Intention-Conditioned Flow Occupancy Models

Anonymous authors
Paper under double-blind review

Abstract

Large-scale pre-training has fundamentally changed how machine learning research is done today: large foundation models are trained once, and then can be used by anyone in the community (including those without data or compute resources to train a model from scratch) to adapt and fine-tune to specific tasks. Applying this same framework to reinforcement learning (RL) is appealing because it offers compelling avenues for addressing core challenges in RL, including sample efficiency and robustness. However, there remains a fundamental challenge to pre-train large models in the context of RL: actions have long-term dependencies, so training a foundation model that reasons across *time* is important. Recent advances in generative AI have provided new tools for modeling highly complex distributions. In this paper, we build a probabilistic model to predict which states an agent will visit in the temporally distant future (i.e., an occupancy measure) using flow matching. As large datasets are often constructed by many distinct users performing distinct tasks, we include in our model a latent variable capturing the user's intention. This intention increases the expressivity of our model and enables adaptation with generalized policy improvement. We call our proposed method **intention-conditioned flow occupancy models (InFOM)**. Comparing with alternative methods for pre-training, our experiments on 36 state-based and 4 image-based benchmark tasks demonstrate that the proposed method achieves $1.8\times$ median improvement in returns and increases success rates by 36%.

1 Introduction

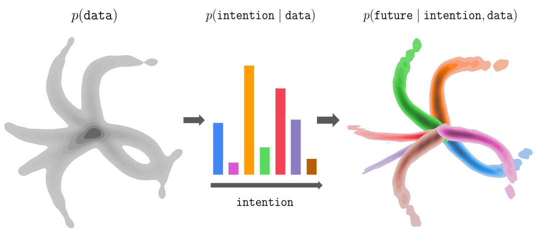

$p(\text{data})$
$p(\text{intention} \mid \text{data})$
$p(\text{future} \mid \text{intention}, \text{data})$
intention

Figure 1: **InFOM** is a latent variable model for pre-training and fine-tuning in reinforcement learning. *(Left)* The datasets are collected by users performing distinct tasks. *(Center)* We encode intentions by maximizing an evidence lower bound of data likelihood, *(Right)* enabling intention-aware future prediction using flow matching. See Sec. 4 for details.

Many of the recent celebrated successes of machine learning have been enabled by training large foundation models on vast datasets, and then adapting those models to downstream tasks. Examples include today's chatbots (e.g., Gemini (Team et al., 2023) and ChatGPT (Achiam et al., 2023)) and generalist robotic systems (e.g., π_0 (Black et al., 2024) and Octo (Team et al., 2024)). This pre-training-fine-tuning paradigm has been wildly successful in fields ranging from computer vision to natural language processing (Devlin et al., 2019; Brown et al., 2020; Touvron et al., 2023; Zhai et al., 2023; Radford et al., 2021; He et al., 2022; Ouyang et al., 2022; Lu et al., 2019), yet harnessing it in the context of reinforcement learning (RL) remains an open problem. What fundamentally makes the RL problem difficult is reasoning about time and intention—an effective RL agent must reason about the long-term effect of actions taken now, and must recognize that the data observed are often collected by distinct users performing multiple tasks. However, current attempts to build foundation models for RL often neglect these two important bits, focusing on predicting the actions in the pre-training dataset instead (Team et al., 2024; O'Neill et al., 2024; Walke et al., 2023).

The closest attempts to building RL algorithms that capture temporal bits are those based on world models (Ding et al., 2024; Hafner et al., 2023; Mendonca et al., 2021) and those based on occupancy models (Janner et al., 2020; Blier et al., 2021; Zheng et al., 2024; Farebrother et al., 2025).[1] World models can achieve great performance in sample efficiency (Janner et al., 2019) and generalize to diverse tasks (Hafner et al., 2023; Mendonca et al., 2021), although their capacity to perform long-horizon reasoning remains limited because of compounding errors (Talvitie, 2014; Janner et al., 2019; Lambert et al., 2022). Occupancy models (Dayan, 1993) and variants that enable scaling to high-dimensional tasks can also achieve great performance in predicting future events (Sikchi et al., 2024; Barreto et al., 2018; Zheng et al., 2024; 2025; Farebrother et al., 2025), but are typically hard to train and ignore user intentions. Recent advances in generative AI (e.g., flow-matching (Lipman et al., 2024; 2023; Liu et al., 2023) and diffusion (Ho et al., 2020; Song et al., 2021) models) enable modeling complex distributions taking various inputs, providing new tools for constructing occupancy models that depend on intentions.

In this paper, we propose a framework (Fig. 1) for pre-training in RL that simultaneously learns a probabilistic model to capture bits about time and intention. Building upon prior work on variational inference (Kingma & Welling, 2013; Alemi et al., 2017) and successor representations (Janner et al., 2020; Touati & Ollivier, 2021; Barreto et al., 2017; Zheng et al., 2024; Farebrother et al., 2025), we learn latent variable models of temporally distant future states, enabling intention-aware prediction. Building upon prior work on generative modeling, we use an expressive flow matching method (Farebrother et al., 2025) to train occupancy models, enabling highly flexible modeling of occupancy measures. We call the resulting algorithm **intention-conditioned flow occupancy models (InFOM)**. Experiments on 36 state-based and 4 image-based benchmark tasks show that InFOM outperforms alternative methods for pre-training and fine-tuning by $1.8\times$ median improvement in returns and 36% improvement in success rates. Additional experiments demonstrate that our latent variable model is capable of inferring underlying user intentions (Sec. 5.2) and enables efficient policy extraction (Sec. 5.3).

2 Related Work

Offline unsupervised RL. The goal of offline unsupervised RL is to pre-train policies, value functions, or models from an unlabeled (reward-free) dataset to enable efficient learning of downstream tasks. Prior work has proposed diverse offline unsupervised RL approaches based on unsupervised skill learning (Touati & Ollivier, 2021; Frans et al., 2024; Park et al., 2024b; Kim et al., 2024; Hu et al., 2023), offline goal-conditioned RL (Eysenbach et al., 2019; 2022; Valieva & Banerjee, 2024; Park et al., 2023a; Zheng et al., 2024; Park et al., 2025a), and model-based RL (Mendonca et al., 2021; Mazzaglia et al., 2022). Among these categories, our method is conceptually related to offline unsupervised skill learning approaches (Park et al., 2024b; Touati et al., 2023), which also learns a model that predictions intention. However, our approach differs in that it does not learn multiple skills during pre-training. Our work is complementary to a large body of prior work on using behavioral cloning for pretraining (O'Neill et al., 2024; Team et al., 2024), demonstrating that there are significant additional gains in performance that can be achieved by modeling intentions and occupancy measures simultaneously.

Unsupervised representation learning for RL. Another way to leverage an unlabeled offline dataset is to learn representations that facilitate subsequent downstream task learning. Some works adapt existing representation learning techniques from computer vision, such as contrastive learning (He et al., 2020; Parisi et al., 2022; Nair et al., 2023) and masked autoencoding (He et al., 2022; Xiao et al., 2022). Others design specific methods for RL, including self-predictive representations (Schwarzer et al., 2020; Ni et al., 2024) and temporal distance learning (Sermanet et al., 2018; Ma et al., 2023; Mazoure et al., 2023). Those learned representations are typically used as inputs for policy and value networks. The key challenge with these representation learning methods is that it is often (Laskin et al., 2020), though not always (Zhang et al., 2021), unclear whether the learned representations will facilitate policy adaptation. In our experiments, we demonstrate that learning occupancy models enables faster policy learning.

RL with generative models. Modern generative models have been widely adopted to solve RL problems. Prior work has employed autoregressive models (Vaswani et al., 2017), iterative generative

[1]We will use "successor representations," "occupancy measures," and "occupancy models" interchangeably.

models (e.g., denoising diffusion (Sohl-Dickstein et al., 2015; Ho et al., 2020) and flow matching (Liu et al., 2023; Lipman et al., 2023; 2024)), or autoencoders (Kingma & Welling, 2013) to model trajectories (Chen et al., 2021; Janner et al., 2021; 2022; Ajay et al., 2023), environment dynamics (Ding et al., 2024; Alonso et al., 2024), skills (Ajay et al., 2021; Pertsch et al., 2021; Frans et al., 2024), policies (Wang et al., 2023; Hansen-Estruch et al., 2023; Park et al., 2025b), and values (Dong et al., 2025; Agrawalla et al., 2025). We employ a state-of-the-art flow-matching objective (Farebrother et al., 2025) to model discounted state occupancy measures.

Successor representations and successor features. Prior work has used successor representations (Dayan, 1993) and successor features (Barreto et al., 2017) for transfer learning (Barreto et al., 2017; 2018; Borsa et al., 2018; Nemecek & Parr, 2021; Kim et al., 2022), unsupervised RL (Machado et al., 2017; Hansen et al., 2019; Ghosh et al., 2023; Touati et al., 2023; Park et al., 2024b; 2023b; Chen et al., 2023; Zheng et al., 2025; Jain et al., 2023; Zhu et al., 2024), and goal-conditioned RL (Eysenbach et al., 2020; 2022; Zheng et al., 2024). Our method is closely related to prior methods that learn successor representations with generative models (Janner et al., 2020; Thakoor et al., 2022; Tomar et al., 2024; Farebrother et al., 2025). In particular, the most closely related to ours is the prior work by Farebrother et al. (2025), which also uses flow-matching to model the occupancy measures and partly employs the generalized policy improvement (GPI) for policy extraction. Unlike Farebrother et al. (2025), which uses forward-backward representations to capture behavioral intentions and perform GPI over a finite set of intentions, our method employs a latent variable model to learn intentions (Sec. 4.2) and uses an expectile loss to perform implicit GPI (Sec. 4.4). We empirically show that these choices lead to higher returns and success rates (Sec. 5.1, Sec. 5.3).

3 Preliminaries

We consider a Markov decision process (MDP) (Sutton et al., 1998) defined by a state space $\mathcal{S}$, an action space $\mathcal{A}$, an initial state distribution $\rho \in \Delta(\mathcal{S})$, a reward function $r : \mathcal{S} \to \mathbb{R}$, a discount factor $\gamma \in [0, 1)$, and a transition distribution $p : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$, where $\Delta(\cdot)$ denotes the set of all possible probability distributions over a space. We will use h to denote a time step in the MDP and assume the reward function only depends on the state at the current time step $r_h \triangleq r(s_h)$ without loss of generality (Tomar et al., 2024; Frans et al., 2024; Thakoor et al., 2022). In Appendix A.1, we briefly review the definition of value functions and the actor-critic framework in RL.

Occupancy measures. Alternatively, one can summarize the stochasticity over trajectories into the *discounted state occupancy measure* (Dayan, 1993; Eysenbach et al., 2022; Janner et al., 2020; Touati & Ollivier, 2021; Zheng et al., 2024; Myers et al., 2024; Blier et al., 2021) that quantifies the discounted visitation frequency of different states under the policy π. Prior work (Dayan, 1993; Janner et al., 2020; Touati & Ollivier, 2021; Zheng et al., 2024) has shown that the discounted state occupancy measure follows a Bellman equation backing up the probability density at the current time step and the future time steps:

$$p_\gamma^\pi(s_f \mid s, a) = (1-\gamma)\delta_s(s_f) + \gamma \mathbb{E}_{\substack{s' \sim p(s' \mid s, a), \\ a' \sim \pi(a' \mid s')}} \left[p_\gamma^\pi(s_f \mid s', a') \right], \tag{1}$$

where $\delta_s(\cdot)$ denotes the Dirac delta measure centered at s.[2] The discounted state occupancy measure allows us to rewrite the Q-function as a linear function of rewards (Barreto et al., 2017; Touati & Ollivier, 2021; Zheng et al., 2024; Sikchi et al., 2024):

$$Q^\pi(s, a) = \frac{1}{1-\gamma} \mathbb{E}_{s_f \sim p_\gamma^\pi(s_f \mid s, a)} \left[r(s_f) \right]. \tag{2}$$

The alternative (dual (Sikchi et al., 2024)) definition of Q-function (Eq. 2) allows us to cast the policy evaluation step as first learning a generative model $p_\gamma(s_f \mid s, a)$ to simulate the discounted state occupancy measure of π^k and then regressing the estimator Q towards the average reward at states sampled from p_γ (Toussaint & Storkey, 2006; Tomar et al., 2024; Thakoor et al., 2022; Zheng et al., 2024). See Sec. 4.4 for detailed formulation.

[2]The recursive relationship in Eq. 1 starts from the current time step (Eysenbach et al., 2022; Touati & Ollivier, 2021) instead of the next time step as in some prior approaches (Janner et al., 2020; Zheng et al., 2024; Thakoor et al., 2022).

Flow matching and TD flows. Flow matching (Lipman et al., 2023; 2024; Liu et al., 2023; Albergo & Vanden-Eijnden, 2023) refers to a family of generative models based on ordinary differential equations (ODEs), which are close cousins of denoising diffusion models (Sohl-Dickstein et al., 2015; Song et al., 2021; Ho et al., 2020), which instead solve a stochastic differential equation (SDE). The deterministic nature of ODEs equips flow-matching methods with more stable learning objectives and faster inference speed than denoising diffusion models (Lipman et al., 2023; 2024; Verine et al., 2023). In Appendix A.2, we discuss the problem setting and the standard learning objective for flow matching.

In the context of RL, prior work has used flow matching to estimate the discounted state occupancy measure (Farebrother et al., 2025) by incorporating the Bellman equation (Eq. 1) into the conditional flow matching loss (Eq. 10), resulting in a temporal difference flow matching procedure (TD flows) (Farebrother et al., 2025). In Appendix A.3, we discuss the detailed formulations of the TD flow objective for a target policy π. Choosing the target policy π to be the same as the behavioral policy β, we obtain a SARSA (Rummery & Niranjan, 1994) variant of the loss optimizing the SARSA flows. We will use the SARSA variant of the TD flow objective to learn our generative occupancy models in Sec. 4.3.

4 Intention-conditioned flow occupancy models

In this section, we will introduce our method for pre-training and fine-tuning in RL. After formalizing the problem setting, we will dive into the latent variable model for pre-training an intention encoder and flow occupancy model. After pre-training the occupancy models, our method will extract polices for solving different tasks by invoking a generalized policy improvement procedure (Barreto et al., 2017). We refer to our method as **intention-conditioned flow occupancy models (InFOM)**.

4.1 Problem setting

We consider learning with purely *offline* datasets, where an unlabeled (reward-free) dataset of transitions $D = \{(s, a, s', a')\}$ collected by the behavioral policy β is provided for pre-training and a reward-labeled dataset $D_{\text{reward}} = \{(s, a, r)\}$ collected by some other policy $\tilde{\beta}$ on a downstream task is used for fine-tuning. Importantly, the behavioral policy β used to collect D can consist of a mixture of policies used by different users to complete distinct tasks. We will call this heterogeneous structure of the unlabeled datasets "intentions," which are latent vectors zs in some latent space $\mathcal{Z}$. In practice, these intentions can refer to desired goal images or language instructions that index the behavioral policy $\beta = \{\beta(\cdot \mid \cdot, z) : z \in \mathcal{Z}\}$. Because these latent intentions are *unobserved* to the pre-training algorithm, we want to infer them as a latent random variable Z from the offline dataset, similar to prior work (Hausman et al., 2017; Li et al., 2017; Henderson et al., 2017). In Appendix B.2, we include discussions distinguishing our problem setting from meta RL and multi-task RL problems.

During pre-training, our method exploits the heterogeneous structure of the unlabeled dataset and extracts actionable information by *(1)* inferring intentions of the data collection policy and *(2)* learning occupancy models to predict long-horizon future states based on those intentions (Sec. 4.2 & 4.3). During fine-tuning, we first recover a set of intention-conditioned Q functions by regressing towards average rewards at future states generated by the occupancy models, and then extract a policy to maximize task-specific discounted cumulative returns (Sec. 4.4). Our method builds upon an assumption regarding the consistency of latent intentions.

Assumption 1 (Consistency)**.** *The unlabeled dataset D for pre-training is obtained by executing a behavioral policy following a mixture of unknown intentions $z \in \mathcal{Z}$. We assume that consecutive transitions (s, a) and (s', a') share the same intention.*

The consistency of intentions across transitions enables both intention inference using two sets of transitions and dynamic programming over trajectory segments. See Appendix B.1 for justifications of this assumption.

4.2 Variational intention inference

The goal of our pre-training framework is to learn a latent variable model that captures both long-horizon temporal information and unknown user intentions in the unlabeled datasets.

This part of our method aims to infer the intention z based on consecutive transitions (s, a, s', a') using the encoder $p_e(z \mid s', a')$ and predict the occupancy measures of a future state s_f using the occupancy models $q_d(s_f \mid s, a, z)$. We want to maximize the likelihood of observing a future state s_f starting from a state-action pair (s, a) (amortized variational inference (Kingma & Welling, 2013; Margossian & Blei, 2024)), both sampled from the unlabeled dataset D following the joint behavioral distribution $p^\beta(s, a, s_f) = p^\beta(s, a)p^\beta(s_f \mid s, a)$:

$$\max_{q_d} \mathbb{E}_{p^\beta(s,a,s_f)} \left[\log q_d(s_f \mid s, a)\right]$$

$$\geq \max_{p_e, q_d} \mathbb{E}_{p^\beta(s,a,s_f,s',a')} \left[\mathbb{E}_{p_e(z|s',a')} \left[\log q_d(s_f \mid s, a, z)\right] - \lambda D_{\text{KL}}(p_e(z \mid s', a') \parallel p(z))\right], \tag{3}$$

where $p(z) = \mathcal{N}(0, I)$ denotes an uninformative standard Gaussian prior over intentions, $\lambda \geq 1$ denotes the coefficient that controls the strength of the KL divergence regularization. In practice, we can use any $\lambda \geq 0$ because rescaling the *input* (s, a, s_f), similar to normalizing the range of images from $\{0, \cdots, 255\}$ to $[0, 1]$ in the original VAE (Kingma & Welling, 2013), preserves the ELBO. We defer the full derivation of the evidence lower bound (ELBO) and the explanation of λ to Appendix C.1. Inferring the intention z from the next transition (s', a') follows from our consistency assumption (Assump. 1), and is important for avoiding overfitting (Frans et al., 2024). Importantly, p_e and q_d are optimized *jointly* with this objective. One way of understanding this ELBO is as maximizing an information bottleneck with the chain of random variables $(S', A') \to Z \to (S, A, S_f)$. See Appendix C.1 for the connection.

We use flow matching to reconstruct the discounted state occupancy measure rather than maximizing the likelihood directly, resulting in minimizing a surrogate objective:

$$\min_{p_e, q_d} \mathcal{L}_{\text{Flow}}(q_d, p_e) + \lambda \mathbb{E}_{p^\beta(s',a')} \left[D_{\text{KL}}(p_e(z \mid s', a') \parallel p(z))\right]. \tag{4}$$

We use $\mathcal{L}_{\text{Flow}}$ to denote a placeholder for the flow matching loss and will instantiate this loss for the flow occupancy models q_d next.

4.3 Predicting the future via SARSA flows

We now present the objective used to learn the flow occupancy models, where we first introduce some motivations and desiderata and then specify the actual loss. Given an unlabeled dataset D and an intention encoder $p_e(z \mid s', a')$, the goal is to learn a *generative* occupancy model $q_d(s_f \mid s, a, z)$ that approximates the discounted state occupancy measure of the behavioral policy conditioned on different intentions, i.e., $q_d(s_f \mid s, a, z) \approx p^\beta(s_f \mid s, a, z)$. We will use $v_d : [0, 1] \times \mathcal{S} \times \mathcal{S} \times \mathcal{A} \times \mathcal{Z} \to \mathcal{S}$ to denote the time-dependent vector field that corresponds to q_d. There are two desired properties of the learned occupancy models: *(1)* distributing the peak probability density to multiple s_f, i.e., modeling multimodal structure, and *(2)* stitching together trajectory segments that share some transitions in the dataset, i.e., enabling combinatorial generalization. The first property motivates us to use an expressive flow-matching model (Lipman et al., 2024), while the second property motivates us to learn those occupancy models using temporal difference approaches (Janner et al., 2020; Tomar et al., 2024; Farebrother et al., 2025). Prior work (Farebrother et al., 2025) has derived the TD version of the regular (Monte Carlo) flow matching loss (Eq. 10) that incorporates the Bellman backup into the flow matching procedure, showing the superiority in sample efficiency and the capability of dynamic programming. We will adopt the same idea and use the SARSA variant of the TD flow loss (Eq. 11) to learn our intention-conditioned flow occupancy models:

$$\mathcal{L}_{\text{SARSA flow}}(v_d, p_e) = (1 - \gamma)\mathcal{L}_{\text{SARSA current flow}}(v_d, p_e) + \gamma \mathcal{L}_{\text{SARSA future flow}}(v_d, p_e), \tag{5}$$

$$\mathcal{L}_{\text{SARSA current flow}}(v_d, p_e) = \mathbb{E}_{\substack{(s,a,s',a') \sim p^\beta(s,a,s',a'),\\ z \sim p_e(z|s',a'),\\ t \sim \textsc{Unif}([0,1]), \epsilon \sim \mathcal{N}(0,I)}} \left[\|v(t, s^t, s, a, z) - (s - \epsilon)\|_2^2\right],$$

$$\mathcal{L}_{\text{SARSA future flow}}(v_d, p_e) = \mathbb{E}_{\substack{(s,a,s',a') \sim p^\beta(s,a,s',a'),\\ z \sim p_e(z|s',a'),\\ t \sim \textsc{Unif}([0,1]), \epsilon \sim \mathcal{N}(0,I)}} \left[\|v_d(t, \bar{s}_f^t, s, a, z) - \bar{v}_d(t, \bar{s}_f^t, s', a', z)\|_2^2\right].$$

Importantly, incorporating the information from latent intentions into the flow occupancy models allows us to *(1)* use the simpler and more stable SARSA bootstrap instead of the Q-learning style bootstrap (Eq. 11) on large datasets, *(2)* generalize over latent intentions, avoiding counterfactual errors. Sec. 5.2 visualizes the latent intentions, and Appendix F.2 contains additional experiments.

4.4 Generative value estimation and implicit generalized policy improvement

We next discuss the fine-tuning process in our algorithm. Our fine-tuning method builds on the dual perspective of value estimation introduced in the preliminaries (Eq. 2). We first estimate a *set* of intention-conditioned Q functions using regression and then use those intention-conditioned Q functions to extract a policy, utilizing generalized policy improvement (GPI) (Barreto et al., 2017). The key idea of GPI is that, in addition to taking the maximum over the actions, we can also take the maximum over the intentions. In our setting, the number of intentions is infinite—one for every choice of continuous z. Thus, taking the maximum over the intentions is both nontrivial and susceptible to instability (Sec. 5.3). We address this issue by replacing the greedy "max" with an upper expectile loss, resulting in an implicit generalized policy improvement procedure.

Generative value estimation. Given a reward-labeled dataset D_{reward} and the pre-trained flow occupancy models q_d, we can estimate intention-conditioned Q values for a downstream task. Specifically, for a fixed latent intention $z \in \mathcal{Z}$, we first sample a set of N future states from the flow occupancy models, $s_f^{(1)}, \cdots, s_f^{(N)} : s_f^{(i)} \sim q_d(s_f \mid s, a, z)$, and then constructs a Monte Carlo (MC) estimation of the Q function using those generative samples:[3]

$$Q_z(s,a) = \frac{1}{(1-\gamma)N} \sum_{i=1}^{N} r\left(s_f^{(i)}\right),\ s_f^{(i)} \sim q_d(s_f \mid s, a, z), \tag{6}$$

where $r(\cdot)$ is the reward function or a learned reward predictor. Importantly, the choice of the number of future states N affects the accuracy and variance of our Q estimate. Ablation experiments in Appendix F.11 indicate that $N = 16$ works effectively in our experiments. Note that we choose to sample z from the prior $p(z)$ instead of from the posterior $q_d(z \mid s', a')$, resembling drawing random samples from a variational auto-encoder (Kingma & Welling, 2013). We include an ablation study in Appendix F.5, comparing the effect of fine-tuning with latents from the prior $p(z)$ and the posterior $q_d(z \mid s', a')$. In practice, we find sampling from the prior $p(z)$ worked well in our experiments.

Implicit generalized policy improvement. We can then use those MC estimation of Q functions to learn a policy by invoking the generalized policy improvement. The naive GPI requires sampling a finite set of latent intentions from the prior distribution $p(z)$, $z^{(1)}, \cdots, z^{(M)} :\ z^{(j)} \sim p(z)$ and greedily choose one Q_z to update the policy (Barreto et al., 2017):

$$\arg\max_{\pi} \mathbb{E}_{\substack{s \sim p^{\tilde{\beta}}(s),\, a \sim \pi(a|s) \\ z^{(1)}, \cdots, z^{(M)}:\, z^{(j)} \sim p(z)}} \left[\max_{z^{(j)}} Q_{z^{(j)}}(s,a) \right].$$

Despite its simplicity, the naive GPI suffers from two main disadvantages. First, using the maximum Q over a finite set of latent intentions to approximate the maximum Q over an infinite number of intentions results in local optima. Second, when we take gradients of this objective with respect to the policy, the chain rule gives one term involving $\nabla_a q_d(s_f|s,a,z)$. Thus, computing the gradients requires differentiating through the ODE solver (backpropagating through time (Park et al., 2025b)), which is unstable. We address these challenges by learning an explicit scalar Q function to distill the MC estimation of intention-conditioned Q functions. This approach is appealing because gradients of the Q function no longer backpropagate through the ODE solver. We also replace the "max" over a finite set of intention-conditioned Q functions with an upper expectile loss L_2^μ (Kostrikov et al., 2022), resulting in the following critic loss

$$\mathcal{L}(Q) = \mathbb{E}_{(s,a)\sim p^{\tilde{\beta}}(s,a),\, z \sim p(z)} \left[L_2^\mu \left(Q_z(s,a) - Q(s,a) \right) \right], \tag{7}$$

where $L_2^\mu(x) = |\mu - \mathbb{1}(x < 0)| x^2$ and $\mu \in [0.5, 1)$. In Appendix C.2, we discuss the intuition and theoretical soundness of this distillation step. After distilling the intention-conditioned Q functions into a single function, we can extract the policy by selecting actions to maximize Q with a behavioral cloning regularization (Fujimoto & Gu, 2021) using the actor loss

$$\mathcal{L}(\pi) = -\mathbb{E}_{(s,a)\sim p^{\tilde{\beta}}(s,a), a^\pi \sim \pi(a^\pi|s)} [Q(s, a^\pi) + \alpha \log \pi(a \mid s)], \tag{8}$$

where α controls the regularization strength. We use the behavioral cloning regularization to both reduce errors from sampling out-of-distribution (OOD) actions (Kumar et al., 2020; Fujimoto & Gu,

[3] We omit the dependency of Q_z on $s_f^{(1)}, \cdots, s_f^{(N)}$ to simplify notations.

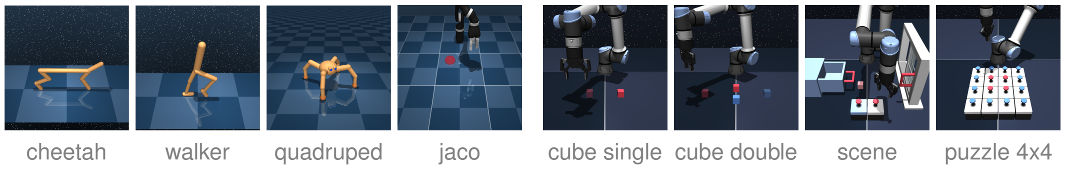
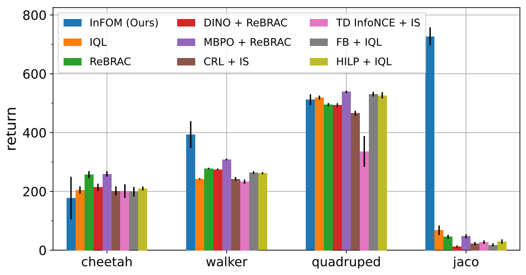
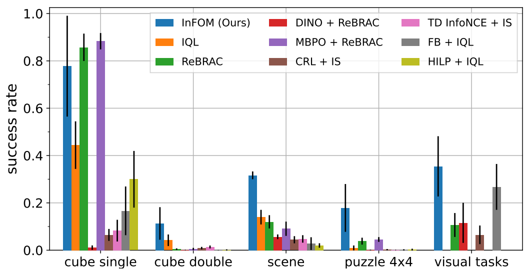


Figure 2: **Domains for evaluation.** *(Left)* ExORL domains (16 state-based tasks). *(Right)* OGBench domains (20 state-based tasks and 4 image-based tasks).



(a) 16 state-based ExORL tasks from Yarats et al. (2022). We average over 4 tasks for each domain.



(b) 20 state-based and 4 image-based OGBench tasks from Park et al. (2025a). We average over 5 tasks for each state-based domain and average over 4 visual tasks.

Figure 3: **Evaluation on ExORL and OGBench tasks.** We compare InFOM against prior methods that utilize various learning paradigms on task-agnostic pre-training and task-specific fine-tuning. InFOM performs similarly to, if not better than, prior methods on 7 out of the 9 domains, including the most challenging visual tasks. We report means and standard deviations over 8 random seeds (4 random seeds for image-based tasks) with error bars indicating one standard deviation. See Table 4 for full results.

2021) and mitigate error propagations through overestimated Q_z values. Ablation experiments in Appendix F.7 and Appendix F.12 show that this behavioral cloning regularization is important for improving the policy performance. Taken together, we call the expectile Q distillation step (Eq. 7) and the policy optimization step (Eq. 8) *implicit generalized policy improvement (implicit GPI)*.

Algorithm summary. We use neural networks to parameterize the intention encoder p_ϕ, the vector field of the occupancy models v_θ, the reward predictor r_η, the critic Q_ψ, and the policy π_ω. We consider two stages: pre-training and fine-tuning. In Alg. 1, we summarize the pre-training process of InFOM. InFOM pre-trains *(1)* the vector field v_θ using the SARSA flow loss (Eq. 5) and *(2)* the intention encoder p_ϕ using the ELBO (Eq. 3). Alg. 2 shows the pseudocode of InFOM for fine-tuning. InFOM mainly learns *(1)* the reward predictor r_η via simple regression, *(2)* the critic Q_ψ using expectile distillation (Eq. 7), and *(3)* the policy π_ω by conservatively maximizing the Q_ψ (Eq. 8). The open-source implementation is available in the supplementary materials.

5 Experiments

Our experiments start with comparing InFOM to prior methods that first pre-train on reward-free datasets and then fine-tune on reward-labeled datasets, measuring the performance on downstream tasks. We then study the two main components of our method: the variational intention encoder and the implicit GPI policy extraction strategy. Visualizations of the latent intention inferred by our variational intention encoder show alignment with the underlying ground-truth intentions. Our ablation experiments reveal the effect of the implicit GPI policy extraction strategy. We also include additional experiments showing InFOM enables faster policy learning during fine-tuning in Appendix F.3. Our algorithm is robust to various choices of hyperparameters (Appendix F.12). Following prior work (Park et al., 2025b), all experiments report means and standard deviations across 8 random seeds for state-based tasks and 4 random seeds for image-based tasks.

5.1 Comparing to prior pre-training and fine-tuning methods

Our experiments study whether the proposed method (InFOM), which captures actionable information conditioned on user intentions from unlabeled datasets, enables effective pre-training and fine-tuning. We select 36 state-based and 4 image-based tasks across diverse robotic navigation and manipula-

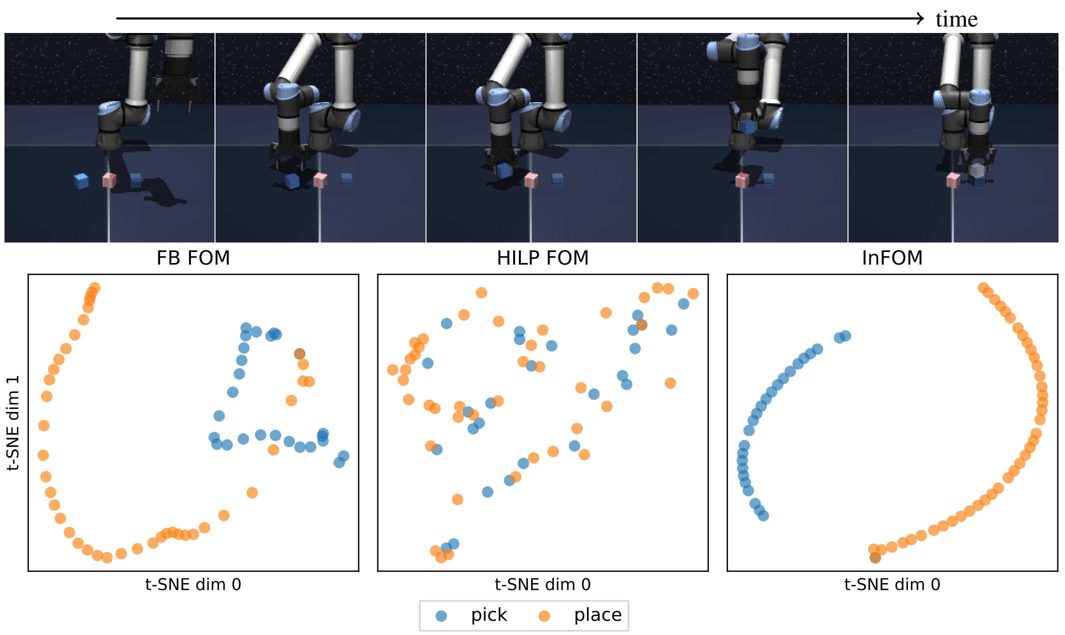


Figure 4: **Visualization of latent intentions.** *(Top)* The optimal policy picks up the blue block from the left and places it on the right. *(Bottom)* Using t-SNE (Maaten & Hinton, 2008), we visualize the latent intentions inferred by the variational intention encoder in InFOM, comparing against latent representations inferred by HILP and FB for learning FOMs. The predictions from InFOM align with the underlying intentions. See Sec. 5.2 for details and Appendix E for more visualizations.

tion domains and compare against 8 baselines. The models pre-trained by those methods include behavioral cloning policies (Kostrikov et al., 2022; Tarasov et al., 2023a), transition models (Janner et al., 2019), representations (Caron et al., 2021), discriminative classifiers that predict occupancy measures (Eysenbach et al., 2022; Zheng et al., 2024), and latent skills (Touati & Ollivier, 2021; Park et al., 2024b). We defer the detailed discussions about benchmarks and datasets to Appendix D.1 and the rationale for choosing different baselines to Appendix D.2. Whenever possible, we use the same hyperparameters for all methods (Table 1). See Appendix D.3 for details of the evaluation protocol and Appendix D.4 for implementations and hyperparameters of each method.

We report results in Fig. 3, aggregating over four tasks in each domain of ExORL and five tasks in each domain of OGBench, and present the full results in Table 4. These results show that InFOM matches or surpasses all baselines on six out of eight domains. On ExORL benchmarks, all methods perform similarly on the two easier domains (`cheetah` and `quadruped`), while InFOM can obtain $20\times$ improvement on `jaco`, where baselines only make trivial progress (Table 4). We suspect the outsized improvement on the `jaco` task is because of the high-dimensional state space (twice that of the other ExORL tasks (Yarats et al., 2022)) and because it has sparse rewards; Appendix Fig. 13 supports this hypothesis by showing that the ReBRAC baseline achieves significantly higher returns when using dense rewards. On those more challenging state-based manipulation tasks from OGBench, we find a marked difference between baselines and InFOM; our method achieves 36% higher success rate over the best performing baseline. In addition, InFOM is able to outperform the best baseline by 31% using RGB images as input directly (`visual tasks`). We hypothesize that the baselines fail to solve these more challenging tasks because of the semi-sparse reward functions. In contrast, our method can explore different regions of the state space using the different intentions, thereby addressing the challenge of reward sparsity. We conjecture that the variance of InFOM across seeds in some experiments (e.g., `cheetah`, `cube single`, and `puzzle 4x4`) reflects stochasticity in the MC Q estimates (Eq. 6), which might be mitigated by increasing the number of sampled future states (See Appendix F.11). In Appendix F.1, we compare InFOM against selective baselines on real robotics datasets, showing 34% improvement.

5.2 Visualizing latent intentions

Our next experiment studies the intention encoder in our algorithm. To investigate whether the proposed method discovers distinct user intentions from an unlabeled dataset, we visualize latent intentions inferred by our variational intention encoder. We include comparisons against two alternative intention encoding mechanisms proposed by prior methods. Specifically, we consider replacing the variational intention encoder with either *(1)* a set of Hilbert representations (Park et al., 2024b) or *(2)* a set of forward-backward representations (Touati & Ollivier, 2021), and then pre-training the flow occupancy models (FOM) conditioned on these two sets of representations. We call these two variants HILP + FOM and FB + FOM. Note that FB + FOM is equivalent to TD flows with GPI in Farebrother et al. (2025). Using t-SNE (Maaten & Hinton, 2008), we visualize latent intentions predicted by these three methods on `cube double task 1` from the OGBench benchmarks.

Fig. 4 shows the optimal trajectory, where the manipulator picks the blue block from the left and then places it on the right, and the visualizations. The 2D t-SNE visualizations indicate that both FB + FOM and HILP + FOM infer mixed latent intentions for "pick" and "place" behaviors, while InFOM predicts a sequence of latent intentions with clear clustering. This result suggests that InFOM is capable of inferring latent intentions that align with the underlying ground-truth intentions. See Appendix E for more visualizations. In Appendix F.2, we include additional experiments comparing the downstream performance between InFOM and HILP + FOM and FB + FOM. Results in Appendix Fig. 9 suggest that InFOM can outperform those two baselines on 3 of 4 tasks.

5.3 Importance of the Implicit Generalized Policy Improvement

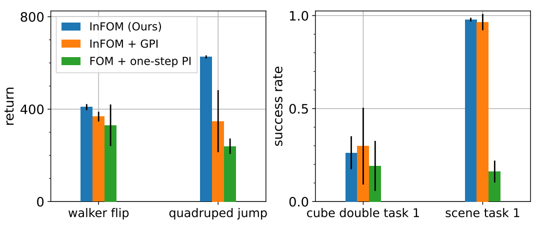


Figure 5: **Comparison to alternative policy extraction strategies.** We compare InFOM to alternative policy extraction strategies based on the standard generalized policy improvement or one-step policy improvement. Our method is 44% more performant with $8\times$ smaller variance than the variant using the standard GPI. See Sec. 5.3 for details.

Our final experiments study different approaches for policy optimization. We hypothesize that our proposed method is more efficient and robust than other policy extraction strategies. To test this hypothesis, we conduct ablation experiments on one task in the ExORL benchmarks (`quadruped jump`) and another task taken from the OGBench benchmarks (`scene task 1`), again following the evaluation protocols in Appendix D.3. We compare two alternative policy learning approaches in the fine-tuning phase. First, we ablate the effect of the upper expectile loss by comparing against the standard GPI, which maximizes Q functions over a finite set of intentions $\{z^{(1)}, \cdots, z^{(M)}\}$. We choose $M = 32$ latent intentions to balance between performance and compute budget, and call this variant InFOM + GPI. Second, we ablate the effect of the variational intention encoder by removing the intention dependency in the flow occupancy models and extracting the policy via one-step policy improvement (PI) (Wang et al., 2018; Brandfonbrener et al., 2021; Peters & Schaal, 2007; Peters et al., 2010). We call this method FOM + one-step PI and defer the detailed formulation to Appendix C.3.

As shown in Fig. 5, InFOM achieves significantly higher returns and success rates than its variant based on one-step policy improvement, suggesting the importance of inferring user intentions. Compared with its GPI counterpart, our method is 44% more performant with $8\times$ smaller variance (the error bar indicates one standard deviation), demonstrating that the implicit GPI indeed performs a relaxed maximization over intentions while maintaining robustness.

Additional experiments. In Appendix F.3, we include additional ablations showing that InFOM enables faster policy learning. Appendix F.4 ablate InFOM against a variant of InFOM with a set of discrete latents trained vector quantization loss, showing that the continous latent space generally performs better. In Appendix F.8, we relate the diversity of the pre-training datasets to their sizes. The dataset size ablations in Appendix F.9 show that using sufficient pre-training and fine-tuning data is important. Appendix F.10 study the effects of fine-tuning on suboptimal datasets. Our hyperparameter ablations can be found in Appendix F.12.

Alternative generative occupancy models. Farebrother et al. (2025) has already discussed using alternative prior generative modeling approaches to learn the occupancy measure. Specifically, they compare flow-based occupancy models against representative generative methods, including denoising diffusion (Ho et al., 2020), VAE (Kingma & Welling, 2013; Higgins et al., 2017), and GAN (Goodfellow et al., 2014). Results in Fig. 2 of Farebrother et al. (2025) show that flow-based occupancy models (TD^2-CFM in the figure) outperforms alternative generative methods in modeling the occupancy measures. For this reason, we do not include comparisons against alternative generative occupancy models to distinguish our contributions.

6 Conclusion

In this work, we presented InFOM, a method that captures diverse intentions and their long-term behaviors from an unstructured dataset, leveraging the expressivity of flow models. We empirically showed that the intentions captured in flow occupancy models enable effective and efficient fine-tuning, outperforming prior unsupervised pre-training approaches on diverse state- and image-based domains.

Limitations. One limitation of InFOM is that our reduction from trajectories to consecutive state-action pairs might not always accurately capture the original intentions in the trajectories. While we empirically showed that this simple approach is sufficient to achieve strong performance on our benchmark tasks, it can be further improved with alternative trajectory encoding techniques and data collection strategies, which we leave for future work.

Reproducibility statement

We implement InFOM and all baselines in the same codebase using JAX (Bradbury et al., 2018). Our implementations build on top of OGBench's and FQL's implementations (Park et al., 2025a;b). We include the common hyperparameters for all the methods in Appendix Table 1, the hyperparameters for InFOM in Appendix Table 2 and Appendix Table 3, and the hyperparameters for baselines in Appendix Table 3. All the experiments were run on a single NVIDIA A6000 GPU and can be finished in 4 hours for state-based tasks and 12 hours for image-based tasks. We provide open-source implementations of InFOM and all baselines in the supplementary materials.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

Bhavya Agrawalla, Michal Nauman, Khush Agrawal, and Aviral Kumar. floq: Training critics via flow-matching for scaling compute in value-based rl. *arXiv preprint arXiv:2509.06863*, 2025.

Anurag Ajay, Aviral Kumar, Pulkit Agrawal, Sergey Levine, and Ofir Nachum. OPAL: Offline primitive discovery for accelerating offline reinforcement learning. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=V69LGwJ0lIN.

Anurag Ajay, Yilun Du, Abhi Gupta, Joshua B. Tenenbaum, Tommi S. Jaakkola, and Pulkit Agrawal. Is conditional generative modeling all you need for decision making? In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=sP1fo2K9DFG.

Michael Samuel Albergo and Eric Vanden-Eijnden. Building normalizing flows with stochastic interpolants. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=li7qeBbCR1t.

Alexander A. Alemi, Ian Fischer, Joshua V. Dillon, and Kevin Murphy. Deep variational information bottleneck. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=HyxQzBceg.

Eloi Alonso, Adam Jelley, Vincent Micheli, Anssi Kanervisto, Amos J Storkey, Tim Pearce, and François Fleuret. Diffusion for world modeling: Visual details matter in atari. *Advances in Neural Information Processing Systems*, 37:58757–58791, 2024.

David Barber and Felix Agakov. The im algorithm: a variational approach to information maximization. *Advances in neural information processing systems*, 16(320):201, 2004.

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

Andre Barreto, Diana Borsa, John Quan, Tom Schaul, David Silver, Matteo Hessel, Daniel Mankowitz, Augustin Zidek, and Remi Munos. Transfer in deep reinforcement learning using successor features and generalised policy improvement. In *International Conference on Machine Learning*, pp. 501–510. PMLR, 2018.

Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013.

Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, et al. π_0: A vision-language-action flow model for general robot control. *arXiv preprint arXiv:2410.24164*, 2024.

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

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

James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: composable transformations of Python+NumPy programs, 2018. URL http://github.com/jax-ml/jax.

David Brandfonbrener, Will Whitney, Rajesh Ranganath, and Joan Bruna. Offline rl without off-policy evaluation. *Advances in neural information processing systems*, 34:4933–4946, 2021.

Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Jasmine Hsu, et al. Rt-1: Robotics transformer for real-world control at scale. *arXiv preprint arXiv:2212.06817*, 2022.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

Yuri Burda, Harrison Edwards, Amos Storkey, and Oleg Klimov. Exploration by random network distillation. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=H1lJJnR5Ym.

Andrew Campbell, Jason Yim, Regina Barzilay, Tom Rainforth, and Tommi Jaakkola. Generative flows on discrete state-spaces: Enabling multimodal flows with applications to protein co-design. In *International Conference on Machine Learning*, pp. 5453–5512. PMLR, 2024.

Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 9650–9660, 2021.

Boyuan Chen, Chuning Zhu, Pulkit Agrawal, Kaiqing Zhang, and Abhishek Gupta. Self-supervised reinforcement learning that transfers using random features. *Advances in Neural Information Processing Systems*, 36:56411–56436, 2023.

Lili Chen, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Misha Laskin, Pieter Abbeel, Aravind Srinivas, and Igor Mordatch. Decision transformer: Reinforcement learning via sequence modeling. *Advances in neural information processing systems*, 34:15084–15097, 2021.

Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. *Advances in neural information processing systems*, 31, 2018.

Quan Dao, Hao Phung, Binh Nguyen, and Anh Tran. Flow matching in latent space. *arXiv preprint arXiv:2307.08698*, 2023.

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

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers)*, pp. 4171–4186, 2019.

Zihan Ding, Amy Zhang, Yuandong Tian, and Qinqing Zheng. Diffusion world model. *arXiv e-prints*, pp. arXiv–2402, 2024.

Perry Dong, Chongyi Zheng, Chelsea Finn, Dorsa Sadigh, and Benjamin Eysenbach. Value flows. *arXiv preprint arXiv:2510.07650*, 2025.

Yan Duan, John Schulman, Xi Chen, Peter L Bartlett, Ilya Sutskever, and Pieter Abbeel. Rl^2: Fast reinforcement learning via slow reinforcement learning. *arXiv preprint arXiv:1611.02779*, 2016.

Rick Durrett. *Probability: theory and examples*, volume 49. Cambridge university press, 2019.

Lasse Espeholt, Hubert Soyer, Remi Munos, Karen Simonyan, Vlad Mnih, Tom Ward, Yotam Doron, Vlad Firoiu, Tim Harley, Iain Dunning, et al. Impala: Scalable distributed deep-rl with importance weighted actor-learner architectures. In *International conference on machine learning*, pp. 1407–1416. PMLR, 2018.

Benjamin Eysenbach, Abhishek Gupta, Julian Ibarz, and Sergey Levine. Diversity is all you need: Learning skills without a reward function. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=SJx63jRqFm.

Benjamin Eysenbach, Ruslan Salakhutdinov, and Sergey Levine. C-learning: Learning to achieve goals via recursive classification. *arXiv preprint arXiv:2011.08909*, 2020.

Benjamin Eysenbach, Tianjun Zhang, Sergey Levine, and Russ R Salakhutdinov. Contrastive learning as goal-conditioned reinforcement learning. *Advances in Neural Information Processing Systems*, 35:35603–35620, 2022.

Jesse Farebrother, Matteo Pirotta, Andrea Tirinzoni, Remi Munos, Alessandro Lazaric, and Ahmed Touati. Temporal difference flows. In *ICLR 2025 Workshop on World Models: Understanding, Modelling and Scaling*, 2025. URL https://openreview.net/forum?id=nYL0pn3z3u.

Kevin Frans, Seohong Park, Pieter Abbeel, and Sergey Levine. Unsupervised zero-shot reinforcement learning via functional reward encodings. In *International Conference on Machine Learning*, pp. 13927–13942. PMLR, 2024.

Kevin Frans, Danijar Hafner, Sergey Levine, and Pieter Abbeel. One step diffusion via shortcut models. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=OlzB6LnXcS.

Scott Fujimoto and Shixiang Shane Gu. A minimalist approach to offline reinforcement learning. *Advances in neural information processing systems*, 34:20132–20145, 2021.

Scott Fujimoto, Herke Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *International conference on machine learning*, pp. 1587–1596. PMLR, 2018.

Dibya Ghosh, Chethan Anand Bhateja, and Sergey Levine. Reinforcement learning from passive data via latent intentions. In *International Conference on Machine Learning*, pp. 11321–11339. PMLR, 2023.

Ian J Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014.

Will Grathwohl, Ricky T. Q. Chen, Jesse Bettencourt, and David Duvenaud. Scalable reversible generative models with free-form continuous dynamics. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=rJxgknCcK7.

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

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pp. 1861–1870. Pmlr, 2018.

Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. Mastering diverse domains through world models. *arXiv preprint arXiv:2301.04104*, 2023.

Steven Hansen, Will Dabney, Andre Barreto, Tom Van de Wiele, David Warde-Farley, and Volodymyr Mnih. Fast task inference with variational intrinsic successor features. *arXiv preprint arXiv:1906.05030*, 2019.

Philippe Hansen-Estruch, Ilya Kostrikov, Michael Janner, Jakub Grudzien Kuba, and Sergey Levine. Idql: Implicit q-learning as an actor-critic method with diffusion policies. *arXiv preprint arXiv:2304.10573*, 2023.

Karol Hausman, Yevgen Chebotar, Stefan Schaal, Gaurav Sukhatme, and Joseph J Lim. Multi-modal imitation learning from unstructured demonstrations using generative adversarial nets. *Advances in neural information processing systems*, 30, 2017.

Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 9729–9738, 2020.

Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 16000–16009, 2022.

Peter Henderson, Wei-Di Chang, Pierre-Luc Bacon, David Meger, Joelle Pineau, and Doina Precup. Optiongan: Learning joint reward-policy options using generative adversarial inverse reinforcement learning. *ArXiv*, abs/1709.06683, 2017. URL https://api.semanticscholar.org/CorpusID:7079525.

Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*, 2016.

Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. beta-VAE: Learning basic visual concepts with a constrained variational framework. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=Sy2fzU9gl.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

Hao Hu, Yiqin Yang, Jianing Ye, Ziqing Mai, and Chongjie Zhang. Unsupervised behavior extraction via random intent priors. *Advances in Neural Information Processing Systems*, 36:51491–51514, 2023.

Arnav Kumar Jain, Lucas Lehnert, Irina Rish, and Glen Berseth. Maximum state entropy exploration using predecessor and successor representations. *Advances in Neural Information Processing Systems*, 36:49991–50019, 2023.

Michael Janner, Justin Fu, Marvin Zhang, and Sergey Levine. When to trust your model: Model-based policy optimization. *Advances in neural information processing systems*, 32, 2019.

Michael Janner, Igor Mordatch, and Sergey Levine. Gamma-models: Generative temporal difference learning for infinite-horizon prediction. *Advances in neural information processing systems*, 33: 1724–1735, 2020.

Michael Janner, Qiyang Li, and Sergey Levine. Offline reinforcement learning as one big sequence modeling problem. *Advances in neural information processing systems*, 34:1273–1286, 2021.

Michael Janner, Yilun Du, Joshua Tenenbaum, and Sergey Levine. Planning with diffusion for flexible behavior synthesis. In *International Conference on Machine Learning*, pp. 9902–9915. PMLR, 2022.

Scott Jeen, Tom Bewley, and Jonathan Cullen. Zero-shot reinforcement learning from low quality data. *Advances in Neural Information Processing Systems*, 37:16894–16942, 2024.

Jaekyeom Kim, Seohong Park, and Gunhee Kim. Constrained gpi for zero-shot transfer in reinforcement learning. *Advances in Neural Information Processing Systems*, 35:4585–4597, 2022.

Junsu Kim, Seohong Park, and Sergey Levine. Unsupervised-to-online reinforcement learning. *arXiv preprint arXiv:2408.14785*, 2024.

Diederik P Kingma. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Ilya Kostrikov, Ashvin Nair, and Sergey Levine. Offline reinforcement learning with implicit q-learning. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=68n2s9ZJWF8.

Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative q-learning for offline reinforcement learning. *Advances in neural information processing systems*, 33:1179–1191, 2020.

Nathan Lambert, Kristofer Pister, and Roberto Calandra. Investigating compounding prediction errors in learned dynamics models. *arXiv preprint arXiv:2203.09637*, 2022.

Michael Laskin, Aravind Srinivas, and Pieter Abbeel. Curl: Contrastive unsupervised representations for reinforcement learning. In *International conference on machine learning*, pp. 5639–5650. PMLR, 2020.

Xuanlin Li, Kyle Hsu, Jiayuan Gu, Karl Pertsch, Oier Mees, Homer Rich Walke, Chuyuan Fu, Ishikaa Lunawat, Isabel Sieh, Sean Kirmani, et al. Evaluating real-world robot manipulation policies in simulation. *arXiv preprint arXiv:2405.05941*, 2024.

Yunzhu Li, Jiaming Song, and Stefano Ermon. Infogail: Interpretable imitation learning from visual demonstrations. *Advances in neural information processing systems*, 30, 2017.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matthew Le. Flow matching for generative modeling. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=PqvMRDCJT9t.

Yaron Lipman, Marton Havasi, Peter Holderrieth, Neta Shaul, Matt Le, Brian Karrer, Ricky TQ Chen, David Lopez-Paz, Heli Ben-Hamu, and Itai Gat. Flow matching guide and code. *arXiv preprint arXiv:2412.06264*, 2024.

Xingchao Liu, Chengyue Gong, and qiang liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=XVjTT1nw5z.

Jiasen Lu, Dhruv Batra, Devi Parikh, and Stefan Lee. Vilbert: Pretraining task-agnostic visiolinguistic representations for vision-and-language tasks. *Advances in neural information processing systems*, 32, 2019.

Yecheng Jason Ma, Shagun Sodhani, Dinesh Jayaraman, Osbert Bastani, Vikash Kumar, and Amy Zhang. VIP: Towards universal visual reward and representation via value-implicit pre-training. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=YJ7o2wetJ2.

Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605, 2008.

Marlos C Machado, Clemens Rosenbaum, Xiaoxiao Guo, Miao Liu, Gerald Tesauro, and Murray Campbell. Eigenoption discovery through the deep successor representation. *arXiv preprint arXiv:1710.11089*, 2017.

Charles C Margossian and David M Blei. Amortized variational inference: When and why? In *Uncertainty in Artificial Intelligence*, pp. 2434–2449. PMLR, 2024.

Bogdan Mazoure, Benjamin Eysenbach, Ofir Nachum, and Jonathan Tompson. Contrastive value learning: Implicit models for simple offline RL. In *7th Annual Conference on Robot Learning*, 2023. URL https://openreview.net/forum?id=oqOfLP6bJy.

Pietro Mazzaglia, Tim Verbelen, Bart Dhoedt, Alexandre Lacoste, and Sai Rajeswar. Choreographer: Learning and adapting skills in imagination. *arXiv preprint arXiv:2211.13350*, 2022.

Russell Mendonca, Oleh Rybkin, Kostas Daniilidis, Danijar Hafner, and Deepak Pathak. Discovering and achieving goals via world models. *Advances in Neural Information Processing Systems*, 34: 24379–24391, 2021.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.

Vivek Myers, Chongyi Zheng, Anca Dragan, Sergey Levine, and Benjamin Eysenbach. Learning temporal distances: Contrastive successor features can provide a metric structure for decision-making. In *International Conference on Machine Learning*, pp. 37076–37096. PMLR, 2024.

Suraj Nair, Aravind Rajeswaran, Vikash Kumar, Chelsea Finn, and Abhinav Gupta. R3m: A universal visual representation for robot manipulation. In *Conference on Robot Learning*, pp. 892–909. PMLR, 2023.

Mark Nemecek and Ronald Parr. Policy caches with successor features. In *International Conference on Machine Learning*, pp. 8025–8033. PMLR, 2021.

Tianwei Ni, Benjamin Eysenbach, Erfan Seyedsalehi, Michel Ma, Clement Gehring, Aditya Mahajan, and Pierre-Luc Bacon. Bridging state and history representations: Understanding self-predictive rl. *arXiv preprint arXiv:2401.08898*, 2024.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744, 2022.

Abby O'Neill, Abdul Rehman, Abhiram Maddukuri, Abhishek Gupta, Abhishek Padalkar, Abraham Lee, Acorn Pooley, Agrim Gupta, Ajay Mandlekar, Ajinkya Jain, et al. Open x-embodiment: Robotic learning datasets and rt-x models: Open x-embodiment collaboration 0. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 6892–6903. IEEE, 2024.

Simone Parisi, Aravind Rajeswaran, Senthil Purushwalkam, and Abhinav Gupta. The unsurprising effectiveness of pre-trained vision models for control. In *international conference on machine learning*, pp. 17359–17371. PMLR, 2022.

Seohong Park, Dibya Ghosh, Benjamin Eysenbach, and Sergey Levine. Hiql: Offline goal-conditioned rl with latent states as actions. *Advances in Neural Information Processing Systems*, 36:34866–34891, 2023a.

Seohong Park, Oleh Rybkin, and Sergey Levine. Metra: Scalable unsupervised rl with metric-aware abstraction. *arXiv preprint arXiv:2310.08887*, 2023b.

Seohong Park, Kevin Frans, Sergey Levine, and Aviral Kumar. Is value learning really the main bottleneck in offline rl? *arXiv preprint arXiv:2406.09329*, 2024a.

Seohong Park, Tobias Kreiman, and Sergey Levine. Foundation policies with hilbert representations. In *International Conference on Machine Learning*, pp. 39737–39761. PMLR, 2024b.

Seohong Park, Kevin Frans, Benjamin Eysenbach, and Sergey Levine. OGBench: Benchmarking offline goal-conditioned RL. In *The Thirteenth International Conference on Learning Representations*, 2025a. URL https://openreview.net/forum?id=M992mjgKzI.

Seohong Park, Qiyang Li, and Sergey Levine. Flow q-learning. *arXiv preprint arXiv:2502.02538*, 2025b.

Karl Pertsch, Youngwoon Lee, and Joseph Lim. Accelerating reinforcement learning with learned skill priors. In *Conference on robot learning*, pp. 188–204. PMLR, 2021.

Jan Peters and Stefan Schaal. Reinforcement learning by reward-weighted regression for operational space control. In *Proceedings of the 24th international conference on Machine learning*, pp. 745–750, 2007.

Jan Peters, Katharina Mulling, and Yasemin Altun. Relative entropy policy search. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 24, pp. 1607–1612, 2010.

Vitchyr H Pong, Ashvin V Nair, Laura M Smith, Catherine Huang, and Sergey Levine. Offline meta-reinforcement learning with online self-supervision. In *International Conference on Machine Learning*, pp. 17811–17829. Pmlr, 2022.

Ben Poole, Sherjil Ozair, Aaron Van Den Oord, Alex Alemi, and George Tucker. On variational bounds of mutual information. In *International conference on machine learning*, pp. 5171–5180. PMLR, 2019.

Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. Improving language understanding by generative pre-training. 2018.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pp. 8748–8763. PmLR, 2021.

Kate Rakelly, Aurick Zhou, Chelsea Finn, Sergey Levine, and Deirdre Quillen. Efficient off-policy meta-reinforcement learning via probabilistic context variables. In *International conference on machine learning*, pp. 5331–5340. PMLR, 2019.

Allen Z. Ren, Justin Lidard, Lars Lien Ankile, Anthony Simeonov, Pulkit Agrawal, Anirudha Majumdar, Benjamin Burchfiel, Hongkai Dai, and Max Simchowitz. Diffusion policy policy optimization. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=mEpqHvbD2h.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 10684–10695, 2022.

Gavin A Rummery and Mahesan Niranjan. *On-line Q-learning using connectionist systems*, volume 37. University of Cambridge, Department of Engineering Cambridge, UK, 1994.

Andrew Michael Saxe, Yamini Bansal, Joel Dapello, Madhu Advani, Artemy Kolchinsky, Brendan Daniel Tracey, and David Daniel Cox. On the information bottleneck theory of deep learning. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=ry_WPG-A-.

John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International conference on machine learning*, pp. 1889–1897. PMLR, 2015.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Max Schwarzer, Ankesh Anand, Rishab Goel, R Devon Hjelm, Aaron Courville, and Philip Bachman. Data-efficient reinforcement learning with self-predictive representations. *arXiv preprint arXiv:2007.05929*, 2020.

Pierre Sermanet, Corey Lynch, Yevgen Chebotar, Jasmine Hsu, Eric Jang, Stefan Schaal, Sergey Levine, and Google Brain. Time-contrastive networks: Self-supervised learning from video. In *2018 IEEE international conference on robotics and automation (ICRA)*, pp. 1134–1141. IEEE, 2018.

Archit Sharma, Shixiang Gu, Sergey Levine, Vikash Kumar, and Karol Hausman. Dynamics-aware unsupervised discovery of skills. *arXiv preprint arXiv:1907.01657*, 2019.

Harshit Sikchi, Qinqing Zheng, Amy Zhang, and Scott Niekum. Dual RL: Unification and new methods for reinforcement and imitation learning. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=xt9Bu66rqv.

Shagun Sodhani, Amy Zhang, and Joelle Pineau. Multi-task reinforcement learning with context-based representations. In *International conference on machine learning*, pp. 9767–9779. PMLR, 2021.

Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International conference on machine learning*, pp. 2256–2265. pmlr, 2015.

Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=PxTIG12RRHS.

Richard S Sutton, Andrew G Barto, et al. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge, 1998.

Erik Talvitie. Model regularization for stable sample rollouts. In *UAI*, pp. 780–789, 2014.

Denis Tarasov, Vladislav Kurenkov, Alexander Nikulin, and Sergey Kolesnikov. Revisiting the minimalist approach to offline reinforcement learning. *Advances in Neural Information Processing Systems*, 36:11592–11620, 2023a.

Denis Tarasov, Alexander Nikulin, Dmitry Akimov, Vladislav Kurenkov, and Sergey Kolesnikov. Corl: Research-oriented deep offline reinforcement learning library. *Advances in Neural Information Processing Systems*, 36:30997–31020, 2023b.

Yuval Tassa, Yotam Doron, Alistair Muldal, Tom Erez, Yazhe Li, Diego de Las Casas, David Budden, Abbas Abdolmaleki, Josh Merel, Andrew Lefrancq, et al. Deepmind control suite. *arXiv preprint arXiv:1801.00690*, 2018.

Gemini Team, Rohan Anil, Sebastian Borgeaud, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, Katie Millican, et al. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*, 2023.

Octo Model Team, Dibya Ghosh, Homer Walke, Karl Pertsch, Kevin Black, Oier Mees, Sudeep Dasari, Joey Hejna, Tobias Kreiman, Charles Xu, et al. Octo: An open-source generalist robot policy. *arXiv preprint arXiv:2405.12213*, 2024.

Shantanu Thakoor, Mark Rowland, Diana Borsa, Will Dabney, Rémi Munos, and André Barreto. Generalised policy improvement with geometric policy composition. In *International Conference on Machine Learning*, pp. 21272–21307. PMLR, 2022.

Naftali Tishby, Fernando C Pereira, and William Bialek. The information bottleneck method. *arXiv preprint physics/0004057*, 2000.

Manan Tomar, Philippe Hansen-Estruch, Philip Bachman, Alex Lamb, John Langford, Matthew E Taylor, and Sergey Levine. Video occupancy models. *arXiv preprint arXiv:2407.09533*, 2024.

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

Ahmed Touati, Jérémy Rapin, and Yann Ollivier. Does zero-shot reinforcement learning exist? In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=MYEap_OcQI.

Marc Toussaint and Amos Storkey. Probabilistic inference for solving discrete and continuous state markov decision processes. In *Proceedings of the 23rd international conference on Machine learning*, pp. 945–952, 2006.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

Khadichabonu Valieva and Bikramjit Banerjee. Quasimetric value functions with dense rewards. *arXiv preprint arXiv:2409.08724*, 2024.

Aaron Van Den Oord, Oriol Vinyals, et al. Neural discrete representation learning. *Advances in neural information processing systems*, 30, 2017.

Hado Van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 30, 2016.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Alexandre Verine, Benjamin Negrevergne, Yann Chevaleyre, and Fabrice Rossi. On the expressivity of bi-lipschitz normalizing flows. In *Asian Conference on Machine Learning*, pp. 1054–1069. PMLR, 2023.

Homer Rich Walke, Kevin Black, Tony Z Zhao, Quan Vuong, Chongyi Zheng, Philippe Hansen-Estruch, Andre Wang He, Vivek Myers, Moo Jin Kim, Max Du, et al. Bridgedata v2: A dataset for robot learning at scale. In *Conference on Robot Learning*, pp. 1723–1736. PMLR, 2023.

Qing Wang, Jiechao Xiong, Lei Han, Han Liu, Tong Zhang, et al. Exponentially weighted imitation learning for batched historical data. *Advances in Neural Information Processing Systems*, 31, 2018.

Zhendong Wang, Jonathan J Hunt, and Mingyuan Zhou. Diffusion policies as an expressive policy class for offline reinforcement learning. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=AHvFDPi-FA.

Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine learning*, 8:279–292, 1992.

Tete Xiao, Ilija Radosavovic, Trevor Darrell, and Jitendra Malik. Masked visual pre-training for motor control. *arXiv preprint arXiv:2203.06173*, 2022.

Denis Yarats, David Brandfonbrener, Hao Liu, Michael Laskin, Pieter Abbeel, Alessandro Lazaric, and Lerrel Pinto. Don't change the algorithm, change the data: Exploratory data for offline reinforcement learning. In *ICLR 2022 Workshop on Generalizable Policy Learning in Physical World*, 2022. URL https://openreview.net/forum?id=Su-zh4a41Z5.

Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. Gradient surgery for multi-task learning. *Advances in neural information processing systems*, 33: 5824–5836, 2020.

Tianhe Yu, Aviral Kumar, Yevgen Chebotar, Karol Hausman, Chelsea Finn, and Sergey Levine. How to leverage unlabeled data in offline reinforcement learning. In *International Conference on Machine Learning*, pp. 25611–25635. PMLR, 2022.

Xiaohua Zhai, Basil Mustafa, Alexander Kolesnikov, and Lucas Beyer. Sigmoid loss for language image pre-training. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 11975–11986, 2023.

Amy Zhang, Rowan Thomas McAllister, Roberto Calandra, Yarin Gal, and Sergey Levine. Learning invariant representations for reinforcement learning without reconstruction. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=-2FCwDKRREu.

Chongyi Zheng, Ruslan Salakhutdinov, and Benjamin Eysenbach. Contrastive difference predictive coding. *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=0akLDTFR9x.

Chongyi Zheng, Jens Tuyls, Joanne Peng, and Benjamin Eysenbach. Can a MISL fly? analysis and ingredients for mutual information skill learning. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=xoIeVdFO7U.

Zhiyuan Zhou, Andy Peng, Qiyang Li, Sergey Levine, and Aviral Kumar. Efficient online reinforcement learning fine-tuning need not retain offline data. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=HN0CYZbAPw.

Chuning Zhu, Xinqi Wang, Tyler Han, Simon S Du, and Abhishek Gupta. Distributional successor features enable zero-shot policy optimization. *arXiv preprint arXiv:2403.06328*, 2024.

Algorithm 1 Intention-Conditioned Flow Occupancy Model (pre-training).

```
1: Input The intention encoder $p_\phi$, the vector field $v_\theta$, the target vector field $v_{\bar{\theta}}$, the policy $\pi_\omega$, and
   the reward-free dataset $D$.
2: for each iteration do
3:     Sample a batch of $\{(s, a, s', a') \sim D\}$.
4:     Sample a batch of $\{\epsilon \sim \mathcal{N}(0, I)\}$ and a batch of $\{t \sim \textsc{Unif}([0, 1])\}$.
5:     Encode intentions $\{z \sim p_\phi(z \mid s', a')\}$ for each $(s', a')$.
       ▽ SARSA flow occupancy model loss.
6:     $s^t \leftarrow (1 - t)\epsilon + ts$
7:     $\bar{s}_f \leftarrow \text{EulerMethod}(v_{\bar{\theta}}, \epsilon, s', a', z)$, $\bar{s}_f^t \leftarrow (1 - t)z + t\bar{s}_f$.
8:     $\mathcal{L}_{\text{SARSA current flow}}(\theta, \phi) \leftarrow \mathbb{E}_{(s,a,z,t,\epsilon,s^t)}\left[\|v_\theta(t, s^t, s, a, z) - (s - z)\|_2^2\right]$.
9:     $\mathcal{L}_{\text{SARSA future flow}}(\theta, \phi) \leftarrow \mathbb{E}_{(s,a,z,t,\epsilon,\bar{s}_f^t)}\left[\|v_\theta(t, \bar{s}_f^t, s, a, z) - v_{\bar{\theta}}(t, \bar{s}_f^t, s', a', z)\|_2^2\right]$.
10:    $\mathcal{L}_{\text{SARSA flow}}(\theta, \phi) \leftarrow (1 - \gamma)\mathcal{L}_{\text{current}}(\theta, \phi) + \gamma\mathcal{L}_{\text{future}}(\theta, \phi)$.      ▷ Eq. 5
       ▽ Intention encoder loss.
11:    $\mathcal{L}_{\text{ELBO}}(\theta, \phi) \leftarrow \mathcal{L}_{\text{SARSA flow}}(\theta, \phi) + \lambda\mathbb{E}_{(s',a')}\left[D_{\text{KL}}\left(p_\phi(z \mid s', a') \,\|\, \mathcal{N}(0, I)\right)\right]$.      ▷ Eq. 4
       ▽ (Optional) Behavioral cloning loss.
12:    $\mathcal{L}_{\text{BC}}(\omega) \leftarrow -\mathbb{E}_{(s,a)}\left[\log \pi_\omega(a \mid s)\right]$.
13:    Update the vector field $\theta$ and the intention encoder $\phi$ by minimizing $\mathcal{L}_{\text{ELBO}}(\theta, \phi)$.
14:    Update the policy $\omega$ by minimizing $\mathcal{L}_{\text{BC}}(\omega)$.
15:    Update the target vector field $\bar{\theta}$ using an exponential moving average of $\theta$.
16: Return $v_\theta$, $p_\phi$, and $\pi_\omega$.
```

A Preliminaries

A.1 Value functions and the Actor-critic framework

The goal of RL is to learn a policy $\pi : \mathcal{S} \to \Delta(\mathcal{A})$ that maximizes the expected discounted return $J(\pi) = \mathbb{E}_{\tau\sim\pi(\tau)}\left[\sum_{h=0}^{\infty}\gamma^h r_h\right]$, where τ is a trajectory sampled by the policy. We will use $\beta : \mathcal{S} \to \Delta(\mathcal{A})$ to denote the behavioral policy. Given a policy π, we measure the expected discounted return starting from a state-action pair (s, a) and a state s as the (unnormalized) Q-function and the value function, respectively:

Algorithm 3 Euler method for solving the flow ODE (Eq. 9).

```
1: Input The vector field $v$ and the noise $\epsilon$. (Optional) The number of steps $T$ with default
   $T = 10$.
2: Initialize $t = 0$ and $x^t = \epsilon$
3: for each step $t = 0, 1, \cdots, T - 1$ do
4:     $x^{t+1} \leftarrow x^t + v(t/T, x^t)/T$
5: Return $\hat{x} = x^T$
```

$$Q^\pi(s, a) = \mathbb{E}_{\tau\sim\pi(\tau)}\left[\sum_{h=0}^{\infty}\gamma^h r_h \,\middle|\, s_0 = s, a_0 = a\right], \; V^\pi(s) = \mathbb{E}_{a\sim\pi(a|s)}\left[Q^\pi(s, a)\right].$$

Prior actor-critic methods (Schulman et al., 2015; 2017; Haarnoja et al., 2018; Fujimoto et al., 2018; Kumar et al., 2020; Fujimoto & Gu, 2021) typically maximize the RL objective $J(\pi)$ by *(1)* learning an estimate Q of Q^π via the temporal difference (TD) loss (policy evaluation) and then *(2)* improving the policy π by selecting actions that maximizes Q (policy improvement):

$$Q^{k+1} \leftarrow \arg\max_Q \mathbb{E}_{(s,a,r,s')\sim p^\beta(s,a,r,s'),a'\sim\pi^k(a'|s')}\left[\left(Q(s, a) - (r + \gamma Q^k(s', a'))\right)^2\right]$$
$$\pi^{k+1} \leftarrow \arg\max_\pi \mathbb{E}_{s\sim p^\beta(s),a\sim\pi(a|s)}\left[Q^{k+1}(s, a)\right],$$

where k indicates the number of updates and β is the behavioral policy representing either a replay buffer (online RL) or a fixed dataset (offline RL).

Algorithm 2 Intention-Conditioned Flow Occupancy Model (fine-tuning).

```
1: Input The intention encoder $p_\phi$, the vector field $v_\theta$, the target vector field $v_{\bar{\theta}}$, the reward predictor
   $r_\eta$, the critic $Q_\psi$, the policy $\pi_\omega$ (random initialization or initialized using $\pi_\omega$ from Alg. 1), and
   the reward-labeled dataset $D_{\text{reward}}$.
2: for each iteration do
3:     Sample a batch of $\{(s, a, r, s', a') \sim D_{\text{reward}}\}$.
4:     Sample a batch of $\{\epsilon \sim \mathcal{N}(0, I)\}$ and a batch of $\{t \sim \textsc{Unif}([0, 1])\}$.
5:     Sample prior intentions $\{z \sim p(z)\}$.
6:     Sample a batch of $\{(\epsilon^{(1)}, \cdots, \epsilon^{(N)}) \sim (\mathcal{N}(0, I), \cdots, \mathcal{N}(0, I))\}$.
       ▽ SARSA flow occupancy model loss and intention encoder loss.
7:     $\mathcal{L}_{\text{ELBO}}(\theta, \psi)$ as in Alg. 1.
       ▽ Reward predictor loss.
8:     $\mathcal{L}_{\text{Reward}}(\eta) \leftarrow \mathbb{E}_{(s,r)}\left[(r_\eta(s) - r)^2\right]$.
       ▽ Critic loss.
9:     $s_f^{(i)} \leftarrow \text{EulerMethod}(v_\theta, \epsilon^{(i)}, s, a, z)$ (Alg. 3) for each $(s, a, z, \epsilon^{(i)})$.
10:    $Q_z(s, a) \leftarrow \frac{1}{(1-\gamma)N} \sum_{i=1}^{N} r_\eta\left(s_f^{(i)}\right)$.                          ▷ Eq. 6
11:    $\mathcal{L}_{\text{Critic}}(\psi) \leftarrow \mathbb{E}_{\left(s,a,z,s_f^{(1)},\cdots,s_f^{(N)}\right)}\left[L_2^{\mu}\left(Q_z(s, a) - Q_\psi(s, a)\right)\right]$.      ▷ Eq. 7
       ▽ Actor loss.
12:    $\mathcal{L}_{\text{Actor}}(\omega) \leftarrow -\mathbb{E}_{(s,a),a^\pi \sim \pi_\omega(a^\pi|s)}\left[Q_\psi(s, a^\pi) + \alpha \log \pi_\omega(a \mid s)\right]$.      ▷ Eq. 8
13:    Update the vector field $\theta$ and the intention encoder $\phi$ by minimizing $\mathcal{L}_{\text{ELBO}}(\theta, \phi)$.
14:    Update the reward predictor $\eta$, the critic $\psi$, and the policy $\omega$ by minimizing $\mathcal{L}_{\text{Reward}}(\eta)$,
       $\mathcal{L}_{\text{Critic}}(\psi)$, and $\mathcal{L}_{\text{Actor}}(\omega)$ respectively.
15:    Update the target vector field $\bar{\theta}$ using an exponential moving average of $\theta$.
16: Return $v_\theta$, $p_\phi$, $r_\eta$, $Q_\phi$, and $\pi_\omega$.
```

A.2 Flow matching

The goal of flow matching methods is to transform a simple noise distribution (e.g., a d-dimensional standard Gaussian) into a target distribution $p_{\mathcal{X}}$ over some space $\mathcal{X} \subset \mathbb{R}^d$ that we want to approximate. Specifically, flow matching uses a time-dependent vector field $v : [0, 1] \times \mathbb{R}^d \to \mathbb{R}^d$ to construct a time-dependent diffeomorphic flow $\phi : [0, 1] \times \mathbb{R}^d \to \mathbb{R}^d$ (Lipman et al., 2023; 2024) that realizes the transformation from a single noise ϵ to a generative sample $\hat{x}$, following the ODE

$$\frac{d}{dt}\phi(t, \epsilon) = v(t, \phi(t, \epsilon)), \ \phi(0, \epsilon) = \epsilon, \ \phi(1, \epsilon) = \hat{x}. \tag{9}$$

We will use t to denote a time step for flow matching and sample the noise ϵ from a standard Gaussian distribution $\mathcal{N}(0, I)$ throughout our discussions.[4] Prior work has proposed various formulations for learning the vector field (Lipman et al., 2023; Campbell et al., 2024; Liu et al., 2023; Albergo & Vanden-Eijnden, 2023) and we adopt the simplest flow matching objective building upon optimal transport (Liu et al., 2023) and conditional flow matching (CFM) (Lipman et al., 2023),

$$\mathcal{L}_{\text{CFM}}(v) = \mathbb{E}_{\substack{t \sim \textsc{Unif}([0,1]), \\ x \sim p_{\mathcal{X}}(x), \epsilon \sim \mathcal{N}(0,I)}}\left[\|v(t, x^t) - (x - \epsilon)\|_2^2\right], \tag{10}$$

where $\textsc{Unif}([0, 1])$ is the uniform distribution over the unit interval and $x^t = tx + (1 - t)\epsilon$ is a linear interpolation between the ground-truth sample x and the Gaussian noise ϵ. Importantly, we can generate a sample from the vector field v by numerically solving the ODE (Eq. 9). We will use the Euler method (Alg. 3) as our ODE solver following prior practice (Grathwohl et al., 2019; Chen et al., 2018; Lipman et al., 2023; Liu et al., 2023; Park et al., 2025b; Frans et al., 2025).

[4]In theory, the noise can be drawn from any distribution, not necessarily limited to a Gaussian (Liu et al., 2023).

A.3 Temporal difference flows

Given a policy π, prior work (Farebrother et al., 2025) models the occupancy measure p_γ^π by optimizing the vector field $v : [0,1] \times \mathcal{S} \times \mathcal{S} \times \mathcal{A} \to \mathcal{S}$ using the following loss:

$$\mathcal{L}_{\text{TD flow}}(v) = (1-\gamma)\mathcal{L}_{\text{TD current flow}}(v) + \gamma \mathcal{L}_{\text{TD future flow}}(v) \tag{11}$$

$$\mathcal{L}_{\text{TD current flow}}(v) = \mathbb{E}_{\substack{t\sim\textsc{Unif}([0,1]),\epsilon\sim\mathcal{N}(0,I),\\ (s,a)\sim p^\beta(s,a)}} \left[\|v(t,s^t,s,a) - (s-\epsilon)\|_2^2\right]$$

$$\mathcal{L}_{\text{TD future flow}}(v) = \mathbb{E}_{\substack{t\sim\textsc{Unif}([0,1]),\epsilon\sim\mathcal{N}(0,I),\\ (s,a,s')\sim p^\beta(s,a,s'),a'\sim\pi(a'|s')}} \left[\|v(t,\bar{s}_f^t,s,a) - \bar{v}(t,\bar{s}_f^t,s',a')\|_2^2\right],$$

where $p^\beta(s,a)$ and $p^\beta(s,a,s')$ denote the joint distribution of transitions, $s^t = ts + (1-t)\epsilon$ is a linear interpolation between the current state s and the noise ϵ, and $\bar{v}$ denotes an exponential moving average of historical v over iterations (a target vector field) (Grill et al., 2020; Mnih et al., 2015; Caron et al., 2021). Of particular note is that we obtain a target future state $\bar{s}_f$ by applying the Euler method (Alg. 3) to $\bar{v}$ at the next state-action pair (s', a'), where a' is sampled from the target policy π of interest, and the noisy future state $\bar{s}_f^t = t\bar{s}_f + (1-t)\epsilon$ is a linear interpolation between this future state $\bar{s}_f$ and the noise ϵ. Intuitively, minimizing $\mathcal{L}_{\text{TD current flow}}$ reconstructs the distribution of current state s, while minimizing $\mathcal{L}_{\text{TD future flow}}$ bootstraps the vector field v at a noisy target future state $\bar{s}_f^t$, similar to Q-learning (Watkins & Dayan, 1992). Choosing the target policy π to be the same as the behavioral policy β, we obtain a SARSA (Rummery & Niranjan, 1994) variant of the loss optimizing the SARSA flows. We call the loss in Eq. 11 the TD flow loss[5] and use the SARSA variant of it to learn generative occupancy models.

B Further discussions on the problem setting

B.1 The consistency assumption on intentions

We now discuss the reason for making the consistency assumption (Assumption 1) on latent intentions. Since we use a heterogeneous behavioral policy to collect the unlabeled dataset, each unknown user intention indexed their own behavioral policy $\beta : \mathcal{S} \times \mathcal{Z} \to \Delta(\mathcal{A})$. The key observation is that the occupancy measure of each intention-conditioned behavioral policy follows its own Bellman equations (Similar to Eq. 1):

$$p_\gamma^\beta(s_f \mid s,a,z) = (1-\gamma)\delta_s(s_f) + \gamma \mathbb{E}_{\substack{s'\sim p(s'|s,a),\\ a'\sim\beta(a'|s',z)}} \left[p_\gamma^\beta(s_f \mid s',a',z)\right],$$

suggesting that the same latent z propagates through the transitions with the same underlying user intentions. Importantly, this propagation requires using a TD loss to estimate the behavioral occupancy measure, which aligns with the goal of our SARSA flow-matching losses (Eq. 5). We note that prior work (Touati & Ollivier, 2021) also adapts the same formulation of the intention-conditioned occupancy measure for zero-shot RL.

B.2 Distinctions from Meta RL and Multi-task RL

Our problem setting is conceptually similar to meta RL (Duan et al., 2016; Rakelly et al., 2019; Pong et al., 2022) with two key distinctions. First, offline meta RL methods typically have access to explicit task descriptions (e.g., a one-hot task indicator) together with task-specific datasets. These descriptions and datasets induce a clear clustering of transitions. In contrast, our method must infer this structure from a heterogeneous dataset in an unsupervised manner. Second, offline meta RL trains on reward-labeled data during the meta-training phase, where task-specific rewards provide supervision for policy learning. In contrast, during pre-training, our method learns a generative model that predicts future states from inferred intentions without using any task-specific reward signals.

Similar to the distinctions between our setting and offline meta RL problems, our method does *not* fall into the multi-task RL category (Sodhani et al., 2021; Yu et al., 2020). During pre-training, *(1)* InFOM does not have access to task descriptions or task-specific datasets, and *(2)* it does not use any supervision from task-specific reward signals. Instead, InFOM pre-trains a generative, multi-step transition model that facilitates value estimation for downstream tasks.

[5]The TD flow loss is called the TD^2-CFM loss in Farebrother et al. (2025) and we rename it for simplicity.

C Theoretical analyses

C.1 The evidence lower bound and its connection with an information bottleneck

We first derive the evidence lower bound for optimizing the latent variable model and then show its connection with an information bottleneck. Given the unlabeled dataset D, we want to maximize the likelihood of consecutive transitions (s, a, s', a') and a future state s_f sampled from the same trajectory following the behavioral joint distribution $p^\beta(s, a, s_f, s', a') = p^\beta(s)\beta(a \mid s)p^\beta_\gamma(s_f \mid s, a)p(s' \mid s, a)\beta(a' \mid s')$. We use (s', a') to encode the intention z by the encoder $p_e(z \mid s, a)$ and (s, a, s_f, z) to learn the occupancy models $q_d(s_f \mid s, a, z)$, employing an ELBO of the likelihood of the prior data:

$$
\begin{aligned}
&\mathbb{E}_{p^\beta(s,a,s_f)}\left[\log q_d(s_f \mid s, a)\right] \\
&= \mathbb{E}_{p^\beta(s,a,s_f,s',a')}\left[\log q_d(s_f \mid s, a)\right] \\
&= \mathbb{E}_{p^\beta(s,a,s_f,s',a')}\left[\log \mathbb{E}_{p(z)}\left[q_d(s_f \mid s, a, z)\right]\right] \\
&\overset{(a)}{=} \mathbb{E}_{p^\beta(s,a,s_f,s',a')}\left[\log \mathbb{E}_{p(z)}\left[q_d(s_f \mid s, a, z)\frac{p_e(z \mid s', a')}{p_e(z \mid s', a')}\right]\right] \\
&\overset{(b)}{\geq} \mathbb{E}_{p^\beta(s,a,s_f,s',a')}[\mathbb{E}_{p_e(z \mid s',a')}[\log q_d(s_f \mid s, a, z)] - D_{\text{KL}}(p_e(z \mid s', a') \parallel p(z))] \\
&\overset{(c)}{\geq} \mathbb{E}_{p^\beta(s,a,s_f,s',a')}[\mathbb{E}_{p_e(z \mid s',a')}[\log q_d(s_f \mid s, a, z)] - \lambda D_{\text{KL}}(p_e(z \mid s', a') \parallel p(z)] \\
&= \text{ELBO}(p_e, q_d),
\end{aligned}
$$

where in *(a)* we introduce the amortized variational encoder $p_e(z \mid s', a')$, in *(b)* we apply the Jensen's inequality (Durrett, 2019), and in *(c)* we introduce a coefficient $\lambda \geq 1$ to control the strength of the KL divergence regularization. In practice, we can use any $\lambda \geq 0$ because rescaling the *input* (s, a, s_f), similar to normalizing the range of images from $\{0, \cdots, 255\}$ to $[0, 1]$ in the original VAE (Kingma & Welling, 2013), preserves the ELBO. Formally, following prior work (Higgins et al., 2017), maximizing this ELBO can also be interpreted as an optimization problem that simultaneously predicts future states while penalizing the intention encoder:

$$
\max_{p_e, q_d} \mathbb{E}_{p^\beta(s,a,s_f,s',a')}[\mathbb{E}_{p_e(z \mid s',a')}[\log q_d(s_f \mid s, a, z)] \quad \text{s.t. } D_{\text{KL}}(p_e(z \mid s', a') \parallel p(z)) \leq \text{const.}.
$$

Rewriting this constrained optimization problem as the Lagrangian produces

$$
\mathbb{E}_{p^\beta(s,a,s_f,s',a')}[\mathbb{E}_{p_e(z \mid s',a')}[\log q_d(s_f \mid s, a, z)] - \lambda D_{\text{KL}}(p_e(z \mid s', a') \parallel p(z)),
$$

where we introduce a coefficient $\lambda \geq 0$ to control the strength of the KL divergence regularization.

Alternatively, the constrained optimization problem can also be cast as a variational lower bound on an information bottleneck with the chain of random variables $(S', A') \to Z \to (S, A, S_f)$ (Tishby et al., 2000; Alemi et al., 2017; Saxe et al., 2018):

$$
\begin{aligned}
&I^\beta(S, A, S_f; Z) - \lambda I^\beta(S', A'; Z) \\
&\overset{(a)}{=} I^\beta(S, A, S_f; Z) - \lambda \mathbb{E}_{p^\beta(s',a')}\left[D_{\text{KL}}(p_e(z \mid s', a') \parallel p_e(z))\right] \\
&\overset{(b)}{\geq} I^\beta(S, A, S_f; Z) - \lambda \mathbb{E}_{p^\beta(s',a')}\left[D_{\text{KL}}(p_e(z \mid s', a') \parallel p(z))\right] \\
&\overset{(c)}{\geq} \mathbb{E}_{\substack{p^\beta(s,a,s_f,s',a') \\ p_e(z \mid s',a')}}\left[\log q_d(s, a, s_f \mid z)\right] - \lambda \mathbb{E}_{p^\beta(s',a')}\left[D_{\text{KL}}(p_e(z \mid s', a') \parallel p(z))\right] + H^\beta(S, A, S_f) \\
&\overset{(d)}{\geq} \mathbb{E}_{\substack{p^\beta(s,a,s_f,s',a') \\ p_e(z \mid s',a')}}\left[\log q_d(s_f \mid s, a, z)\right] - \lambda \mathbb{E}_{p^\beta(s',a')}\left[D_{\text{KL}}(p_e(z \mid s', a') \parallel p(z))\right] + \text{const.}
\end{aligned}
$$

where in *(a)* we use the definition of $I^\beta(S', A'; Z)$ and $p_e(z)$ is the marginal distribution of latent intentions z defined as $p_e(z) = \int p^\beta(s', a')p_e(z \mid s', a')ds'da'$, in *(b)* we apply the non-negative property of the KL divergence $D_{\text{KL}}(p_e(z) \parallel p(z))$, in *(c)* we apply the standard variation lower bound of the mutual information (Barber & Agakov, 2004; Poole et al., 2019) to incorporate the decoder (occupancy models) $q_d(s, a, s_f \mid z)$, and in *(d)* we choose the variational decoder to satisfy $\log q_d(s, a, s_f \mid z) = \log p^\beta(s, a) + \log q_d(s_f \mid s, a, z)$ and consider the entropy $H^\beta(S, A, S_f)$ as a constant. Therefore, the lower bound in Eq. 3 can also be interpreted as maximizing the information bottleneck $I^\beta(S, A, S_f; Z) - \lambda I^\beta(S', A'; Z)$ with $\lambda \geq 0$.

C.2 Intutions and discussions about the implicit generalized policy improvement

The intuition for the expectile distillation loss (Eq. 7) is that the scalar Q function $Q(\cdot, \cdot)$ is a *one-step* summary of the average reward at future states sampled from the flow occupancy models, while the expectile loss serves as a "softmax" operator over the entire latent intention space. Theoretically, this expectile loss is guaranteed to converge to the maximum over $p(z)$ when $\mu \to 1$ (See Sec. 4.4 in Kostrikov et al. (2022) for details). Therefore, given an infinite amount of samples ($N \to \infty$) and an expectile $\mu \to 1$, the Q converges to the greedy value functions:

$$Q^\star(s, a) = \max_{z \sim p(z)} \frac{1}{(1-\gamma)} \mathbb{E}_{q_d(s_f \mid s, a, z)}[r(s_f)].$$

If we further assume that the flow occupancy models are optimal, i.e., $q_d^\star(s_f \mid s, a, z) = p^\beta(s_f \mid s, a, z)$, then the optimal Q corresponds to a greedy value function under the behavioral policy β:

$$Q^\star(s, a) = \max_{z \sim p(z)} Q^\beta(s, a, z).$$

Unlike Q-learning, which converges to the optimal Q-function sequentially (Watkins & Dayan, 1992; Sutton et al., 1998), the implicit GPI proposes a new policy that is strictly no worse than the set of policies that correspond to each Q_z in parallel (See Sec. 4.1 in Barreto et al. (2017) for further discussions). Unlike one-step policy improvement (Wang et al., 2018; Brandfonbrener et al., 2021; Peters & Schaal, 2007; Peters et al., 2010), implicit GPI is able to converge to the optimal policy for a downstream task, assuming that the task-specific intention has been captured during pre-training.

C.3 One-step policy improvement with flow occupancy models

The FOM + one-step PI variant performs one-step policy improvement using a flow occupancy model $q_d(s_f \mid s, a)$ that is *not* conditioned on latent intentions. This flow occupancy model captures the discounted state occupancy measure of the (average) behavioral policy. After training the flow occupancy model, FOM + one-step PI fits a Q function and extracts a behavioral-regularized policy:

$$Q \leftarrow \arg\min_Q \frac{1}{1-\gamma} \mathbb{E}_{(s,a) \sim p^{\tilde{\beta}}(s,a), s_f \sim q_d(s_f \mid s, a)}[(Q(s, a) - r(s_f))^2],$$
$$\pi \leftarrow \arg\max_\pi \mathbb{E}_{(s,a) \sim p^{\tilde{\beta}}(s,a), a^\pi \sim \pi(a^\pi \mid s)} \left[Q(s, a^\pi) + \alpha \log \pi(a \mid s)\right].$$

Intuitively, the first objective fits the behavioral Q function based on the dual definition (Eq. 2), and the second objective trains a policy to maximize this behavioral Q function, invoking one-step policy improvement. While this simple objective sometimes achieves strong performance on some benchmark tasks (Brandfonbrener et al., 2021; Eysenbach et al., 2022), it does not guarantee convergence to the optimal policy due to the use of a behavioral value function.

D Experimental details

D.1 Tasks and datasets

Our experiments use a suite of 36 state-based and 4 image-based control tasks taken from ExORL benchmarks Yarats et al. (2022) and OGBench task suite (Park et al., 2025a) (Fig. 2).

ExORL. We use 16 state-based tasks from the ExORL (Yarats et al., 2022) benchmarks based on the DeepMind Control Suite (Tassa et al., 2018). These tasks involve controlling four robots (`cheetah`, `walker`, `quadruped`, and `jaco`) to achieve different locomotion behaviors. For each domain, the specific tasks are: `cheetah` {`run`, `run backward`, `walk`, `walk backward`}, `walker` {`walk`, `run`, `stand`, `flip`}, `quadruped` {`run`, `jump`, `stand`, `walk`}, `jaco` {`reach top left`, `reach top right`, `reach bottom left`, `reach bottom right`}. For all tasks in `cheetah`, `walker`, and `quadruped`, both the episode length and the maximum return are 1000. For all tasks in `jaco`, both the episode length and the maximum return are 250. Following prior work (Park et al., 2024b), we multiply the return of `jaco` tasks by 4 to match other ExORL tasks during aggregation.

Following the prior work (Touati et al., 2023; Park et al., 2024b; Kim et al., 2024), we will use 5M unlabeled transitions collected by some exploration methods (e.g., RND (Burda et al., 2019)) for pre-training, and another 500K reward-labeled transitions collected by the same exploratory policy for fine-tuning. The fine-tuning datasets are labeled with task-specific dense rewards (Yarats et al., 2022), except in the `jaco` domains, where the reward signals are sparse.

OGBench. We use 20 state-based manipulation tasks from four domains (`cube single`, `cube double`, `scene`, and `puzzle 4x4`) in the OGBench task suite Park et al. (2025a), where the goal is to control a simulated robot arm to rearrange various objects. For each domain, the specific tasks are: `cube single` {`task 1` (pick and place cube to left), `task 2` (pick and place cube to front), `task 3` (pick and place cube to back), `task 4` (pick and place cube diagonally), `task 5` (pick and place cube off-diagonally)}, `cube double` {`task 1` (pick and place one cube), `task 2` (pick and place two cubes to right), `task 3` (pick and place two cubes off-diagonally), `task 4` (swap cubes), `task 5` (stack cubes)}, `scene` {`task 1` (open drawer and window), `task 2` (close and lock drawer and window), `task 3` (open drawer, close window, and pick and place cube to right), `task 4` (put cube in drawer), `task 5` (fetch cube from drawer and close window)}, `puzzle 4x4` {`task 1` (all red to all blue), `task 2` (all blue to central red), `task 3` (two blue to mix), `task 4` (central red to all red), `task 5` (mix to all red)}. Note that some of these tasks, e.g., `cube double task 5` (stack cubes) and `scene task 4` (put cube in drawer), involve interacting with the environment in a specific order and thus require long-horizon temporal reasoning. For all tasks in `cube single`, `cube double`, and `scene`, the maximum episode length is 400. For all tasks in `puzzle 4x4`, the maximum episode length is 800. We also use 4 image-based tasks in the OGBench task suite. Specifically, we consider `visual cube single task 1`, `visual cube double task 1`, `visual scene task 1`, and `visual puzzle 4x4 task 1` from each domain respectively. The observations are $64 \times 64 \times 3$ RGB images. These tasks are challenging because the agent needs to reason from pixels directly. All the manipulation tasks from OGBench are originally designed for evaluating goal-conditioned RL algorithms (Park et al., 2025a).

For both state-based and image-based tasks from OGBench, we will use 1M unlabeled transitions collected by a non-Markovian expert policy with temporally correlated noise (the `play` datasets) for pre-training, and another 500K reward-labeled transitions collected by the same noisy expert policy for fine-tuning. Unlike the ExORL benchmarks, the fine-tuning datasets for OGBench tasks are relabeled with *semi-sparse* rewards (Park et al., 2025b), providing less supervision for the algorithm.

D.2 Baselines

We compare InFOM with eight baselines across five categories of prior methods, focusing on different strategies for pre-training and fine-tuning in RL. First, implicit Q-Learning (IQL) (Kostrikov et al., 2022) and revisited behavior-regularized actor-critic (ReBRAC) (Tarasov et al., 2023a) are state-of-the-art offline RL algorithms based on the standard actor-critic framework (Appendix A.1). Second, we compare to a variant of ReBRAC learning on top of representations pre-trained on the unlabeled datasets. We chose an off-the-shelf self-supervised learning objective in vision tasks called self-distillation with no labels (DINO) (Caron et al., 2021) as our representation learning loss and name the resulting baseline DINO + ReBRAC. Third, our next baseline, model-based policy optimization (MBPO) (Janner et al., 2019), pre-trains a one-step model to predict transitions in the environment, similar to the next token prediction in language models (Radford et al., 2018). The one-step model is then used to augment the datasets for downstream policy optimization. We will again use ReBRAC to extract the policy (MBPO + ReBRAC). Fourth, we also include comparisons against the InfoNCE variant of contrastive RL (Eysenbach et al., 2019) and temporal difference InfoNCE (Zheng et al., 2024), which pre-train the discounted state occupancy measure using Monte Carlo or temporal difference contrastive losses. While our method fits generative occupancy models, These two approaches predict the ratio of occupancy measures over some marginal densities serving

Table 1: Common hyperparameters for our method and the baselines.

Hyperparameter	Value
learning rate	3×10^{-4}
optimizer	Adam (Kingma, 2014)
pre-training gradient steps	1×10^6 for state-based tasks, 2.5×10^5 for image-based tasks
fine-tuning gradient steps	5×10^5 for state-based tasks, 1×10^5 for image-based tasks
batch size	256
MLP hidden layer sizes	$(512, 512, 512, 512)$
MLP activation function	GELU (Hendrycks & Gimpel, 2016)
discount factor γ	0.99
target network update coefficient	5×10^{-3}
double Q aggregation	min
policy update frequency in fine-tuning	$1/4$
image encoder	small IMPALA encoder (Espeholt et al., 2018; Park et al., 2025b)
image augmentation method	random cropping
image augmentation probability	1.0 for DINO + ReBRAC, 0.5 for all other methods
image frame stack	3

as the discriminative counterparts. After pre-training the ratio predictors, importance sampling is required to recover the Q function (CRL + IS & TD InfoNCE + IS) (Mazoure et al., 2023; Zheng et al., 2024), enabling policy maximization. Fifth, our final set of baselines are prior unsupervised RL methods that pre-train a set of latent intentions and intention-conditioned policies using forward-backward representations (Touati & Ollivier, 2021) or a Hilbert space (Park et al., 2024b). Given a downstream task, these methods first infer the corresponding intention in a zero-shot manner and then fine-tune the policy using offline RL (Kim et al., 2024), differing from the implicit GPI as in our method. We will use IQL as the fine-tuning algorithm and call the resulting methods FB + IQL and HILP + IQL. For image-based tasks, we selectively compare to four baselines: ReBRAC, CRL + IS, DINO + ReBRAC, and FB + IQL.

D.3 Evaluation protocols

We compare the performance of InFOM against the eight baselines (Sec. 5.1) after first pre-training each method for 1M gradient steps (250K gradient steps for image-based tasks) and then fine-tuning for 500K gradient steps (100K gradient steps for image-based tasks). We measure the episode return for tasks from ExORL benchmarks and the success rate for tasks from the OGBench task suite. For OGBench tasks, the algorithms still use the semi-sparse reward instead of the success rate for training. Following prior practice (Park et al., 2025b; Tarasov et al., 2023b), we do *not* report the best performance during fine-tuning and report the evaluation results averaged over 400K, 450K, and 500K gradient steps instead. For image-based tasks, we report the evaluation results averaged over 50K, 75K, and 100K gradient steps during fine-tuning. For evaluating the performance of different methods throughout the entire fine-tuning process, we defer the details to specific figures (e.g., Fig. 10 & 9).

D.4 Implementations and hyperparameters

In this section, we discuss the implementation details and hyperparameters for InFOM and the eight baselines. Whenever possible, we use the same set of hyperparameters for all methods (Table 1) across all tasks, including learning rate, network architecture, batch size, image encoder, etc. Of particular note is that we use asynchronous policy training (Zhou et al., 2025), where we update the policy 4 times less frequently than other models during fine-tuning. For specific hyperparameters of each method, we tune them on the following tasks from each domain and use one set of hyperparameters for every task in that domain. For image-based tasks, we tune hyperparameters for each task individually.

- `cheetah`: `cheetah run`
- `walker`: `walker walk`
- `quadruped`: `quadruped jump`

Table 2: **Hyperparameters for InFOM.** See Appendix D.4 for descriptions of each hyperparameter.

Hyperparameter	**Value**
latent intention dimension d	See Table 3
number of steps for the Euler method T	10
number of future states N	16
normalize the Q loss term in $\mathcal{L}(\pi)$ (Eq. 8)	No
expectile μ	See Table 3
KL divergence regularization coefficient λ	See Table 3
behavioral cloning regurlaization coefficient α	See Table 3

Table 3: **Domain-specific hyperparameters for our method and the baselines.** We individually tune these hyperparameters for each domain and use the same set of hyperparameters for tasks in the same domain. See Appendix D.4 for tasks used to tune these hyperparameters and descriptions of each hyperparameter. "-" indicates that the hyperparameter does not exist.

	InFOM (Ours)				IQL	ReBRAC		DINO + ReBRAC	MBPO + ReBRAC		CRL + IS	TD InfoNCE + IS	FB + IQL		HILP + IQL
Domain or Task	d	μ	λ	α	α	α_{actor}	α_{critic}	κ_{student}	$N_{\text{imaginary}}$	$H_{\text{imaginary}}$	α	α	α_{repr}	α_{AWR}	α
`cheetah`	128	0.9	0.05	0.3	1	0.1	0.1	0.1	128	1	0.03	0.003	1	1	1
`walker`	512	0.9	0.1	0.3	1	10	0.1	0.1	128	1	0.03	0.03	1	10	10
`quadruped`	512	0.9	0.005	0.3	10	1	1	0.1	128	1	0.03	0.03	10	1	10
`jaco`	512	0.9	0.2	0.1	0.1	0.1	0.1	0.1	128	1	0.003	0.03	1	1	1
`cube single`	512	0.95	0.05	30	1	1	1	0.04	256	2	30	30	10	1	1
`cube double`	128	0.9	0.025	30	1	1	1	0.04	256	2	30	30	1	10	1
`scene`	128	0.99	0.2	300	1	1	1	0.1	256	2	3	3	10	10	1
`puzzle 4x4`	128	0.95	0.1	300	10	0.1	0.1	0.1	256	2	3	3	10	10	1
`visual cube single task 1`	512	0.95	0.025	30	-	1	0	0.1	-	-	30	-	10	1	-
`visual cube double task 1`	128	0.9	0.01	30	-	0.1	0	0.1	-	-	30	-	10	1	-
`visual scene task 1`	128	0.99	0.1	300	-	0.1	0.01	0.1	-	-	3	-	10	10	-
`visual puzzle 4x4 task 1`	128	0.95	0.1	300	-	0.1	0.01	0.1	-	-	3	-	10	10	-

- `jaco`: `jaco reach top left`
- `cube single`: `cube single task 2`
- `cube double`: `cube double task 2`
- `scene`: `scene task 2`
- `puzzle 4x4`: `puzzle 4x4 task 4`

InFOM. InFOM consists of two main components for pre-training: the intention encoder and the flow occupancy models. First, we use a Gaussian distribution conditioned on the next state-action pair as the intention encoding distribution. Following prior work (Kingma & Welling, 2013; Alemi et al., 2017), we model the intention encoder as a multilayer perceptron (MLP) that takes the next state-action pair (s', a') as input and outputs two heads representing the mean and the (log) standard deviation of the Gaussian. We apply layer normalization to the intention encoder to stabilize optimization. We use the reparameterization trick (Kingma & Welling, 2013) to backpropagate the gradients from the flow-matching loss and the KL divergence regularization (Eq. 4) into the intention encoder. Our initial experiments suggest that the dimension of the latent intention space d is an important hyperparameter, and we sweep over $\{64, 128, 256, 512\}$ and find that $d = 512$ is sufficient for most ExORL tasks and $d = 128$ is generally good enough for all OGBench tasks. For the coefficient of the KL divergence regularization λ, we sweep over $\{2.0, 1.0, 0.2, 0.1, 0.05, 0.025, 0.01, 0.005\}$ to find the best λ for each domain. Second, we use flow-matching vector fields to model the flow occupancy models. The vector field is an MLP that takes in a noisy future state s_f^t, a state-action pair (s, a), and a latent intention z, and outputs the vector field with the same dimension as the state. We apply layer normalization to the vector field to stabilize optimization. As mentioned in Sec. 3, we use flow-matching objectives based on optimal transport (linear path) and sample the time step t from the uniform distribution over the unit interval. Following prior work (Park et al., 2025b), we use a fixed $T = 10$ steps (step size = 0.1) for the Euler method and do not apply a sinusoidal embedding for the time. To make a fair comparison with other baselines, we also pre-train a behavioral cloning policy that serves as initialization for fine-tuning.

For fine-tuning, InFOM learns three components: the reward predictor, the critic, and the policy, while fine-tuning the intention encoder and the flow occupancy models. The reward predictor is an

MLP that predicts the scalar reward of a state trained using mean squared error. We apply layer normalization to the reward predictor to stabilize learning. The critic is an MLP that predicts double Q values (Van Hasselt et al., 2016; Fujimoto et al., 2018) of a state-action pair, without conditioning on the latent intention. We apply layer normalization to the critic to stabilize learning. We train the critic using the expectile distillation loss (Eq. 7) and sweep the expectile over $\{0.9, 0.95, 0.99\}$ to find the best μ for each domain. We use $N = 16$ future states sampled from the flow occupancy models to compute the average reward, which we find to be sufficient. We use the minimum of the double Q predictions to prevent overestimation. The policy is an MLP that outputs a Gaussian distribution with a unit standard deviation. In our initial experiments, we find that the behavioral cloning coefficient α in Eq. 8 is important, and we sweep over $\{300, 30, 3, 0.3\}$ to find the best α for each domain. Following prior practice (Park et al., 2025b), we do not normalize the Q loss term in the actor loss $\mathcal{L}(\pi)$ (Eq. 8) as in Fujimoto & Gu (2021). Other choices of the policy network include the diffusion model (Ren et al., 2025; Wang et al., 2023) and the flow-matching model (Park et al., 2025b), and we leave investigating these policy networks to future work.

For image-based tasks, following prior work (Park et al., 2025b), we use a smaller variant of the IMPALA encoder (Espeholt et al., 2018) and apply random cropping augmentation with a probability of 0.5. We also apply frame stacking with three images. Table 2 and Table 3 summarize the hyperparameters for InFOM.

IQL and ReBRAC. We reuse the IQL (Kostrikov et al., 2022) implementation and the ReBRAC (Tarasov et al., 2023a) implementation from Park et al. (2025b). Since learning a critic requires reward-labeled datasets or relabeling rewards for unlabeled datasets (Yu et al., 2022), we simply pre-train a behavioral cloning policy. During the fine-tuning, we use the behavioral cloning policy as initialization and train a critic from scratch using the TD error (Kostrikov et al., 2022; Fujimoto & Gu, 2021; Tarasov et al., 2023a). Following prior work (Park et al., 2025b), we use the same expectile value 0.9 for IQL on all tasks, and sweep over $\{100, 10, 1, 0.1, 0.01\}$ to find the best AWR inverse temperature α for each domain. For ReBRAC, we tune the behavioral cloning (BC) regularization coefficients for the actor and the critic separately. We use the range $\{100, 10, 1, 0.1\}$ to search for the best actor BC coefficient α_{actor} and use the range $\{100, 10, 1, 0.1, 0\}$ to search for the best critic BC coefficient α_{critic}. We use the default values for other hyperparameters following the implementation from Park et al. (2025b). See Table 3 for domain-specific hyperparameters.

DINO + ReBRAC. We implement DINO on top of ReBRAC. DINO (Caron et al., 2021) learns a state encoder using two augmentations of the same state. For state-based tasks, the state encoder is an MLP that outputs representations. We apply two clipped Gaussian noises centered at zero to the same state to obtain those augmentations. The standard deviation of the Gaussian noise is set to 0.2, and we clip the noise into $[-0.2, 0.2]$ on all domains. For image-based tasks, the state encoder is the small IMPALA encoder that also outputs representations. We apply two different random cropings to the same image observation to obtain those augmentations. We sweep over $\{0.01, 0.04, 0.1, 0.4\}$ for the temperature for student representations κ_{student} and use a fixed temperature 0.04 for teacher representations on all domains. We use a representation space with 512 dimensions. We update the target representation centroid with a fixed ratio 0.1. During pre-training, we learns the DINO representations along with a behavioral cloning policy. During fine-tuning, we learn the actor and the critic using ReBRAC on top of DINO representations, while continuing to fine-tune those DINO representations. We use the same BC coefficients α_{actor} and α_{critic} as in ReBRAC. For image-based tasks, we apply random cropping to the same image twice with a probability of 1.0 and use those two augmentations to compute the teacher and the student representations. See Table 3 for domain-specific temperatures for student representations.

MBPO + ReBRAC. We implement MBPO (Janner et al., 2019) on top of ReBRAC and only consider this baseline for state-based tasks. MBPO learns a one-step transition MLP to predict the residual between the next state s' and the current state s conditioned on the current state-action pair (s, a). We pre-train the one-step model with a behavioral cloning policy. During fine-tuning, we use the model with a learned reward predictor to collect imaginary rollouts. We *only* use these imaginary rollouts to learn the actor and the critic. We sweep over $\{64, 128, 256\}$ for the number of imaginary rollouts to collect for each gradient step $N_{\text{imaginary}}$ and sweep over $\{1, 2, 4\}$ for the number of steps in each rollout $H_{\text{imaginary}}$. We use the same BC coefficient as in ReBRAC. See Table 3 for the domain-specific number of imaginary rollouts and number of steps in each rollout.

CRL + IS and TD InfoNCE + IS. We mostly reuse the CRL (Eysenbach et al., 2022) implementation based on the InfoNCE loss from Park et al. (2025a) and adapt it to our setting by adding the important sampling component. We implement TD InfoNCE by adapting the official implementations (Zheng et al., 2024). For both methods, we pre-train the classifiers that predict the ratio between the occupancy measures and the marginal densities over future states with a behavioral cloning policy. We use the SARSA variant of TD InfoNCE during pre-training. After pre-training the classifiers, we learn a reward predictor and apply importance sampling weights predicted by the classifiers to a set of future states sampled from the fine-tuning datasets to estimate Q. This Q estimation then drives policy optimization. We use a single future state from the fine-tuning dataset to construct the importance sampling estimation, which is sufficient. We use 512-dimensional contrastive representations. We sweep over $\{300, 30, 3, 0.3, 0.03\}$ for the BC coefficient α (Table 3).

FB + IQL and HILP + IQL. We implement FB (Touati & Ollivier, 2021) and HILP (Park et al., 2024b) by adapting the FB implementation from Jeen et al. (2024) and the HILP implementation from Kim et al. (2024). During pre-training, for FB, we pre-train the forward-backward representations and the intention-conditioned policies in an actor-critic manner. We use a coefficient 1 for the orthonormality regularization of the backward representations. We use 512-dimensional forward-backward representations. We sample the latent intentions for pre-training from either a standard Gaussian distribution (with probability 0.5) or the backward representations for a batch of states (with probability 0.5), normalizing those latent intentions to length $\sqrt{512}$. We sweep over $\{100, 10, 1, 0.1\}$ for the BC coefficient α_{repr}. For HILP, we pre-train the Hilbert representations ϕ and Hilbert foundation policies using an actor-critic framework as well. We use implicit value learning to learn the Hilbert representations following implementations from Park et al. (2024a; 2025a). We set the expectile to 0.9 for all domains. We sweep over $\{100, 10, 1, 0.1\}$ to find the best AWR inverse temperature α. We also use a 512-dimensional Hilbert representation space. To construct the intrinsic rewards, we first sample the latent intention z from a standard Gaussian, normalizing them to length $\sqrt{512}$, and then use the representation of the next state $\phi(s')$ and the representation of the current state $\phi(s)$ to compute the intrinsic reward $(\phi(s') - \phi(s))^\top z$.

During fine-tuning, we first infer a task-specific backward representation or a Hilbert representation using a small amount of transitions (10K) from the fine-tuning datasets, and then invoke IQL to learn the critic and the actor using downstream rewards conditioned on the inferred representations. For FB, we sweep over $\{100, 10, 1, 0.1\}$ for the AWR inverse temperature α_{AWR} for IQL. For HILP, we reuse the same AWR inverse temperature in representation learning for IQL. See Table 3 for domain-specific BC coefficients and AWR inverse temperatures.

E Additional visualizations of latent intentions

We include additional visualization of latent intentions on `quadruped-jump` in Fig. 6.

F Additional Experiments

F.1 Evaluation on robotics benchmarks

To further study the pre-training and fine-tuning effects of our method on realistic datasets. Specifically, we choose the RT-1 dataset (Brohan et al., 2022), which contains 73499 episodes of transitions. This dataset was collected by commanding a Google robot to pick, place, and move 17 objects in the Google micro-kitchens, covering a diverse set of intentions. Since collecting distinct robotics datasets for pre-training and fine-tuning is difficult, we use the entire dataset as both the reward-free pre-training dataset and the reward-labeled fine-tuning dataset. For the evaluation task, we use `google robot pick coke can` from the SimplerEnv (Li et al., 2024), which contains a suite of simulation tasks that efficiently and informatively complement real-world evaluations of the Google robot.

We compare against two baselines from our experiments (ReBRAC and DINO + ReBRAC) due to computational constraints, and also include a behavioral cloning (BC) baseline for reference. Our initial experiments indicate that all the algorithms (except DINO + ReBRAC) perform poorly when trained end-to-end from pixels directly. Following prior practice in latent flow matching (Rombach

Table 4: **Evaluation on ExORL and OGBench benchmarks.** Following OGBench (Park et al., 2025a), we bold values at and above 95% of the best performance for each task.

Task	InFOM (Ours)	IQL	ReBRAC	DINO + ReBRAC	MBPO + ReBRAC	CRL + IS	TD InfoNCE + IS	FB + IQL	HILP + IQL
`cheetah run`	97.6 ± 7.8	80.0 ± 8.4	97.2 ± 12.9	87.2 ± 8.6	$\mathbf{104.7} \pm 2.4$	73.3 ± 6.7	68.2 ± 8.9	83.3 ± 10.9	90.3 ± 1.9
`cheetah run backward`	$\mathbf{104.7} \pm 7.3$	77.0 ± 12.6	84.9 ± 3.7	67.1 ± 6.4	87.0 ± 4.8	74.7 ± 8.1	74.3 ± 17.1	67.3 ± 7.0	64.4 ± 6.4
`cheetah walk`	254.8 ± 158.6	357.9 ± 16.4	$\mathbf{443.4} \pm 15.3$	383.5 ± 10.3	$\mathbf{447.4} \pm 12.7$	327.4 ± 38.7	336.7 ± 22.1	346.5 ± 24.3	366.8 ± 6.9
`cheetah walk backward`	251.8 ± 116.9	303.7 ± 12.6	$\mathbf{403.0} \pm 16.1$	318.4 ± 23.0	$\mathbf{398.6} \pm 16.0$	330.2 ± 8.5	326.3 ± 45.1	298.0 ± 22.8	318.1 ± 11.4
`walker walk`	$\mathbf{467.3} \pm 82.1$	208.6 ± 3.7	208.1 ± 5.8	228.0 ± 3.7	327.6 ± 4.5	213.3 ± 7.8	212.2 ± 13.2	225.3 ± 6.7	225.4 ± 3.7
`walker run`	$\mathbf{116.3} \pm 15.3$	92.4 ± 0.6	97.8 ± 1.2	98.5 ± 1.0	107.6 ± 1.2	91.5 ± 3.2	91.0 ± 3.7	97.4 ± 1.2	97.4 ± 2.2
`walker stand`	$\mathbf{581.2} \pm 72.1$	409.1 ± 2.3	460.6 ± 1.1	453.0 ± 3.1	458.1 ± 2.5	409.0 ± 7.5	397.2 ± 6.0	446.8 ± 7.1	443.3 ± 3.8
`walker flip`	$\mathbf{358.8} \pm 10.3$	260.3 ± 2.8	$\mathbf{344.6} \pm 2.7$	320.3 ± 4.3	$\mathbf{341.8} \pm 3.7$	255.0 ± 8.0	231.6 ± 6.9	287.0 ± 3.1	280.7 ± 5.4
`quadruped run`	341.8 ± 41.2	358.0 ± 6.2	343.0 ± 2.6	344.7 ± 2.9	$\mathbf{395.1} \pm 2.6$	323.4 ± 2.9	222.1 ± 39.7	367.0 ± 3.8	371.1 ± 11.5
`quadruped jump`	626.0 ± 6.8	628.5 ± 7.8	605.2 ± 7.8	573.0 ± 9.6	$\mathbf{666.9} \pm 3.4$	576.7 ± 13.7	421.4 ± 93.4	$\mathbf{639.4} \pm 8.9$	626.5 ± 14.5
`quadruped stand`	$\mathbf{718.3} \pm 18.7$	$\mathbf{714.2} \pm 9.8$	688.6 ± 5.0	663.2 ± 8.3	$\mathbf{703.7} \pm 3.6$	653.1 ± 8.4	457.1 ± 47.7	$\mathbf{728.9} \pm 11.5$	$\mathbf{715.6} \pm 13.9$
`quadruped walk`	360.7 ± 7.9	$\mathbf{375.1} \pm 3.7$	343.5 ± 7.1	$\mathbf{391.4} \pm 7.2$	$\mathbf{390.0} \pm 5.7$	309.6 ± 9.6	243.1 ± 29.2	$\mathbf{388.9} \pm 7.0$	$\mathbf{393.4} \pm 3.4$
`jaco reach top left`	$\mathbf{742.5} \pm 43.7$	74.7 ± 19.6	59.0 ± 4.9	17.5 ± 3.8	60.1 ± 6.2	29.1 ± 4.7	31.5 ± 3.0	25.0 ± 11.4	40.4 ± 11.5
`jaco reach top right`	$\mathbf{687.5} \pm 46.7$	40.6 ± 14.0	38.0 ± 13.1	11.0 ± 4.1	52.5 ± 10.8	21.4 ± 6.5	25.5 ± 10.3	16.2 ± 3.2	25.1 ± 9.6
`jaco reach bottom left`	$\mathbf{746.7} \pm 12.6$	77.1 ± 12.5	44.5 ± 4.0	13.7 ± 2.8	43.4 ± 4.6	19.8 ± 8.8	26.6 ± 5.9	19.8 ± 4.0	27.8 ± 4.6
`jaco reach bottom right`	$\mathbf{733.0} \pm 19.6$	78.7 ± 19.1	41.4 ± 5.0	8.3 ± 2.8	34.0 ± 6.0	19.6 ± 2.0	25.4 ± 5.7	12.4 ± 2.7	24.7 ± 3.9
`cube single task 1`	$\mathbf{92.5} \pm 4.0$	53.0 ± 8.7	67.3 ± 14.2	1.8 ± 1.0	77.8 ± 11.7	10.1 ± 2.7	13.8 ± 3.8	17.7 ± 8.8	32.9 ± 9.2
`cube single task 2`	78.4 ± 12.3	51.7 ± 15.1	$\mathbf{93.7} \pm 3.5$	1.2 ± 0.6	$\mathbf{94.2} \pm 2.0$	3.7 ± 2.8	8.5 ± 5.6	16.7 ± 8.6	26.5 ± 15.4
`cube single task 3`	56.4 ± 36.9	41.5 ± 5.3	$\mathbf{94.8} \pm 0.8$	1.5 ± 1.4	$\mathbf{93.1} \pm 4.7$	12.5 ± 3.2	11.7 ± 7.4	16.0 ± 12.2	35.5 ± 14.7
`cube single task 4`	$\mathbf{91.5} \pm 14.2$	42.2 ± 8.3	$\mathbf{89.5} \pm 3.6$	0.5 ± 1.0	$\mathbf{88.7} \pm 4.7$	1.7 ± 1.7	3.3 ± 3.0	18.7 ± 9.9	36.4 ± 14.9
`cube single task 5`	70.0 ± 39.1	33.7 ± 12.9	83.3 ± 6.8	0.5 ± 0.6	$\mathbf{87.8} \pm 2.7$	4.3 ± 2.2	4.0 ± 3.2	14.2 ± 12.0	18.5 ± 5.6
`cube double task 1`	$\mathbf{29.3} \pm 10.5$	17.8 ± 9.6	2.2 ± 1.7	0.0 ± 0.0	2.7 ± 1.1	4.1 ± 1.9	6.7 ± 2.7	0.2 ± 0.3	0.7 ± 1.1
`cube double task 2`	$\mathbf{12.5} \pm 10.7$	1.3 ± 1.2	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0
`cube double task 3`	$\mathbf{11.6} \pm 8.3$	0.3 ± 0.4	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0
`cube double task 4`	$\mathbf{0.3} \pm 0.4$	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0
`cube double task 5`	$\mathbf{2.8} \pm 4.6$	1.5 ± 1.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.3 ± 0.7	0.2 ± 0.3	0.0 ± 0.0	0.0 ± 0.0
`scene task 1`	$\mathbf{97.8} \pm 1.0$	66.5 ± 13.1	47.7 ± 7.2	26.7 ± 4.3	35.3 ± 7.7	17.5 ± 5.1	21.0 ± 4.3	12.3 ± 11.3	8.8 ± 3.0
`scene task 2`	$\mathbf{15.6} \pm 3.4$	2.5 ± 1.5	7.8 ± 4.9	1.3 ± 0.0	5.6 ± 5.6	2.3 ± 0.7	1.7 ± 1.3	1.5 ± 1.8	1.2 ± 1.7
`scene task 3`	$\mathbf{43.5} \pm 2.8$	0.7 ± 0.5	1.7 ± 1.1	0.2 ± 0.3	2.4 ± 0.8	0.8 ± 0.3	0.5 ± 1.0	0.0 ± 0.0	0.0 ± 0.0
`scene task 4`	1.0 ± 0.7	0.2 ± 0.3	$\mathbf{2.8} \pm 0.8$	0.2 ± 0.3	2.0 ± 1.3	1.2 ± 1.4	0.7 ± 1.3	0.2 ± 0.3	0.0 ± 0.0
`scene task 5`	$\mathbf{0.3} \pm 0.4$	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	$\mathbf{0.3} \pm 0.1$	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0
`puzzle 4x4 task 1`	$\mathbf{24.2} \pm 14.4$	2.3 ± 2.3	12.8 ± 3.1	0.3 ± 0.7	16.9 ± 1.4	0.0 ± 0.0	0.0 ± 0.0	0.2 ± 0.3	0.3 ± 0.6
`puzzle 4x4 task 2`	$\mathbf{14.5} \pm 9.4$	0.5 ± 0.6	0.5 ± 0.6	0.0 ± 0.0	0.2 ± 0.4	0.3 ± 0.4	0.0 ± 0.0	0.2 ± 0.3	0.4 ± 0.6
`puzzle 4x4 task 3`	$\mathbf{26.3} \pm 13.4$	1.0 ± 0.9	5.0 ± 2.7	0.0 ± 0.0	5.1 ± 2.8	0.3 ± 0.4	0.0 ± 0.0	0.2 ± 0.3	0.1 ± 0.3
`puzzle 4x4 task 4`	$\mathbf{12.0} \pm 7.1$	0.3 ± 0.7	0.8 ± 0.8	0.0 ± 0.0	0.4 ± 0.4	0.0 ± 0.0	0.0 ± 0.0	0.2 ± 0.3	0.1 ± 0.3
`puzzle 4x4 task 5`	$\mathbf{12.3} \pm 6.2$	0.7 ± 0.8	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.5 ± 0.6	0.0 ± 0.0	0.0 ± 0.0	0.1 ± 0.3
`visual cube single task 1`	$\mathbf{52.1} \pm 20.8$	-	10.6 ± 7.2	15.3 ± 14.6	-	12.0 ± 5.6	-	31.0 ± 15.0	-
`visual cube double task 1`	$\mathbf{11.2} \pm 9.2$	-	0.0 ± 0.0	5.0 ± 2.0	-	5.0 ± 3.6	-	1.3 ± 1.5	-
`visual scene task 1`	$\mathbf{72.4} \pm 17.7$	-	32.0 ± 13.0	26.0 ± 17.2	-	9.0 ± 6.6	-	$\mathbf{74.7} \pm 22.2$	-
`visual puzzle 4x4 task 1`	$\mathbf{6.0} \pm 3.2$	-	0.0 ± 0.0	0.0 ± 0.0	-	0.0 ± 0.0	-	0.0 ± 0.0	-

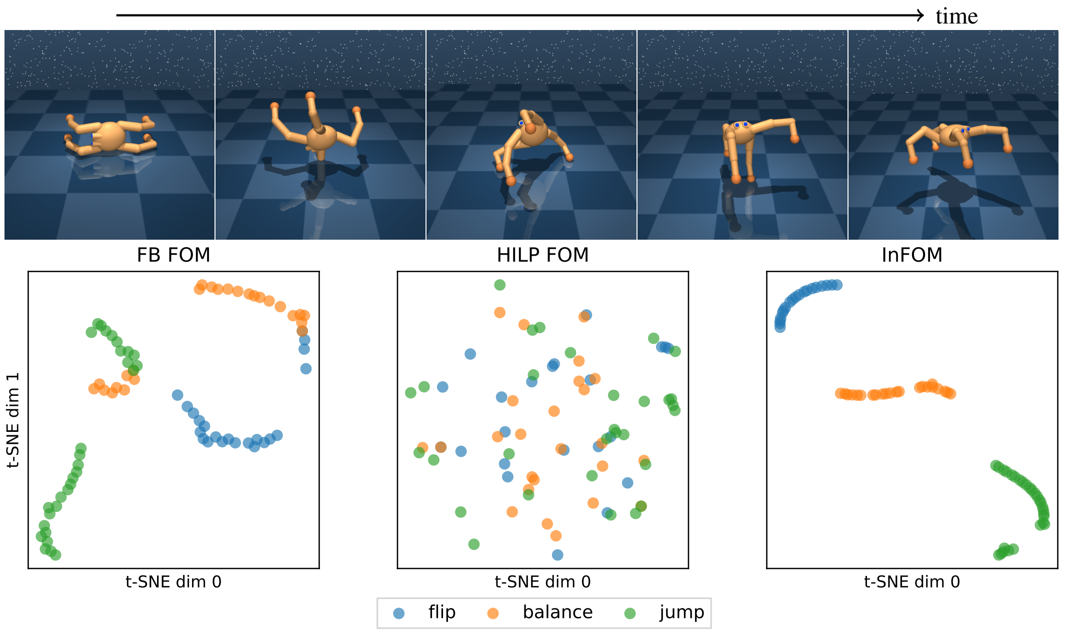
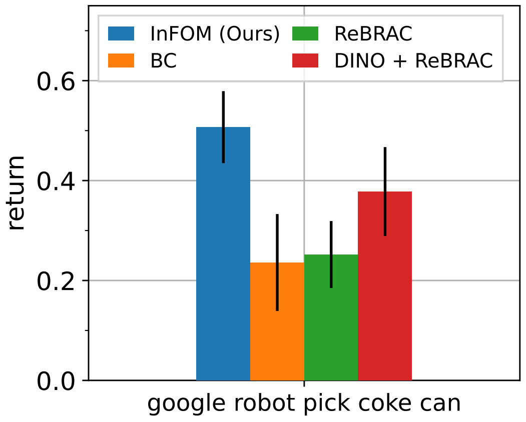


Figure 6: Visualization of latent intentions on `quadruped-jump`.

et al., 2022; Dao et al., 2023), we therefore pre-train a β-VAE (Higgins et al., 2017) to encode images into a latent embedding space and then learn algorithms on top of those embeddings. For DINO + ReBRAC, we directly use the image representations learned by DINO to train the actor and the critic. We report means and standard deviations of success rates over 4 random seeds.



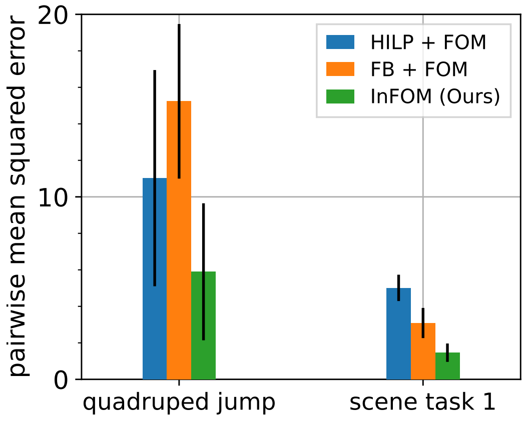


Figure 7: **Evaluation on robotics datasets.** InFOM outperforms the best baseline by 34% when trained on top of embeddings from a fixed image encoder. See Appendix F.1 for details.

Figure 8: **Comparison to prior intention encoding mechanisms after pre-training.** We compare InFOM to prior intention encoding mechanisms based on unsupervised skill discovery (HILP (Park et al., 2024b)) or successor feature learning (FB (Touati & Ollivier, 2021)) after pre-training. FB + FOM is equivalent to TD flows with GPI in Farebrother et al. (2025). InFOM achieves lower prediction errors on both tasks.

Results in Fig. 7 suggest that InFOM outperforms the best baseline by 34% when trained on top of embeddings from a fixed image encoder, indicating that our method can effectively fine-tune on challenging, realistic datasets with overlapping intentions.

F.2 Variational intention inference is simple and performant

We now conduct experiments ablating a key component in our method: the variational intention encoder. To investigate whether this framework induces a simple and performant way to infer diverse user intentions from an unlabeled dataset, we compare it to various intention encoding mechanisms proposed by prior methods. Specifically, we consider replacing the variational intention encoder with either *(1)* a set of Hilbert representations and Hilbert foundation policies (Park et al., 2024b) (HILP + FOM) or *(2)* a set of forward-backward representations and representation-conditioned policies (Touati & Ollivier, 2021) (FB + FOM), and then pre-training the flow occupancy models conditioned on these two sets of representations. Note that FB + FOM is equivalent to TD flows with GPI in Farebrother et al. (2025).

We first compare the future state predictions from InFOM against HILP + FOM and FB + FOM on two ExORL tasks (`quadruped jump` and `scene task 1`) after pre-training. Specifically, we compute the pairwise mean squared error (MSE) between predicted future states and ground-truth future states along a trajectory. We first sample 100 trajectories from the pre-training datasets, and then, for each trajectory, we sample 400 future states from InFOM and the two baselines starting from the same initial (s, a) pair. We compute the pairwise MSE between each sampled future state and the corresponding sequence of ground-truth future states within the same trajectory. The prediction error is reported as the pairwise MSE averaged over all transitions in the 100 trajectories and the 400 sampled future states. Results in Fig. 8 show that InFOM achieves lower prediction errors than two FOM baselines.

We then compare the performance of InFOM against HILP + FOM and FB + FOM after fine-tuning. We choose two tasks in the ExORL benchmarks (`walker flip` and `quadruped jump`) and another two tasks taken from the OGBench benchmarks (`cube double task 1` and `scene task 1`), following the same evaluation protocols as in Appendix D.3. Results in Fig. 9 indicate that InFOM can outperform prior intention encoding methods on 3 of 4 tasks, while being simpler. Both HILP and FB capture intentions with full unsupervised RL objectives based on an actor-critic backbone. In contrast, we capture intentions by simply training an intention encoder based on a latent

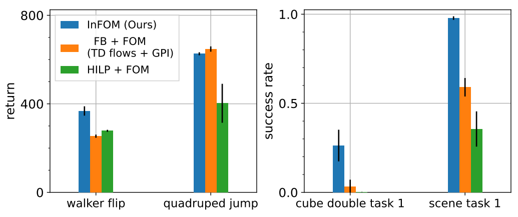


Figure 9: **Comparison to prior intention encoding mechanisms after fine-tuning.** We compare InFOM to prior intention encoding mechanisms based after fine-tuning. We observe that InFOM outperforms prior methods on 3 out of the 4 tasks.

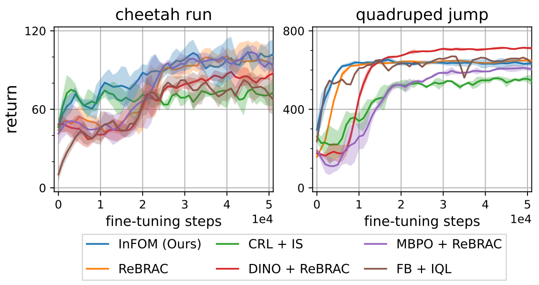


Figure 10: **Convergence speed during fine-tuning.** On tasks where InFOM and baselines perform similarly, our flow occupancy models enable faster policy learning.

variable model over adjacent transitions, without relying on a potentially complicated offline RL procedure (Tarasov et al., 2023b; Park et al., 2024a).

F.3 Flow occupancy models enable faster policy learning

We then investigate whether the proposed method leads to faster policy learning on downstream tasks. We answer this question by an ablation study with a high evaluation frequency, analyzing the performance of various methods throughout the entire fine-tuning phase every 2K gradient steps. We compare InFOM to prior methods on two ExORL tasks (`cheetah run` and `quadruped jump`), including ReBRAC, CRL + IS, DINO + ReBRAC, MBPO + ReBRAC, and FB + IQL (See Appendix D.2 for details of these baselines). We choose these baselines because they perform similarly to our method, helping to prevent counterfactual errors derived from the performance deviation when comparing convergence speed.

We compare different algorithms by plotting the returns at each evaluation step, with the shaded regions indicating one standard deviation. As shown in Fig. 10, InFOM converges faster than prior methods that only pre-train behavioral cloning policies (ReBRAC) or self-supervised state representations (DINO + ReBRAC), demonstrating the effectiveness of extracting temporal information. The observation that methods utilizing a one-step transition model (MBPO + ReBRAC) or a future state classifier (CRL + IS) learn more slowly than our method highlights the importance of predicting long-horizon future events using expressive generative models. Additionally, our flow occupancy models extract rich latent intentions from the unlabeled datasets, resulting in adaptation speed similar to the prior zero-short RL method (FB + IQL).

F.4 Learning with discrete intentions

The choice of the prior over latent variables $p(z)$ is still an open question in the literature. Prior work has used a standard Gaussian distribution (Frans et al., 2024), a uniform von Mises–Fisher distribution Park et al. (2024b); Touati & Ollivier (2021); Zheng et al. (2025), a continuous uniform distribution (Sharma et al., 2019), and a discrete uniform distribution (Eysenbach et al., 2019).

To further investigate the effect of using a discrete set of latent intentions for InFOM, we run additional ablation experiments. We selected a set of discrete latent embeddings $\mathcal{Z} = \{z_1, \cdots, z_K\}$ (a lookup table with $K = 256$), and used a vector quantization (VQ) loss to learn those embeddings together with InFOM as in VQ-VAE (Van Den Oord et al., 2017). Specifically, given a consecutive transition (s, a, z, s', a'), the flow-based intention decoder $q_d(z \mid s, a)$ remains the same, while the intention encoder $p_e(z \mid s', a')$ can now be decomposed into two components: *(1)* the deterministic encoder $p_{\text{enc}} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}^d$ and the quantizer $p_{\text{quant}} : \mathbb{R}^d \to \mathcal{Z}$. The role of the quantizer is to query the closest discrete latent intentions from the encoder outputs using the nearest neighbor,

$$p_{\text{quant}}(p_{\text{enc}}(s', a')) = z_k, \quad \text{where } k = \text{argmin}_{i=1,\cdots,K} \|p_{\text{enc}}(s', a') - z_i\|_2.$$

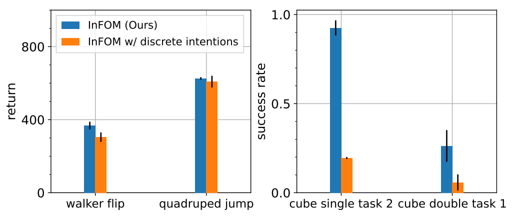


Figure 11: Using discrete intentions slightly decreases InFOM's performance on ExORL tasks (-11%), while drastically decreasing the mean success rate of InFOM on OGBench tasks (-78%).

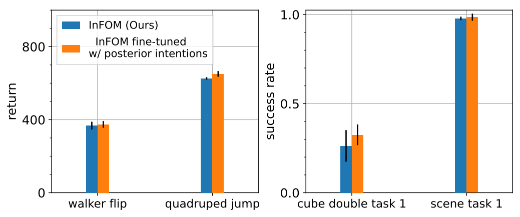


Figure 12: Using the posterior $q_d(z \mid s', a')$ to sample the latents does not significantly change the performance of InFOM ($+7\%$), suggesting that our method is robust against unseen latents. We choose to use the prior $p(z)$ for sampling latents to estimate Q_z throughout our experiments.

Using this quantizer, we replace the surrogate objective in Eq. 4 with the following SARSA flow loss with a vector quantization loss:

$$\mathcal{L}_{\text{SARSA flow}}(p_{\text{enc}}, p_{\text{quant}}, q_d) + \mathcal{L}_{\text{VQ}}(p_{\text{enc}}, p_{\text{quant}}, q_d),$$

$$\begin{aligned}\mathcal{L}_{\text{VQ}}(p_{\text{enc}}, p_{\text{quant}}, q_d) = \mathbb{E}_{p^{\beta}(s',a')}[\| \lfloor p_{\text{enc}}(s', a') \rfloor_{\text{sg}} - z_k(s', a') \|_2^2] \\ + \lambda \mathbb{E}_{p^{\beta}(s',a')}[\| p_{\text{enc}}(s', a') - \lfloor z_k(s', a') \rfloor_{\text{sg}} \|_2^2],\end{aligned}$$

where $\lfloor \cdot \rfloor_{\text{sg}}$ denotes the stop gradient operator, and we use straight-through gradients (Bengio et al., 2013) to optimize the SARSA flow loss. During fine-tuning, we use all the discrete latents $\{z_1, \cdots, z_K\}$ to construct intention-conditioned Q_z estimations (Eq. 6) and distill them into the critic Q as in Eq. 7.

We conducted ablation experiments on two ExORL tasks (`walker flip` and `quadruped jump`) and two OGBench tasks (`cube single task 2` and `cube double task 1`) and report performances aggregated over 8 random seeds. Results in Fig. 11 suggest that using discrete intentions slightly decreases InFOM's performance on ExORL tasks (-11%), while drastically decreasing the mean success rate of InFOM on OGBench tasks (-78%). These results indicate that using a continous latent space generally leads to better performance in our experiments.

F.5 Fine-tuning with posterior intentions

In Sec. 4.4, when estimating the intention-conditioned Q_z for a specific task, we have already sampled the latent z from the prior $p(z)$ instead of the posterior $q_d(z \mid s', a')$. Sampling from the prior, in general, increases the possibility of drawing out-of-distribution latents. We hypothesize that InFOM can generalize over unseen latents on different (s, a) pairs. To quantitatively test this hypothesis, we conduct additional ablation experiments to study the effect of estimating intention-conditioned Q_z using in-distribution latents on the final performance of InFOM. Specifically, we replace the distillation loss in Eq. 7 with a variant that samples z from the posterior $q_d(z \mid s', a')$:

$$\widetilde{\mathcal{L}}(Q) = \mathbb{E}_{(s,a,s',a') \sim p^{\tilde{\beta}}(s,a,s',a'),\, z \sim q_d(z|s',a')} \left[L_2^{\mu} \left(Q_z(s, a) - Q(s, a) \right) \right].$$

We choose to conduct ablation experiments on two ExORL tasks (`walker flip` and `quadruped jump`) and two OGBench tasks (`cube double task 1` and `scene task 1`), aggregating the return and the success rate over 8 random seeds. Results in Fig. 12 indicate that using the posterior to sample the latents for each Q_z does not significantly change the performance of InFOM ($+7\%$). Conversely, these results suggest that InFOM is robust against unseen latents for different (s, a) pairs and using the prior $p(z)$ to sample latents provides sufficient learning signals to drive fine-tuning. We choose to use the prior $p(z)$ for sampling latents to estimate Q_z throughout our experiments.

F.6 Learning with sparse rewards is challenging

We hypothesize that the sparse reward function on `jaco` tasks explains the performance gap between InFOM and baselines. To test this hypothesis, we conduct ablation experiments on `jaco reach top left` and `jaco reach bottom right`, studying whether using *dense* rewards will mitigate the performance gap. Specifically, the dense reward function is defined as $r(s, g) = -\|s - g\|_2$

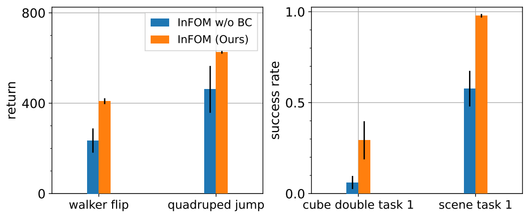


Figure 14: The behavioral cloning regularization in the policy loss is a key component of InFOM.

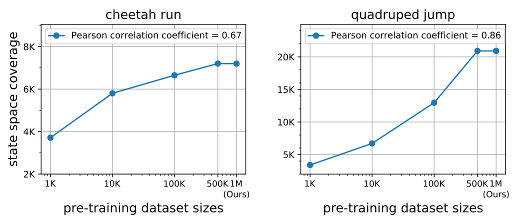


Figure 15: The diversity of the pre-training datasets has a positive correlation with their sizes.

with g as the target position. To make a fair comparison, we fine-tune the ReBRAC baseline on variants of those two `jaco` tasks with dense reward functions, measuring the performance in the original environments. We report returns across 8 random seeds.

Results in Fig. 13 highlight that using a dense reward function results in 3.6$\times$ smaller performance gap, suggesting that the original sparse reward function imposes challenges for learning on `jaco` tasks. We note that Yarats et al. (2022) has also included consistent evidence for this observation, where TD3 + BC (the base algorithm for ReBRAC) performed poorly on the `jaco` domain (Fig. 9 of Yarats et al. (2022)).

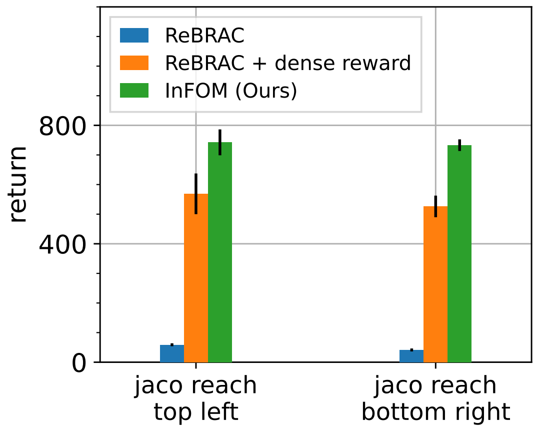


Figure 13: **Reward function structure can impose challenges.** The baseline ReBRAC achieves 3.6$\times$ higher performance on variants of `jaco` tasks with a dense reward function.

F.7 Importance of the behavioral cloning regularization

To study the effect of the BC regularizer (Eq. 8), we conduct experiments comparing a variant of InFOM without the behavioral cloning regularization coefficient ($\alpha = 0$) to our full algorithm with domain-dependent α values (Table 2). We select the same ExORL and OGBench tasks as in Fig. 5 (`walker flip`, `quadruped jump`, `cube double task 1`, and `scene task 1`) and report the means and standard deviations of performance over 8 random seeds after fine-tuning. Results in Fig. 14 suggest that behavioral cloning regularization ($\alpha > 0$) in the policy loss is a key component of our algorithm.

F.8 Diversity of the pre-training datasets

To quantify the diversity of the pre-training dataset, we conduct a statistical analysis on the datasets for two ExORL tasks (`cheetah run` and `quadruped jump`), analyzing the relationship between the size of the dataset and the diversity of the dataset. Following prior work (Park et al., 2023b), we discretize the continuous state space as a high-dimensional grid (up to 10^{-2}) and use the number of unique grid points covered by the dataset to measure the diversity. Results in Fig. 15 show that increasing the dataset size induces a higher diversity in the pre-training datasets, with an average correlation coefficient of 0.76 over those two tasks. Thus, we can study the effect of diverse pre-training datasets on InFOM's performance by varying the pre-training dataset size.

F.9 The effect of dataset sizes

Pre-training dataset size. Since we aim to predict temporally distant future states from heterogeneous data (Sec. 4.1), InFOM implicitly requires a sufficiently diverse dataset for effective pre-training. To study the relationship between the size of pre-training datasets and the performance of our algorithm, we conduct ablation experiments varying the pre-training dataset size in $\{1\text{K}, 10\text{K}, 100\text{K}, 500\text{K}, 1\text{M}\}$. We compare the performances of InFOM on two ExORL tasks

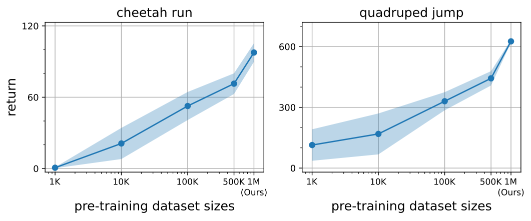


Figure 16: **The effect of pre-training dataset size on InFOM.** Increasing pre-training dataset sizes boosts the final performances of InFOM.

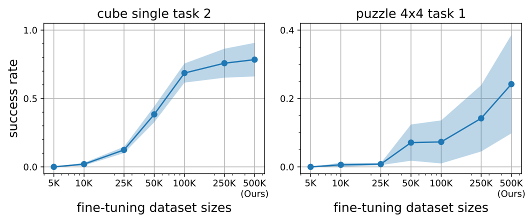


Figure 17: **The effect of fine-tuning dataset size on InFOM.** Increasing the fine-tuning dataset size yields consistent improvements in success rates.

(`cheetah run` and `quadruped jump`) after fine-tuning on the same reward-labeled dataset. We report results across 8 random seeds, following the same evaluation protocol in Appendix D.3.

Results in Fig. 16 indicate that larger pre-training datasets yield higher returns on these tasks. We conjecture that pre-training InFOM on a diverse, reward-free dataset reduces the possibility of sampling out-of-distribution (unseen) intentions, resulting in a higher final performance.

Fine-tuning dataset size. We also conduct ablation experiments studying the effect of fine-tuning dataset sizes. Specifically, we select two OGBench tasks (`cube single task 2` and `puzzle 4x4 task 1`) and vary the size of the fine-tuning datasets in $\{5\text{K}, 10\text{K}, 25\text{K}, 50\text{K}, 100\text{K}, 250\text{K}, 500\text{K}\}$. Again, we aggregate the performance of InFOM over 8 random seeds, following the same evaluation protocol in Appendix D.3.

Results Fig. 17 show that increasing the fine-tuning dataset size (within the chosen range) yields consistent improvements in success rates on the OGBench tasks. Our explanation for these observations is that the size of the fine-tuning dataset affects the accuracy of the reward prediction.

F.10 Fine-tuning on suboptimal datasets

We hypothesize that using highly suboptimal fine-tuning datasets will decrease the downstream performance of InFOM. To study the effect of fine-tuning on suboptimal datasets, we conduct ablation experiments on two ExORL tasks (`cheetah run` and `quadruped jump`) because they have dense reward functions and can still produce diverse rewards. To construct suboptimal datasets, we use the reward quantile to filter each transition in the 10^6 ExORL dataset collected by RND (see Appendix D.1 for details) and then sample 5×10^5 reward-labeled transitions from the remaining transitions. After constructing these suboptimal datasets, we use them to fine-tune InFOM. Results in Fig. 18 indicate that fine-tuning InFOM on highly suboptimal datasets (0.2 reward quantile) achieved only 9% performance of the original InFOM, while using datasets with 0.8 reward quantile can already achieve 85% performance of the original InFOM. These results suggest that using a sufficiently optimal dataset is important for improving the fine-tuning performance.

F.11 The sufficient number of future states in the Q estimation

Since we use MC future states from the InFOM to estimate the intention-conditioned Q_z (Eq. 6), it is possible that the model produces unrealistic future states. Thus, the number of future states N affects the accuracy and variance of the Q value estimation (Eq. 6). To investigate the effect of N, we conduct ablation studies on a total of 8 tasks, with 4 tasks from the ExORL benchmarks (`cheetah walk`, `walker walk`, `walker flip`, and `quadruped jump`) and 4 tasks from the OGBench benchmarks (`cube double task 3`, `puzzle 4x4 task 1`, `cube double task 1`, and `scene task 1`). Below, we report returns and success rates after fine-tuning, aggregating the results over 8 random seeds.

Fig. 19 suggests that, in `cheetah walk` and `puzzle 4x4 task 1`, increasing the number of flow future states yields better performance with consistent variance. In `walker walk` and `cube double task 3`, a larger N does mitigate the high variance in Q_z, at the cost of increasing computation. Taken together, these results indicate that a sufficiently large number of flow future states used in Q_z achieves more accurate estimation of Q values, while reducing the variance. In

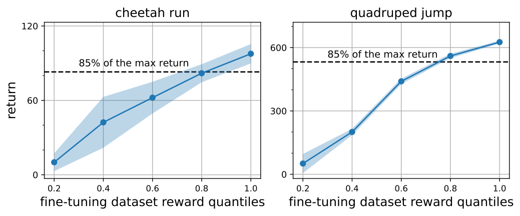


Figure 18: **Fine-tuning on suboptimal datasets.** Fine-tuning on highly suboptimal datasets (0.2 reward quantile) decreased the performance of InFOM, while using a sufficiently optimal (0.8 reward quantile) dataset can already retain the performance.

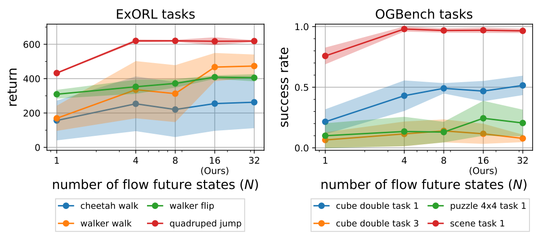


Figure 19: **Using a sufficient number of flow future states is important.** Increasing the number of flow future states (N) in the Q_z estimate boosts the accuracy while reducing variance, resulting in higher final performances of InFOM. We choose $N = 16$ as a balance between the accuracy, variance, and computational constraints in our experiments.

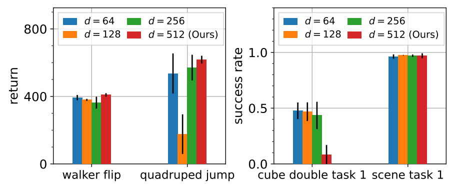
contrast, a smaller number of N potentially yields errors in Q_z from unrealistic future states, resulting in high variance. In practice, our choice of $N = 16$ is a balance between the accuracy, variance, and computational constraints of the estimator.

F.12 Additional hyperparameter ablations

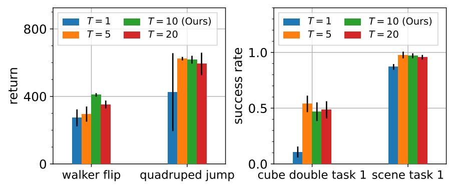
We conduct additional ablation experiments on `walker flip`, `quadruped jump`, `cube double task 1`, and `scene task 1` to study the effect of some key hyperparameters in InFOM (Table 2). Following the same evaluation protocols as in Appendix D.3, we report means and standard deviations across eight random seeds after fine-tuning each variant.

As shown in Fig. 20a, our algorithm is sensitive to the latent intention dimension d. Additionally, the effect of the number of steps for the Euler method T (Fig. 20b) saturates after increasing it to a certain threshold ($T = 10$), suggesting the usage of a common value for all tasks.

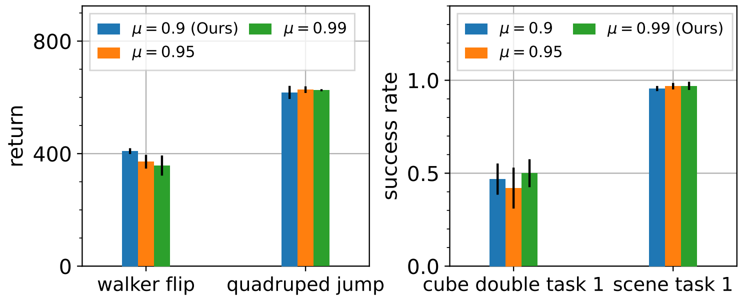
Results in Fig. 20c, Fig. 20d, and Fig. 20e suggest that the expectile μ can affect the performance on ExORL tasks, while having minor effects on OGBench tasks. Importantly, the KL divergence regularization coefficient λ and the behavioral cloning regularization coefficient α are crucial hyperparameters for InFOM, where domain-specific hyperparameter tuning is required. As discussed in Appendix D.4, we generally select one task from each domain to sweep hyperparameters and then use one set of hyperparameters for every task in that domain.



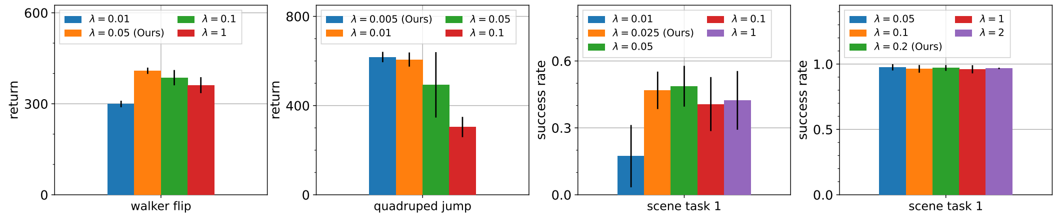
(a) Latent dimension d



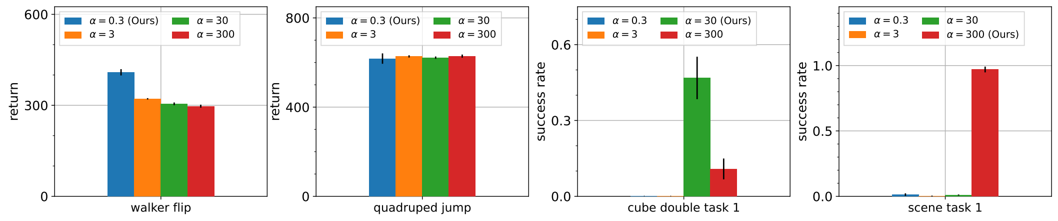
(b) Number of steps for the Euler method T



(c) Expectile μ



(d) KL divergence regularization coefficient λ



(e) Behavioral cloning regularization coefficient α

Figure 20: **Hyperparameter ablations.** We conduct ablations to study the effect of key hyperparameters of InFOM as listed in Table 2 on `walker flip`, `quadruped jump`, `cube double task 1`, and `scene task 1`.