or "trajectory stitching," is crucial to off-policy RL methods as it ensures data efficiency and avoids requirements for exponential data coverage. GOMs naturally enables this type of trajectory stitching via the distributional Bellman backup, recovering "best-in-data" policies for downstream tasks. To evaluate the ability of GOMs to perform trajectory stitching, we consider the environments introduced in [45]. Here, the data only consists of trajectories that complete individual subtasks (e.g. grasping or placing), while the task of interest rewards the completion of both subtasks. Since the goal of this experiment is to

evaluate stitching, not transfer, we choose the features as the task rewards $\phi(s) = r(s)$. We find that GOMs are able to show non-trivial success rates by stitching together subtrajectories. Since RaMP [8] predicts the summed features from a sequence of actions, and the optimal action sequence is not present in the dataset, it fails to solve any task. Likewise, return-conditioned supervised learning methods like Decision Transformer [9] do not stitch together trajectories and fails to learn meaningful behaviors.

7 Discussion

This work introduced generalized occupancy models, a method for transferable reinforcement learning that does not incur compounding error or test-time policy optimization. GOMs can quickly adapt to provide optimal policies for *any* reward by modeling the distribution of *all* possible future outcomes along with policies to reach them. We presented an efficient algorithm to learn GOMs and demonstrated the benefits of GOMs over standard successor features and model-based RL techniques. The limitations of our work open future research opportunities. First, GOMs require a choice of features $\phi(s)$ that linearly express the rewards; this assumption may fail, necessitating more expressive feature learning methods. Second, GOMs model the behavior distribution of the dataset; hence, policy optimality can be affected by dataset skewness, which motivates the use of more efficient exploration methods for data collection. Finally, the current version of GOMs infer the reward from offline state-reward pairs; a potential future direction could apply this paradigm to online adaptation, where the reward is inferred from online interactions.

References

[1] K. Asadi, D. Misra, and M. L. Littman. Lipschitz continuity in model-based reinforcement learning. In J. G. Dy and A. Krause, editors, *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pages 264–273. PMLR, 2018.

[2] A. Barreto, D. Borsa, J. Quan, T. Schaul, D. Silver, M. Hessel, D. Mankowitz, A. Zidek, and R. Munos. Transfer in deep reinforcement learning using successor features and generalised policy improvement. In *International Conference on Machine Learning*, pages 501–510. PMLR, 2018.

[3] A. Barreto, W. Dabney, R. Munos, J. J. Hunt, T. Schaul, H. P. van Hasselt, and D. Silver. Successor features for transfer in reinforcement learning. *Advances in Neural Information Processing Systems*, 30, 2017.

[4] M. G. Bellemare, W. Dabney, and R. Munos. A distributional perspective on reinforcement learning. In D. Precup and Y. W. Teh, editors, *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, volume 70 of *Proceedings of Machine Learning Research*, pages 449–458. PMLR, 2017.

[5] D. Borsa, A. Barreto, J. Quan, D. J. Mankowitz, H. van Hasselt, R. Munos, D. Silver, and T. Schaul. Universal successor features approximators. In *International Conference on Learning Representations*, 2019.

[6] D. Borsa, A. Barreto, J. Quan, D. J. Mankowitz, H. van Hasselt, R. Munos, D. Silver, and T. Schaul. Universal successor features approximators. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019.

[7] S. Casper, X. Davies, C. Shi, T. K. Gilbert, J. Scheurer, J. Rando, R. Freedman, T. Korbak, D. Lindner, P. Freire, T. Wang, S. Marks, C. Ségerie, M. Carroll, A. Peng, P. J. K. Christoffersen, M. Damani, S. Slocum, U. Anwar, A. Siththaranjan, M. Nadeau, E. J. Michaud, J. Pfau, D. Krasheninnikov, X. Chen, L. Langosco, P. Hase, E. Biyik, A. D. Dragan, D. Krueger, D. Sadigh, and D. Hadfield-Menell. Open problems and fundamental limitations of reinforcement learning from human feedback. *CoRR*, abs/2307.15217, 2023.

[8] B. Chen, C. Zhu, P. Agrawal, K. Zhang, and A. Gupta. Self-supervised reinforcement learning that transfers using random features. *CoRR*, abs/2305.17250, 2023.

[9] L. Chen, K. Lu, A. Rajeswaran, K. Lee, A. Grover, M. Laskin, P. Abbeel, A. Srinivas, and I. Mordatch. Decision transformer: Reinforcement learning via sequence modeling. In M. Ranzato, A. Beygelzimer, Y. N. Dauphin, P. Liang, and J. W. Vaughan, editors, *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages 15084–15097, 2021.

[10] K. Chua, R. Calandra, R. McAllister, and S. Levine. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. *CoRR*, abs/1805.12114, 2018.

[11] M. P. Deisenroth and C. E. Rasmussen. PILCO: A model-based and data-efficient approach to policy search. In L. Getoor and T. Scheffer, editors, *Proceedings of the 28th International Conference on Machine Learning, ICML 2011, Bellevue, Washington, USA, June 28 - July 2, 2011*, pages 465–472. Omnipress, 2011.

[12] P. Dhariwal and A. Q. Nichol. Diffusion models beat GANs on image synthesis. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, editors, *Advances in Neural Information Processing Systems*, 2021.

[13] P. Dhariwal and A. Q. Nichol. Diffusion models beat gans on image synthesis. In M. Ranzato, A. Beygelzimer, Y. N. Dauphin, P. Liang, and J. W. Vaughan, editors, *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages 8780–8794, 2021.

[14] J. Fu, A. Kumar, O. Nachum, G. Tucker, and S. Levine. D4RL: Datasets for deep data-driven reinforcement learning. *https://arxiv.org/abs/2004.07219*, 2020.

[15] J. Fu, A. Kumar, O. Nachum, G. Tucker, and S. Levine. D4RL: datasets for deep data-driven reinforcement learning. *CoRR*, abs/2004.07219, 2020.

[16] D. Ghosh, A. Gupta, A. Reddy, J. Fu, C. M. Devin, B. Eysenbach, and S. Levine. Learning to reach goals via iterated supervised learning. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021.

[17] A. Gupta, J. Yu, T. Z. Zhao, V. Kumar, A. Rovinsky, K. Xu, T. Devlin, and S. Levine. Reset-free reinforcement learning via multi-task learning: Learning dexterous manipulation behaviors without human intervention. *arXiv preprint arXiv:2104.11203*, 2021.

[18] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. *arXiv preprint arXiv:1801.01290*, 2018.

[19] D. Hafner, T. P. Lillicrap, J. Ba, and M. Norouzi. Dream to control: Learning behaviors by latent imagination. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020.

[20] D. Hafner, T. P. Lillicrap, M. Norouzi, and J. Ba. Mastering atari with discrete world models. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021.

[21] D. Han, B. Mulyana, V. Stankovic, and S. Cheng. A survey on deep reinforcement learning algorithms for robotic manipulation. *Sensors*, 23(7):3762, 2023.

[22] J. Ho, A. Jain, and P. Abbeel. Denoising diffusion probabilistic models, 2020.

[23] R. A. Howard. Dynamic programming and markov processes. *John Wiley*, 1960.

[24] M. Janner, Y. Du, J. B. Tenenbaum, and S. Levine. Planning with diffusion for flexible behavior synthesis. In K. Chaudhuri, S. Jegelka, L. Song, C. Szepesvári, G. Niu, and S. Sabato, editors, *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*, volume 162 of *Proceedings of Machine Learning Research*, pages 9902–9915. PMLR, 2022.

[25] M. Janner, J. Fu, M. Zhang, and S. Levine. When to trust your model: Model-based policy optimization. In H. M. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. B. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 12498–12509, 2019.

[26] M. Janner, I. Mordatch, and S. Levine. Gamma-models: Generative temporal difference learning for infinite-horizon prediction. In H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

[27] J. Kober, J. A. Bagnell, and J. Peters. Reinforcement learning in robotics: A survey. *Int. J. Robotics Res.*, 32(11):1238–1274, 2013.

[28] I. Kostrikov, A. Nair, and S. Levine. Offline reinforcement learning with implicit q-learning. In *International Conference on Learning Representations*, 2022.

[29] A. Kumar, A. Zhou, G. Tucker, and S. Levine. Conservative q-learning for offline reinforcement learning. *CoRR*, abs/2006.04779, 2020.

[30] N. O. Lambert, K. S. J. Pister, and R. Calandra. Investigating compounding prediction errors in learned dynamics models. *CoRR*, abs/2203.09637, 2022.

[31] N. O. Lambert, A. Wilcox, H. Zhang, K. S. J. Pister, and R. Calandra. Learning accurate long-term dynamics for model-based reinforcement learning. In *2021 60th IEEE Conference on Decision and Control (CDC), Austin, TX, USA, December 14-17, 2021*, pages 2880–2887. IEEE, 2021.

[32] S. Levine. Reinforcement learning and control as probabilistic inference: Tutorial and review. *CoRR*, abs/1805.00909, 2018.

[33] S. Levine, A. Kumar, G. Tucker, and J. Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *CoRR*, abs/2005.01643, 2020.

[34] I. Loshchilov and F. Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019.

[35] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. P. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In M. Balcan and K. Q. Weinberger, editors, *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, volume 48 of *JMLR Workshop and Conference Proceedings*, pages 1928–1937. JMLR.org, 2016.

[36] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. A. Riedmiller. Playing atari with deep reinforcement learning. *CoRR*, abs/1312.5602, 2013.

[37] A. Nagabandi, G. Kahn, R. S. Fearing, and S. Levine. Neural network dynamics for model-based deep reinforcement learning with model-free fine-tuning. In *2018 IEEE International Conference on Robotics and Automation, ICRA 2018, Brisbane, Australia, May 21-25, 2018*, pages 7559–7566. IEEE, 2018.

[38] A. Nagabandi, K. Konolige, S. Levine, and V. Kumar. Deep dynamics models for learning dexterous manipulation. In L. P. Kaelbling, D. Kragic, and K. Sugiura, editors, *3rd Annual Conference on Robot Learning, CoRL 2019, Osaka, Japan, October 30 - November 1, 2019, Proceedings*, volume 100 of *Proceedings of Machine Learning Research*, pages 1101–1112. PMLR, 2019.

[39] S. Park, D. Ghosh, B. Eysenbach, and S. Levine. Hiql: Offline goal-conditioned rl with latent states as actions. *Advances in Neural Information Processing Systems*, 2023.

[40] E. Perez, F. Strub, H. de Vries, V. Dumoulin, and A. C. Courville. Film: Visual reasoning with a general conditioning layer. In *AAAI*, 2018.

[41] A. Rahimi and B. Recht. Random features for large-scale kernel machines. *Advances in Neural Information Processing Systems*, 20, 2007.

[42] A. Rahimi and B. Recht. Weighted sums of random kitchen sinks: Replacing minimization with randomization in learning. *Advances in Neural Information Processing Systems*, 21, 2008.

[43] T. Ren, J. Li, B. Dai, S. S. Du, and S. Sanghavi. Nearly horizon-free offline reinforcement learning. *Advances in neural information processing systems*, 34:15621–15634, 2021.

[44] O. Rybkin, C. Zhu, A. Nagabandi, K. Daniilidis, I. Mordatch, and S. Levine. Model-based reinforcement learning via latent-space collocation. In M. Meila and T. Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pages 9190–9201. PMLR, 2021.

[45] A. Singh, A. Yu, J. Yang, J. Zhang, A. Kumar, and S. Levine. Cog: Connecting new skills to past experience with offline reinforcement learning. *Preprint arXiv:2010.14500*, 2020.

[46] J. Song, C. Meng, and S. Ermon. Denoising diffusion implicit models, 2022.

[47] Y. Song, C. Durkan, I. Murray, and S. Ermon. Maximum likelihood training of score-based diffusion models. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, editors, *Advances in Neural Information Processing Systems*, 2021.

[48] Y. Song, J. Sohl-Dickstein, D. P. Kingma, A. Kumar, S. Ermon, and B. Poole. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations*, 2021.

[49] R. S. Sutton. Dyna, an integrated architecture for learning, planning, and reacting. *SIGART Bull.*, 2(4):160–163, 1991.

[50] C. Szepesvári and R. Munos. Finite time bounds for sampling based fitted value iteration. In *Proceedings of the 22nd international conference on Machine learning*, pages 880–887, 2005.

[51] M. Tancik, P. P. Srinivasan, B. Mildenhall, S. Fridovich-Keil, N. Raghavan, U. Singhal, R. Ramamoorthi, J. T. Barron, and R. Ng. Fourier features let networks learn high frequency functions in low dimensional domains. *NeurIPS*, 2020.

[52] R. Tedrake. *Underactuated Robotics*. 2023.

[53] A. Touati and Y. Ollivier. Learning one representation to optimize all rewards. *Advances in Neural Information Processing Systems*, 34:13–23, 2021.

[54] A. Touati, J. Rapin, and Y. Ollivier. Does zero-shot reinforcement learning exist? In *The Eleventh International Conference on Learning Representations*, 2023.

[55] A. Wagenmaker, G. Shi, and K. Jamieson. Optimal exploration for model-based RL in nonlinear systems. *CoRR*, abs/2306.09210, 2023.

[56] M. Welling and Y. W. Teh. Bayesian learning via stochastic gradient langevin dynamics. In L. Getoor and T. Scheffer, editors, *Proceedings of the 28th International Conference on Machine Learning, ICML 2011, Bellevue, Washington, USA, June 28 - July 2, 2011*, pages 681–688. Omnipress, 2011.

[57] G. Williams, N. Wagener, B. Goldfain, P. Drews, J. M. Rehg, B. Boots, and E. A. Theodorou. Information theoretic MPC for model-based reinforcement learning. In *2017 IEEE International Conference on Robotics and Automation, ICRA 2017, Singapore, Singapore, May 29 - June 3, 2017*, pages 1714–1721. IEEE, 2017.

[58] H. Wiltzer, J. Farebrother, A. Gretton, Y. Tang, A. Barreto, W. Dabney, M. G. Bellemare, and M. Rowland. A distributional analogue to the successor representation, 2024.

[59] T. Xie and N. Jiang. Batch value-function approximation with only realizability. In *International Conference on Machine Learning*, pages 11404–11413. PMLR, 2021.

[60] Y. Xu, J. Parker-Holder, A. Pacchiano, P. J. Ball, O. Rybkin, S. Roberts, T. Rocktäschel, and E. Grefenstette. Learning general world models in a handful of reward-free deployments. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 2022.

[61] K. J. Young, A. Ramesh, L. Kirsch, and J. Schmidhuber. The benefits of model-based generalization in reinforcement learning. In A. Krause, E. Brunskill, K. Cho, B. Engelhardt, S. Sabato, and J. Scarlett, editors, *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pages 40254–40276. PMLR, 2023.

[62] T. Yu, A. Kumar, R. Rafailov, A. Rajeswaran, S. Levine, and C. Finn. Combo: Conservative offline model-based policy optimization. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 28954–28967. Curran Associates, Inc., 2021.

[63] T. Yu, G. Thomas, L. Yu, S. Ermon, J. Zou, S. Levine, C. Finn, and T. Ma. Mopo: Model-based offline policy optimization. *Preprint arXiv:2005.13239*, 2020.

[64] M. R. Zhang, T. Paine, O. Nachum, C. Paduraru, G. Tucker, Z. Wang, and M. Norouzi. Autoregressive dynamics models for offline policy evaluation and optimization. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021.

[65] T. Zhang, T. Ren, M. Yang, J. Gonzalez, D. Schuurmans, and B. Dai. Making linear mdps practical via contrastive representation learning. In K. Chaudhuri, S. Jegelka, L. Song, C. Szepesvári, G. Niu, and S. Sabato, editors, *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*, volume 162 of *Proceedings of Machine Learning Research*, pages 26447–26466. PMLR, 2022.

Supplementary Materials for
"Transferable Reinforcement Learning via Generalized Occupancy Models"

A Missing Proofs

We provide the complete proofs of theorems and corollaries stated in Sec. 5. Throughout the following sections, we use $1_{\mathcal{E}}$ to denote the indicator of event $\mathcal{E}$.

A.1 Formal Statement and Proof of Theorem 5.3

To state Theorem 5.3 rigorously, we introduce the basic setting here. Without loss of generality, let the feature at time step i, $\phi(s_i) \in [0, 1-\gamma]^d$ and the outcome $\psi = \sum_{i=0}^{\infty} \gamma^i \phi(s_i) \in [0,1]^d$. Moreover, we have a readout policy π s.t. $\hat{a} \sim \hat{\pi}(s, \psi)$ always leads to a successor state s' s.t. $\hat{p}(\frac{1}{\gamma}(\psi - \phi(s)) \mid s') > 0$.

We simplify the planning phase of GOM into the following form: for a given reward weight w_r, in each time step, we have

1. Infer optimal outcome $\psi^* = \mathrm{A}(s, \hat{p})$ through random shooting;
2. Get corresponding action from $\hat{\pi}$, $\hat{a} \sim \hat{\pi}(s, \psi^*)$.

where the random shooting oracle A with sampling optimality τ satisfies

$$w_r^T \mathrm{A}(s, \hat{p}) \geq \min\{R \mid \int 1_{[w_r^T \psi \geq R]} \hat{p}[\psi \mid s] d\psi \leq \tau\}.$$

This intuitively means that there are at most probability $\tau \in [0,1]$ of the behavior policy achieving higher reward. We proceed to prove Theorem 5.3.

Proof. It suffices to prove that $\forall s, (s, \hat{\pi}(s))$ is at least $(\tau + \epsilon, \pi_\beta)$-good. For simplicity, we denote $\widehat{R} := \min\{R \mid \int 1_{[w_r^T \psi \geq R]} \hat{p}[\psi \mid s] d\psi \leq \tau\}$. Then

$$\begin{aligned}
&\mathbb{P}_{\pi^\beta}[w_r^T \sum_{t=1}^{\infty} \gamma^{t-1} \phi(s_t) \geq Q^{\pi_\beta}(s, \hat{\pi}(s))] \\
&= \mathbb{P}_{\psi \sim p_0(\cdot \mid s)}[w_r^T \psi \geq w_r^T \psi^*] \\
&= \int 1_{[w_r^T \psi \geq w_r^T \psi^*)]} p_0(\psi \mid s) d\psi \\
&\leq \int 1_{[w_r^T \psi \geq \widehat{R}]} p_0(\psi \mid s) d\psi \\
&= \int 1_{[w_r^T \psi \geq \widehat{R}]} \hat{p}(\psi \mid s) d\psi \\
&\qquad + \int 1_{[w_r^T \psi \geq \widehat{R}]} \left[p_0(\psi \mid s) - \hat{p}(\psi \mid s)\right] d\psi \\
&\leq \tau + \int 1_{[w_r^T \psi \geq \widehat{R}]} \epsilon d\psi \\
&\leq \tau + \epsilon.
\end{aligned}$$

□

A.2 Proof of Corollary 5.4

Proof. Intuitively, policy $\hat{\pi}$ fall behind by at most $\lambda(\tau+\epsilon)$ at each time step: $\forall s_1 \in S$,

$$\begin{aligned}
&V^*(s_1) - V^{\hat{\pi}}(s_1) \\
&= Q^*(s_1, a^*) - Q^{\hat{\pi}}(s_1, \hat{a}) \\
&= Q^*(s_1, a^*) - Q^*(s_1, \hat{a}) \\
&\qquad + Q^*(s_1, \hat{a}) - Q^{\hat{\pi}}(s_1, \hat{a}) \\
&\leq \lambda(\tau+\epsilon) + Q^*(s_1, \hat{a}) - Q^{\hat{\pi}}(s_1, \hat{a}) \\
&= \lambda(\tau+\epsilon) + \gamma \mathbb{E}_{s_2 \sim p(s_1, \hat{a})}[V^*(s_2) - V^{\hat{\pi}}(s_2)] \\
&\leq \lambda(\tau+\epsilon) + \gamma\lambda(\tau+\epsilon) \\
&\qquad + \gamma^2 \mathbb{E}_{s_3}[V^*(s_3) - V^{\hat{\pi}}(s_3)] \\
&\leq \cdots \\
&\leq \sum_{i=0}^{\infty} \gamma(\tau+\epsilon) \\
&= \frac{\lambda}{1-\gamma}(\tau+\epsilon).
\end{aligned}$$

□

A.3 Proof of Theorem 5.5

Proof. Under the full coverage condition, the sampling optimality τ can be set to be zero for concrete actions, and the condition $p(\psi|s) > \epsilon$ becomes $p(\psi|s) > 0$. Then we see that planning only focuses on the supporting set of $\hat{p}$: $\hat{p}(\psi \mid s) = 0$ if and only $(s, \psi) \in \mathcal{D}$ for some time during execution, which equals $p_0(\psi \mid s) = 0$. This indicates that $\mathrm{supp}(\hat{p}) = \mathrm{supp}(p_0)$. Therefore the conclusion follows. □

B Alternative Derivation of Guided Diffusion Planner

In this section, we derive the guided diffusion planner from the perspective of control as inference [32]. To start, we define the trajectory-level optimality variable $\mathcal{O}$ as a Bernoulli variable taking the value of 1 with probability $\exp(R(\tau))$ and 0 otherwise, where $R(\tau) = \sum_{t=0}^{T} \gamma^t r(s_t) - R_{\text{max}}$. Note we subtract the max discounted return R_{max} to make the density a valid probability distribution. Planning can be cast as an inference problem where the goal is to sample $\psi^* \sim p(\psi|O)$. By Bayes rule, we have

$$p(\psi|O) \propto p(O|\psi)p(\psi)$$

Taking the gradient of the log of both sides, we get

$$\begin{aligned}
\nabla_\psi \log p(\psi|O) &= \nabla_\psi \log p(O|\psi) + \nabla_\psi \log p(\psi) \\
&= \nabla_\psi \log \exp(w^\top \psi) + \nabla_\psi \log p(\psi) \\
&= \nabla_\psi w^\top \psi + \nabla_\psi \log p(\psi) \\
&= w + \nabla_\psi \log p(\psi)
\end{aligned}$$

This implies that we can sample from $p(\psi|O)$ by adding the regression weights w to the score at each timestep, yielding the same guided diffusion form as in Sec. 4.2.

C Implementation Details

C.1 Model architecture

We parameterize the random Fourier features using a randomly initialized 2-layer MLP with 2048 units in each hidden layer, followed by sine and cosine activations. For a d-dimensional feature,

the network's output dimension is $\lfloor d/2 \rfloor$ and the final feature is a concatenation of sine and cosine activated outputs. We set $d = 128$ for all of our experiments.

We implement the outcome model and policy using conditional DDIMs [46]. The noise prediction network is implemented as a 1-D Unet with down dimensions $[256, 512, 1024]$. Each layer is modulated using FiLM [40] to support conditioning.

C.2 Training details

For each environment, we train our models on the offline dataset for 100,000 gradient steps using the AdamW optimizer [34] with batch size 2048. The learning rate for the outcome model and the policy are set to $3e^{-4}$ and adjusted according to a cosine learning rate schedule with 500 warmup steps. We train the diffusion noise predictor with 1000 diffusion timesteps. To generate samples from a distribution, we maintain an exponential moving average model with decay rate 0.995 and employ the DDIM sampler with 50 timesteps [46] on the EMA models. For reward identification, we perform stochastic gradient descent on the regression weights w with learning rate $3e^{-4}$. **The same set of training hyperparameters is shared across all evaluation domains.**

We use the guided diffusion planner for transfer. The guidance coefficient α is 0.5 for antmaze, 0.01 for Franka kitchen, and 0.05 for Roboverse. We found planning to be sensitive to the guidance coefficient. Hence, for new environments, we suggest using the random shooting planner to get a baseline performance and then tuning the guided diffusion coefficient to improve performance and speed up inference. We run each experiment with 4 random seeds and report the mean and standard deviation in the tables. Each experiment (pretraining + adaptation) takes 3 hours on a single Nvidia L40 GPU.

C.3 Baselines

Successor Features We use the implementation of Universal Successor Features [5] and Forward-Backward Representation [53, 54] from the code release of [54]. We choose random Fourier features for universal SF as it performs best across the evaluation suite. To ensure fairness of comparison, we set the feature dimension to be 128 for both Universal SF and FB. Both methods are pretrained for 1 million gradient steps and adapted using the same relabelled data as GOMs.

RaMP We adapt the original RaMP implementation [8] and convert it into an offline method. RaMP originally consists of an offline training and an online adaptation stage, where online adaptation alternates between data collection and linear regression. We instead transfer by relabelling the offline dataset with the test-time reward function, thus removing the exploration challenge. We use an MPC horizon of 15 for all experiments.

Model-based RL We use the original implementations of MOPO [63] and COMBO [62] in our evaluations. We pretrain only the trainsition model on the offline transition datasets. To transfer to downstream tasks, we freeze the transition model, train the reward model on state reward pairs, and optimize the policy using model-based rollouts. We set the model rollout length for both methods to 5 and the CQL coefficient to be 0.5 for COMBO.

Goal-conditioned RL We use the GC-IQL baseline from [39]. To remove the privileged information, we modify the sampling distribution to only sample from half of the goal space excluding the test-time goal location.

D Environment Details

Antmaze D4RL Antmaze [15] is a navigation domain that involves controlling an 8-DoF quadruped robot to reach some designated goal location in a maze. Each task corresponds to reaching a different goal location. We use the standard D4RL offline dataset for pretraining. For downstream task adaptation, we replace the standard sparse reward $\mathbb{1}(s = g)$ with a dense reward $\exp(-||s - g||_2^2/20)$ to mitigate the challenge of sparse reward in long-horizon problems.

Franka Kitchen Franka Kitchen [17] is a robotics domain where the goal is to control a Franka arm to interact with appliances in the kitchen. Each task corresponds to interacting with a set of items

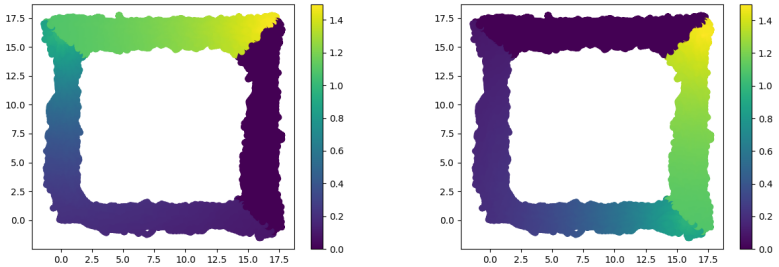

Figure 6: Data distribution and reward for Antmaze Preference environments. The left figure illustrates a preference for taking the vertical path, and the right figure illustrates the preference for taking the horizontal path.

Table 4: Full offline multitask RL on AntMaze and Kitchen. GOMs show superior transfer performance (in average episodic return) than successor features, model-based RL, and misspecified goal-conditioned baselines, while being competitive with an oracle using privileged information.

	GOM (Ours)	USF	FB	RaMP	MOPO	COMBO	GC-IQL	GC-Oracle
umaze	**593** ± 16	462 ± 4	469 ± 12	459 ± 3	451 ± 2	574 ± 10	571 ± 15	623 ± 7
umaze-diverse	**568** ± 12	447 ± 3	474 ± 2	460 ± 7	467 ± 5	547 ± 11	577 ± 7	576 ± 43
medium-diverse	**631** ± 67	394 ± 52	294 ± 61	266 ± 2	236 ± 4	418 ± 16	403 ± 10	659 ± 44
medium-play	**624** ± 58	370 ± 31	264 ± 29	271 ± 5	232 ± 4	397 ± 12	390 ± 33	673 ± 45
large-diverse	**359** ± 59	215 ± 20	181 ± 46	132 ± 1	128 ± 1	244 ± 19	226 ± 9	493 ± 9
large-play	**306** ± 18	250 ± 41	165 ± 12	134 ± 3	128 ± 2	248 ± 4	229 ± 5	533 ± 8
kitchen-partial	**43** ± 6	0 ± 0	4 ± 4	0 ± 0	8 ± 7	11 ± 9	-	33 ± 23
kitchen-mixed	**46** ± 5	10 ± 10	5 ± 5	0 ± 0	0 ± 0	0 ± 0	-	43 ± 7

in no particular order. We use the standard D4RL offline dataset for pretraining. For downstream task adaptation, we use the Markovian sparse rewards, where at each timestep the robot gets a reward equal to the number of completed tasks. We report the number of tasks completed throughout the entire episode in Table 1.

Preference Antmaze Preference Antmaze is a variant of D4RL Antmaze [14] where the goal is to reach the top right cell from the bottom left cell in a custom maze shown in Fig. 4. The two tasks in this environment are the two paths to reaching the goal, simulating different human preferences. To construct the dataset, we collect 1 million transitions using the D4RL waypoint controller. For each preference, we design a reward function that encourages the agent to take one path and not the other. Fig. 6 visualizes the dataset and the reward function for each preference.

Roboverse Robotic Manipulation Roboverse [45] is a tabletop manipulation environment consisting of a WidowX arm aiming to complete multi-step problems. Each task consists of two phases, and the offline dataset contains separate trajectories for each phasebut not full task completion. We use the standard sparse reward, assigning a reward of 1 for each timestep the task is completed.

E Additional Experiments

To understand the impact of various design decisions on the performance of GOMs, we conducted systematic ablations on the various components, using the antmaze-medium-diverse-v2 as a test bed.

E.1 Full D4RL Results

Table. 4 displays the full D4RL results with oracle goal-conditioned baseline labeled **GC-Oracle**. The baseline is trained on a goal distribution covering the testing-time goals, granting it privileged information. Despite this, GOMs are competitive with the oracle in most domains.

Table 5: Ablation of planning method. Wall time is measured over 1000 planning steps.

	Return ↑	Wall time (s) ↓
ATRL (Ours)	631 $\pm$ 67	42.9
Random shooting @ 1000	650 $\pm$ 50	94.8
Random shooting @ 100	619 $\pm$ 90	58.6
Random shooting @ 10	513 $\pm$ 52	55.5

Table 6: Ablation of feature dimension and type.

	Return ↑
ATRL (Ours)	631 $\pm$ 67
Random Fourier (64-dim)	561 $\pm$ 45
Random Fourier (32-dim)	295 $\pm$ 30
Random Fourier (16-dim)	307 $\pm$ 38
Random	382 $\pm$ 43
Forward dynamics	402 $\pm$ 36
Laplacian	376 $\pm$ 33

Table 7: Ablation of dataset coverage.

	Return ↑
Full dataset	631 $\pm$ 67
Random Subsampling	459 $\pm$ 57
Adversarial Subsampling	390 $\pm$ 26

E.2 Ablation of Planning Method

We compare the guided diffusion planner with the random shooting planner described in Sec. 4.2. As shown in Table 5, the guided diffusion planner achieves comparable performance to random shooting with 1000 particles while taking significantly less wall-clock time. While we can decrease the number of samples in the random shooting planner to improve planning speed, this comes at the cost of optimality.

E.3 Ablation of Feature Dimension and Type

To understand the importance of feature dimension and type, we compare variants of our method that use lower-dimensional random Fourier features, plain random features, and the two top-performing pretrained features from [54]. From Table. 6 we observe that as feature dimension decreases, their expressivity diminishes, resulting in lower performance. We found random features to perform much worse than random Fourier features. Interestingly, pretrained features with dynamics prediction and graph Laplacian objectives also achieve lower returns than random Fourier features. We hypothesize these pretrained features overfit to the training objective and are less expressive than random Fourier features

E.4 Ablation of Dataset Coverage

We investigate the effect of dataset coverage on the performance of our method. We compare GOMs trained on the full D4RL dataset against two variants, one where we randomly subsample half of the transitions, and the other where we adversarially remove the transitions from the half of the state space containing the test-time goal. As shown in Table 7, the performance of GOM drops as dataset coverage degrades.

E.5 Nonparametric Baseline

To confirm the intuition of our method, we implemented a nonparametric baseline that constructs an empirical estimate of the outcome distribution. First, we calculate the empirical discounted summ of features along trajectories in the dataset. Given the set of empirical (s, a, ψ) pairs and the downstream reward weight w, we can then select the optimal action at state s through the following steps: (1) query the k nearest neighbors of s, (2) evaluate their corresponding values $w^\top \psi$, (3) take the action of the top-valued neighbor.

Table 8: Comparison to a nonparametric baseline that takes the top-valued action among the k nearest neighbors of a state.

	Return ↑
GOM (Ours)	631 $\pm$ 67
Nonparametric $k = 10$	308 $\pm$ 20
Nonparametric $k = 100$	299 $\pm$ 21
Nonparametric $k = 1000$	287 $\pm$ 10

We found this baseline to perform surprisingly well on antmaze-medium-diverse-v2. While it does not achieve the performance of GOMs, it outperforms the RaMP and MOPO baselines. This result confirms the intuition behind the GOM design, which involves selecting the optimal outcome under dataset coverage and taking an action to realize it. We attribute the performance gap between this baseline and GOMs to their trajectory stitching ability (acquired via dynamic programming), infinite horizon modeling, and neural network generalization.

F Algorithm Pseudocode

Algorithm 1 GOM Training

Given transition dataset $\mathcal{D}$, feature function $\phi(\cdot)$
Initialize $p_\theta(\psi|s)$, $\pi_\rho(a|s,\psi)$.
while not converged **do**
  Draw B transition tuples $\{s_i, a_i, s'_i\}_{i=1}^B \sim \mathcal{D}$.
  Sample successor features for next states $\psi'_i \sim p_\theta(\psi|s'_i), i = 1 \dots N$.
  Construct target successor feature $\psi_i^{\text{targ}} = \phi(s) + \gamma * \psi'_i$.
  `// ` ψ `model learning`
  Update feature distribution: $\theta \leftarrow \arg\max_\theta \log p_\theta(\psi_i^{\text{targ}}|s_i)$.
  `// Policy extraction`
  Update policy: $\rho \leftarrow \arg\max_\rho \log \pi_\rho(a|s_i, \psi_i^{\text{targ}})$.
end while

Algorithm 2 GOM Offline Adaptation

Given transition dataset $\mathcal{D}$, feature function $\phi(\cdot)$, reward function $r(s)$.
Relabel offline dataset $\mathcal{D}$ with reward function.
Initialize regression weights w.
Fit w to $\mathcal{D}$ using linear regression $w = \arg\min_w \mathbb{E}_{\mathcal{D}}[\| w^\top \phi(s) - r(s) \|_2^2]$.

Algorithm 3 GOM Online Adaptation

Given $p_\theta(\psi|s), \pi_\rho(a|s,\psi)$, feature function $\phi(\cdot)$.
Prefill online buffer $\mathcal{D}_{\text{buf}}$ with random exploration policy.
Initialize regression weights w.
for time steps $1 \dots T$ do **do**
  Fit w using linear regression $w = \arg\min_w \mathbb{E}_{\mathcal{D}_{\text{buf}}}[\| w^\top \phi(s) - r(s) \|_2^2]$.
  Infer optimal $\psi^* = \arg\max_\psi w^\top \psi \quad \text{s.t.} \quad p_\theta(\psi|s) > \epsilon$.
  Sample optimal action $a^* \sim \pi(a|s, \psi^*)$.
  Execute action in the environment and add transition to buffer.
end for

Algorithm 4 GOM Inference (Random Shooting)

Given $p_\theta(\psi|s), \pi_\rho(a|s,\psi)$, regression weight w, current state s.
Sample N outcomes $\{\psi_i\}_{i=1}^N \sim p_\theta(\psi|s)$.
Compute corresponding values $\{v_i\}_{i=1}^N$, where $v_i = w^\top \psi_i$.
Take optimal cumulant $\psi^* = \psi_i$, where $i = \arg\max_i \{v_i\}_{i=1}^N$.
Sample optimal action $a^* \sim \pi(a|s,\psi^*)$.

Algorithm 5 GOM Inference (Guided diffusion)

Given diffusion model $p_\theta(\psi|s)$, $\pi_\rho(a|s,\psi)$, regression weight w, current state s, guidance coefficient β.
Initialize outcome ψ_1 from prior.
for diffusion timestep $t = 1...T$ **do**
    Compute noise at timestep $\epsilon = \epsilon_\theta(\psi_t, t, s)$.
    Update noise $\epsilon' = \epsilon - \beta\sqrt{1-\bar{\alpha}_t}w$.
    Sample next timestep action ψ_{t+1} using ϵ'.
end for
Sample optimal action $a^* \sim \pi_\rho(a|s,\psi_T)$.