DENSITY MATTERS: NORMALIZED SUCCESSOR MEASURES FOR ZERO-SHOT CONTROL

Anonymous authors
Paper under double-blind review

ABSTRACT

Behavior foundation models (BFMs) promise zero-shot control: pretrain on offline data, then reuse the learned representation for downstream tasks without task-specific training. Successor-measure methods such as Forward–Backward representations (FB) realize this promise for diverse locomotion behaviors, but can struggle when success depends on precise interactions. For example, in pick-and-place tasks, FB policies can reach an object but fail to maintain a grasp or carry it to a specified goal. We attribute these failures to *control bottlenecks*, transitions that are necessary passages to valuable regions where the choice of action substantially affects return. We show that the low-rank squared approximation used by FB can suppress the differences in action values at these states, even at its global optimum, causing the agent to choose incorrect actions. To preserve these differences with limited capacity, we introduce Density-FB, which treats the bilinear output as a score, normalizes the scores into a distribution over future states with a softmax, and learns this distribution with a categorical Bellman objective. Across OGBench, MetaWorld, and ManiSkill, Density-FB improves manipulation success over FB by a median of 64.4%, 30.0%, and 8.5% percentage points, respectively. Additionally, Density-FB preserves correct action rankings with smaller embeddings than vanilla FB.

1 INTRODUCTION

The successor measure (SM) of a policy is its discounted distribution over future states, independent of any reward, so the value of a reward specified only after training is simply its expectation under the SM. Reinforcement Learning (RL) methods such as Forward-Backward (FB) representations use these predictions to learn policy families for zero-shot control (Touati & Ollivier, 2021; Touati et al., 2023; Agarwal et al., 2025). However, in practice even small errors in the approximation can lead these methods to give the wrong ranking to actions, which matters specifically for problems where this ranking is important for downstream control. In manipulation, FB can reach the correct object yet fail to grasp it securely or place it at the goal (Mohan et al., 2026). This raises a natural question: *Why does FB fail to capture such precise behavior?*

We hypothesize that this failure arises because of *control bottlenecks*: transitions that are necessary passages to valuable regions where the choice of action substantially affects return. For example, passing through a doorway determines which side of an environment can subsequently be explored; making the correct contact during an insertion determines whether the object can subsequently be manipulated; and establishing a stable grasp determines whether an object can be carried at all. Although these decisions may occupy only a few steps, their consequences extend far into the future. Methods that learn SM representations must preserve those consequences well enough to distinguish the actions that enable progress.

Carrying a cube toward a goal requires distinguishing actions that maintain a secure grasp from those that let it slip. In Figure 1, FB already holds the cube, yet predicts future cube positions broadly dispersed across the workspace, resembling the dataset marginal, and its rollout eventually drops the cube. With limited capacity, least-squares fitting can capture these broad occupancy patterns while losing the differences between actions needed for successful carrying. If bottleneck decisions are infrequent, such errors can contribute little to the average fitting loss despite preventing the model from identifying the better action.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

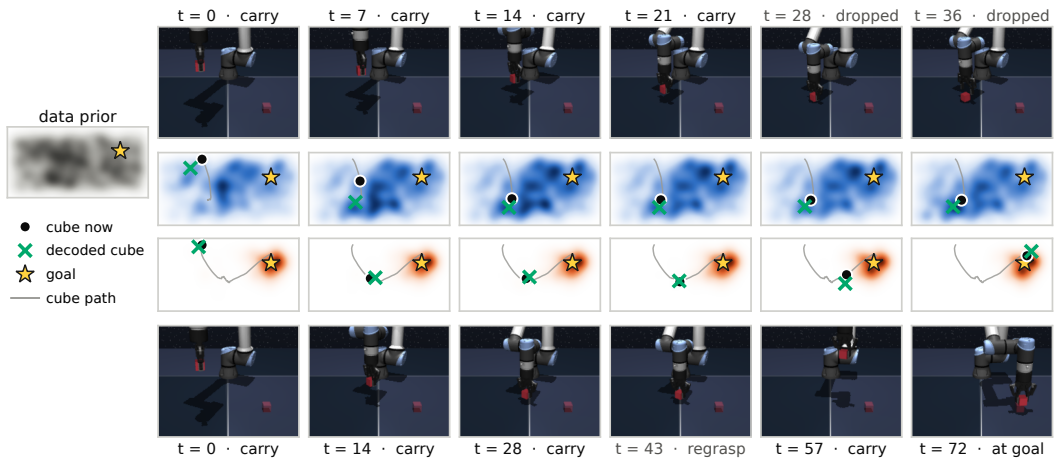


Figure 1: **Predicted future occupancies of the cube position and observed behavior on the workspace.** Both methods start from the same state and goal. FB (blue) predicts future cube occupancy broadly, resembling the dataset distribution (gray), an drops the cube Density-FB (orange) predicts visits concentrated near the goal, and reaches the goal

We introduce Density-FB to better preserve such differences between action-values at control bottlenecks. Our idea is simple: train the low-rank approximation to predict how likely different future states are relative to one another. Instead of using the bilinear output directly as future occupancy, we treat it as a score and pass these scores through a softmax over future states. The resulting occupancy is therefore a nonlinear function of the low-rank features, allowing the same representation size to express action-dependent differences that a direct low-rank fit may lose. We learn these probabilities with a categorical Bellman objective. Because the bilinear output now represents a score rather than an occupancy, FB's usual linear value readout no longer applies; we instead evaluate actions by averaging downstream rewards under their predicted future-state distributions.

We characterize how differences in future occupancy not captured by FB's learned features translate into errors in action values (Section 4). We then outline our modifications to mitigate this in Density-FB (Section 5). Through experiments on didactic MDPs with control bottlenecks, we show that the softmax parameterization preserves action-value differences with fewer features than FB (Section 6.1). We then connect these findings to manipulation through diagnostics of critic predictions during object interactions (Section 4), and demonstrate improved performance across OGBench, MetaWorld, and ManiSkill (Section 6.2).

2 Related Work

Our work connects the learning of reusable future occupancies with the action differences that matter for control. Appendix A provides extended discussion.

Successor representations for zero-shot control. Successor features enable transfer by separating future feature occupancies from reward weights (Dayan, 1993; Barreto et al., 2017). FB and Proto-Successor Measures build on this principle to support rewards specified after pretraining (Touati & Ollivier, 2021; Touati et al., 2023; Agarwal et al., 2025), with subsequent work improving offline learning and task expressivity (Jeen et al., 2024; Cetin et al., 2024). We examine how fitting these reusable representations can erase the action-value differences needed at control bottlenecks.

Contrastive occupancy estimation. Classification offers an alternative way to learn future occupancy, as demonstrated by C-learning and Contrastive RL (Eysenbach et al., 2021; 2022). TD InfoNCE (Zheng et al., 2024) develops this approach through normalized exponentiated scores and a categorical TD objective, which form the basis of our estimator. Density-FB integrates this evaluator into reward-free policy pretraining: predicted occupancies supply reward expectations for policy improvement, while task inference selects a policy for each downstream reward.

Interactions and control bottlenecks. Interaction-based learning highlights decisions that change what an agent can subsequently achieve. HyPE, COInS, and SkiLD use object or factor interactions to discover skills (Chuck et al., 2020; 2024; Wang et al., 2024), while HInt uses them to guide hindsight relabeling (Chuck et al., 2025). Our bottleneck analysis connects this emphasis on consequential interactions to learning SMs by asking whether fitted occupancies preserve the differences in return between actions at these decisions.

3 PRELIMINARIES

Offline reward-free setting. We consider a controlled Markov process $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mu, P)$ and a dataset $\mathcal{D} = \{(s_i, a_i, s_i')\}_{i=1}^N$ collected without downstream reward labels. Here, $\mathcal{S}$ denotes the state space, $\mathcal{A}$ denotes the action space, $\mu \in \Delta(\mathcal{S})$ denotes the initial state distribution, and $P(\cdot \mid s, a) \in \Delta(\mathcal{S})$ denotes the environment transition kernel. Let ρ denote the marginal distribution of next states induced by the offline data-collection process, from which the observed next states $\{s_i'\}_{i=1}^N$ are sampled. Downstream tasks are specified by state rewards $r : \mathcal{S} \to \mathbb{R}$.

Successor measures. For a policy π and discount factor $0 < \gamma < 1$, the normalized SM is the discounted distribution of future states:

$$M^\pi(s, a, ds^+) := (1-\gamma) \sum_{t \geq 0} \gamma^t \mathbb{P}_\pi\big(s_{t+1} \in ds^+ \mid s_0 = s,\, a_0 = a\big)\,, \tag{1}$$

Each term is the distribution of the state $t+1$ steps ahead after taking a and following π, s^+ denotes a generic future state and ds^+ indicates the integration variable in $M^\pi(s, a, ds^+)$ and $\rho(ds^+)$. Since $(1-\gamma)\sum_{t\geq 0}\gamma^t = 1$, $M^\pi(s, a, \cdot)$ is a probability measure. It satisfies the Bellman recursion $M^\pi(s, a, \cdot) = (1-\gamma)P(\cdot \mid s, a) + \gamma \mathbb{E}_{\substack{s' \sim P(\cdot|s,a) \\ a' \sim \pi(\cdot|s')}}[M^\pi(s', a', \cdot)]$, and for any bounded state reward r, integrating the reward under the SM recovers the action value:

$$Q_r^\pi(s, a) = \frac{1}{1-\gamma} \mathbb{E}_{s^+ \sim M^\pi(s,a,\cdot)}[r(s^+)] = \frac{1}{1-\gamma} \int_{\mathcal{S}} r(s^+)\, M^\pi(s, a, ds^+). \tag{2}$$

Forward-Backward representations. FB (Touati & Ollivier, 2021; Touati et al., 2023) learns a family of policies π_z indexed by task vectors $z \in \mathbb{R}^d$, together with a forward representation $F_\theta(s, a, z) \in \mathbb{R}^d$ and a backward representation $B_\theta(s^+) \in \mathbb{R}^d$, which factorize the density of the SM relative to the dataset state marginal ρ as $M_\theta(s, a, z, ds^+) := F_\theta(s, a, z)^\top B_\theta(s^+)\, \rho(ds^+)$. For a dataset transition (s, a, s'), an independently sampled state $s^+ \sim \rho$, and $a' \sim \pi_z(\cdot \mid s')$, FB learns these representations from the Bellman equation through the squared regression objective

$$\mathcal{L}_{\mathrm{FB}} = \mathbb{E}\Big[\big(F_\theta(s, a, z)^\top B_\theta(s^+) - \gamma \bar{F}(s', a', z)^\top \bar{B}(s^+)\big)^2 - 2(1-\gamma)\, F_\theta(s, a, z)^\top B_\theta(s')\Big], \tag{3}$$

where $\bar{F}$ and $\bar{B}$ denote target representations. For a downstream reward r, FB forms $z_r := \mathbb{E}_{s\sim\rho}[r(s)B_\theta(s)]$. Substituting the factorization into Equation (2) gives the linear readout $Q_r^{\pi_z}(s, a) \approx F_\theta(s, a, z)^\top z_r/(1-\gamma)$

TD InfoNCE for policy evaluation. TD InfoNCE (Zheng et al., 2024) estimates a policy's successor distribution using bootstrapped contrastive learning. For a fixed policy π, let $h(s, a, \cdot)$ be a normalized density relative to ρ, and $\bar{h}$ a fixed target density. With exact expectations and normalization, its evaluation objective is

$$\mathcal{L}_{\mathrm{TD}}(h) := -\mathbb{E}\Big[(1-\gamma)\log h(s, a, s') + \gamma \mathbb{E}_{s^+\sim\rho}\big[\bar{h}(s', a', s^+)\log h(s, a, s^+)\big]\Big], \tag{4}$$

where $(s, a, s') \sim \mathcal{D}$ and $a' \sim \pi(\cdot \mid s')$. The first term fits the observed next state; the second fits the target model's predicted continuation. The practical objective approximates normalization and future-state expectations using sampled candidates.

4 A FAILURE MODE AT CONTROL BOTTLENECKS

An SM approximation can capture broad patterns of future occupancy yet fail to distinguish the actions needed for successful control. Prior work has observed that methods for approximating

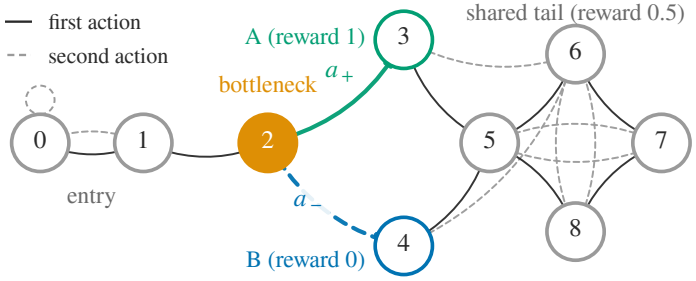


Figure 2: **A control bottleneck.** At state 2, action a_+ enters state A (3), while a_- enters state B (4); both then step into the shared tail (states 5–8). Entering A gives reward 1, B gives 0, and every tail state gives 0.5, so a_+ has a higher return although the two actions' futures differ for a single step.

SMs struggle on robotic manipulation tasks requiring precise interactions (Cetin et al., 2024; Shen et al., 2026; Mohan et al., 2026). As Shen et al. (2026) argue, such interactions can change an object's dynamics and, consequently, its future reachability. An SM approximation must capture these consequences to distinguish actions that enable valuable outcomes. We isolate this requirement through *control bottlenecks*: necessary passages to valuable regions where the choice of action matters for return.

Definition 1 (Control bottleneck)**.** Fix μ, a bounded reward r, a continuation policy π, and thresholds $\Delta > 0$ and v_{high}. Let X be reachable from μ, with $\mu(X) = 0$ and $V_r^\pi(x) \geq v_{\text{high}}$ for every $x \in X$. A measurable set C with $\mu(C) = 0$ is *necessary* for X if, under every policy, trajectories from μ that reach X visit C before or upon first reaching X, almost surely.

For $s \notin C \cup X$, define

$$\mathcal{A}_+(s) := \{a \in \mathcal{A}(s) : P(C \mid s, a) > 0\}, \qquad \mathcal{A}_-(s) := \mathcal{A}(s) \setminus \mathcal{A}_+(s).$$

A necessary set C is a Δ-*control bottleneck* if every entry state s with $\mathcal{A}_+(s) \neq \varnothing$ also satisfies $\mathcal{A}_-(s) \neq \varnothing$ and

$$\sup_{a \in \mathcal{A}_+(s)} Q_r^\pi(s, a) - \sup_{a \in \mathcal{A}_-(s)} Q_r^\pi(s, a) \geq \Delta.$$

At a bottleneck entry state, the critic must distinguish actions with different returns. Yet two actions can have identical expected backward features under their successor distributions despite different values. Lemma 1 shows that, with these features fixed, even optimal least-squares fitting then assigns the actions equal values, hiding the advantage of the better action.

Lemma 1 (Loss of a bottleneck action distinction)**.** *Fix $\pi = \pi_z$ and $B \in L^2(\rho; \mathbb{R}^d)$. At a bottleneck entry state s, choose $a_+ \in \mathcal{A}_+(s)$ and $a_- \in \mathcal{A}_-(s)$ with $Q_r^\pi(s, a_+) > Q_r^\pi(s, a_-)$.*

For each $a \in \{a_+, a_-\}$, let $M_a := M^\pi(s, a, \cdot)$ have density $m_a \in L^2(\rho)$ relative to ρ. Let $\widehat{m}_a$ be its best $L^2(\rho)$ approximatio of the form $f^\top B$, with $f \in \mathbb{R}^d$.

The fitted densities coincide ρ-almost everywhere iff

$$\mathbb{E}_{M_{a_+}}[B] = \mathbb{E}_{M_{a_-}}[B].$$

Under this condition, the fitted values $\widehat{Q}_r^\pi(s, a) := (1 - \gamma)^{-1} \int_{\mathcal{S}} r(s^+) \widehat{m}_a(s^+) \rho(ds^+)$ satisfy

$$\widehat{Q}_r^\pi(s, a_+) = \widehat{Q}_r^\pi(s, a_-),$$

despite the strictly positive true action-value gap.

Proof sketch. Optimal linear fitting projects each successor density onto the span of the backward features. The projections coincide exactly when the density difference is orthogonal to every feature, equivalently when the feature expectations agree; integrating the fitted densities against r then gives identical values (Appendix B).

Lemma 1 shows that, with backward features fixed, identical feature expectations can make actions with different returns indistinguishable to the fitted critic. The didactic MDP in Figure 2 makes this concrete: at state 2, a_+ enters rewarded state A, whereas a_- enters unrewarded state B, before both

branches reach a shared tail. Although most of their future occupancy is shared, visiting A makes a_+ preferable. In Section 6.1, we test whether FB retains enough information to rank these actions correctly. To mitigate this failure, Density-FB exponentiates and normalizes bilinear scores into distributions, fits them to Bellman targets, and evaluates actions by averaging rewards under them.

5 Density-FB: Learning Successor Measures as Distributions

To distinguish actions with different returns, Density-FB changes how FB's compact features represent successor distributions. We use the bilinear output to score future states, then exponentiate and normalize these scores so that the predicted densities can extend beyond the linear combinations of backward features used by FB. Pretraining then alternates fitting these distributions to Bellman targets by cross-entropy (Section 5.1) with improving policies through reward expectations (Section 5.2). After pretraining, a downstream reward determines the task index used to select and execute a frozen policy.

5.1 Learning a normalized successor distribution

For temperature $\tau > 0$, define the score $f_\theta(s, a, z, s^+) := F_\theta(s, a, z)^\top B_\theta(s^+)/\tau$ and normalize its exponential relative to ρ, where $\tau > 0$ is a temperature and $\tilde{s}^+$ is a reference state drawn from ρ:

$$h_\theta(s, a, z, s^+) := \frac{\exp f_\theta(s, a, z, s^+)}{\mathbb{E}_{\tilde{s}^+ \sim \rho}[\exp f_\theta(s, a, z, \tilde{s}^+)]}. \tag{5}$$

Assuming a finite denominator, the predicted SM $\widehat{M}_\theta(s, a, z, ds^+) := h_\theta(s, a, z, s^+)\rho(ds^+)$ is a probability measure. The following proposition relates its approximation accuracy to preservation of a bottleneck distinctions between action-values in the bottleneck.

Proposition 1 (Preservation of a bottleneck distinction)**.** *Fix $\pi = \pi_z$ and a bounded measurable reward r. At a bottleneck entry state s, choose $a_+ \in \mathcal{A}_+(s)$ and $a_- \in \mathcal{A}_-(s)$ with $\delta := Q_r^\pi(s, a_+) - Q_r^\pi(s, a_-) > 0$, and $R := \sup_{\mathcal{S}} r - \inf_{\mathcal{S}} r$. Any fit satisfying*

$$D_{\mathrm{KL}}\Big(M^\pi(s, a, \cdot) \,\Big\|\, \widehat{M}_\theta(s, a, z, \cdot)\Big) < \frac{(1-\gamma)^2\delta^2}{2R^2}, \qquad a \in \{a_+, a_-\},$$

preserves their action ordering $\widehat{Q}_r^\pi(s, a_+) > \widehat{Q}_r^\pi(s, a_-)$.

Proof sketch. The total variation distance by the square root of half the KL divergence, thereby controlling differences in expected bounded rewards. Under the stated KL condition, each action-value error is smaller than $\delta/2$, so their combined error cannot reverse the true gap δ (Appendix B).

For a fixed policy in a finite MDP, if each update matches its Bellman target exactly, the predicted distributions eventually become accurate enough to preserve the action ordering (Appendix B.3). In practice, we approximate these updates using minibatches of offline transitions and sampled future states.

Evaluating task-conditioned policies. We apply TD InfoNCE (Equation (4)) to the policy family $\pi_\omega(\cdot \mid s, z)$. Tasks are specified by latent vectors $z \sim p_Z$, sampled randomly or inferred from goal-reaching rewards, or directly by goal states, with benchmark-specific choices detailed in Appendix C.7. For a minibatch $\{(s_i, a_i, s_i', \mathrm{terminal}_i)\}_{i=1}^n$, the next states serve as candidates $s_j^+ := s_j'$, giving probabilities

$$p_j(s, a, z) := \frac{\exp f_\theta(s, a, z, s_j^+)}{\sum_{\ell=1}^n \exp f_\theta(s, a, z, s_\ell^+)}. \tag{6}$$

Here i indexes transitions and j indexes candidates, so candidate i is the observed next state of transition i. Let $\bar{p}_j$ denote target-network probabilities, sample $a_i' \sim \pi_\omega(\cdot \mid s_i', z_i)$, and set $d_i := \gamma(1 - \mathrm{terminal}_i)$. We fit the Bellman targets by cross-entropy:

$$t_{ij} := \mathrm{sg}[(1 - d_i)\mathbf{1}[j = i] + d_i \bar{p}_j(s_i', a_i', z_i)], \quad a_i' \sim \pi_\omega(\cdot \mid s_i', z_i),\ d_i := \gamma(1 - \mathrm{terminal}_i),$$
$$\mathcal{L}_{\mathrm{DFB}} := -\frac{1}{n}\sum_{i,j} t_{ij} \log p_j(s_i, a_i, z_i), \tag{7}$$

Algorithm 1 Density-FB training and zero-shot control

Require: Dataset $\mathcal{D}$, successor networks (F_θ, B_θ), target networks $(\bar{F}, \bar{B})$, actor π_ω
1: **for** each training step **do**
2: Sample transitions and task vectors and use next states as future-state candidates
3: Compute online and target probabilities (Equation (6))
4: Fit Bellman targets by cross-entropy (Equation (7))
5: Improve the actor using reward-averaged values and behavior regularization (Equation (10))
6: Update target networks, reward encoder B_ξ, and feature mean $\hat{\mu}$
7: **end for**
Zero-shot control for reward r
8: Infer z_r from reward-labeled offline states
9: Execute the frozen policy $\pi_\omega(\cdot \mid s, z_r)$

where sg stands for stop-gradient through the targets, which combine the observed next state with the predicted continuation. For terminal transitions, all mass remains on the observed next state, following the absorbing-state convention.

5.2 Policy improvement during pretraining

To improve the policies being evaluated, we compute action values by averaging rewards under their predicted distributions:

$$\widehat{Q}_{\theta,r}(s,a,z) := \frac{1}{1-\gamma} \sum_{j=1}^{n} p_j(s,a,z)\, r(s_j^+). \tag{8}$$

Because F_θ parameterizes scores, the FB readout $F_\theta(s,a,z)^\top z$ does not generally recover this reward expectation: the cross-entropy objective fixes the scores only up to a constant per state-action pair, and our preliminary ablations demonstrated that training the actor on $F_\theta(s,a,z)^\top z$ generally gave much lower success (Appendix C.4). During reward-free pretraining, each task vector z defines a reward by weighting the learned state features:

$$r_z(s^+) := \frac{\big(B_\xi(s^+) - \hat{\mu}\big)^\top z}{\sqrt{d}}, \tag{9}$$

where B_ξ is a slowly updated backward encoder, $\hat{\mu}$ its running feature mean, and $1/\sqrt{d}$ sets the reward scale. Each policy $\pi_\omega(\cdot \mid s, z)$ is therefore trained to maximize the expected discounted return under its corresponding reward r_z.

$$\mathcal{J}_{\text{value}}(\omega) := \mathbb{E}_{\substack{s\sim\mathcal{D},\ z\sim p_Z \\ a\sim\pi_\omega(\cdot|s,z)}} \left[\widehat{Q}_{\theta,r_z}(s,a,z)\right]. \tag{10}$$

We combine this objective with behavior regularization for offline training; Appendix C.3 gives the full objectives and value scaling.in Appendix C.3. We outline the connection between our objective and successor features in Appendix B.4.

Zero-shot control. For a downstream reward r, we infer z_r from reward-labeled offline states using the encoder in Appendix C.7, then execute the frozen policy $\pi_\omega(\cdot \mid s, z_r)$ without further training.

6 Experiments

Our experiments test whether FB's least squares approximation can discard rare but control-critical action distinctions, and whether the Density-FB preserves those distinctions through normalization and the subsequent policy extraction process introduced in Section 5. We first isolate approximation errors in a didactic MDP with exact SMs, varying feature dimension, examining FB's learned feature span, and ablating Density-FB's components (Section 6.1). To connect these findings to manipulation, we examine policy failures alongside critic predictions of how actions change future cube occupancies (Figure 5). We then evaluate downstream performance on OGBench (Park et al.,

2025a), MetaWorld (McLean et al., 2025), and ManiSkill (Tao et al., 2025), and assess performance beyond manipulation using ExoRL datasets (Yarats et al., 2022) on Deepmind Control Suite (DMC) (Section 6.2).

Experimental protocol. All methods operate on proprioceptive states and are trained for 10^6 gradient steps on the same fixed offline dataset. We report results over 5 independent training seeds. For offline policy learning, we use the actor from Bagatella et al. (2026a), based on Flow Q-Learning (Park et al., 2025b). Success rates are averaged over the final three evaluations. Full implementation details are given in Appendix C.

Baselines. We compare against two groups of methods. (i) We consider methods that learn reusable representations from the offline dataset: **FB** (Touati & Ollivier, 2021; Touati et al., 2023), our primary baseline, together with **HILP** (Park et al., 2024) and **RLDP** (Jajoo et al., 2026) to test whether the manipulation failure is shared by alternative geometric and predictive representations. (ii) We compare against goal-conditioned offline RL methods: **GCIQL** (Kostrikov et al., 2022; Park et al., 2025a), **CRL** (Eysenbach et al., 2022), and **QRL** (Wang et al., 2023a). These methods train directly for goal reaching using hindsight goals sampled from the offline data, and help us examine how well the zero-shot policies compare with methods trained explicitly for goal-reaching control.

6.1 Distinguishing actions at a control bottleneck

We use the nine-state MDP in Figure 2, with a fixed continuation policy and exact successor measures. At state 2, the actions enter differently rewarded branches before reaching a shared tail, giving a true action-value gap of 1. We vary the feature dimension and report ten runs per setting; Appendix E gives the setup and additional policy settings.

Do the critics distinguish the two actions? FB assigns the bottleneck actions nearly identical values, whereas Density-FB preserves their ordering (Figure 3a). To examine whether better forward fitting could recover the missing gap, we freeze FB's learned backward features and refit the forward vectors to the exact successor densities by ρ-weighted least squares. The fitted occupancies remain indistinguishable (Figure 3b), illustrating Lemma 1: the learned feature span cannot recover the difference between the two branches. Figure 3c) relates Density-FB's KL error to the sufficient preservation threshold in Proposition 1.

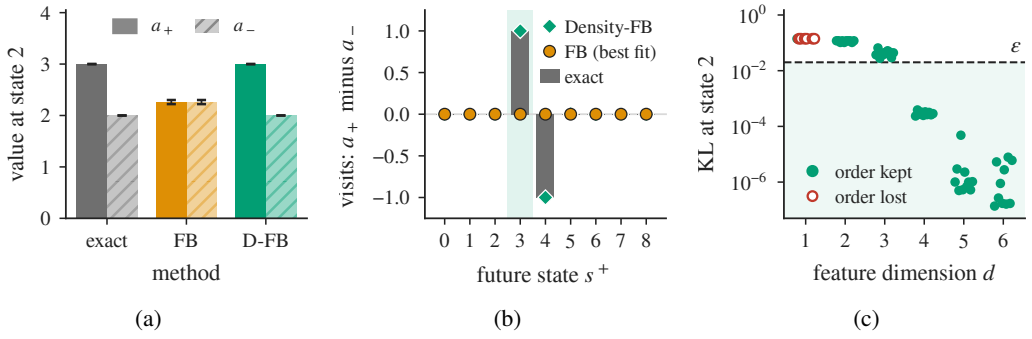


Figure 3: **At the bottleneck of Figure 2, FB's features erase the action-value difference; Density-FB keeps it.** *(a)* Predicted and exact action values at the bottleneck, 95% CI over ten runs. *(b)* Differences ($a_+ - a_-$) in discounted future visits. Black bars are the ground truth, Orange circles: the best LSIF with FB. Green circles: Density-FB at the same rank. *(c)* KL divergence from the exact model to Density-FB's predicted successor distribution at state 2 (hollow red: a_- ranked above a_+). Below the dashed threshold ε of Proposition 1 the ordering is guaranteed.

Preserving action-value gaps with fewer features. Normalizing FB's target leaves recovery near zero through six features, whereas applying a softmax recovers the full gap with four features under either squared error or cross-entropy (Figure 4). The softmax parameterization therefore accounts for the capacity advantage in this MDP. Smoothing the immediate target instead limits recovery to approximately 71%; Appendix E.1 outlines the variants.

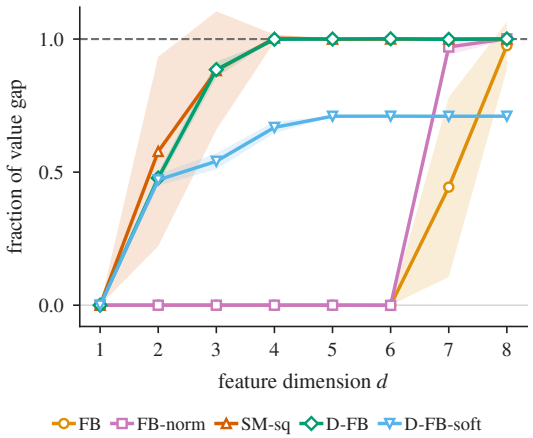


Figure 4: **Density-FB keeps the action-value difference with fewer features.** Predicted value difference at state 2 divided by the exact one, against the feature dimension; mean and 95% interval over ten TD runs. The dashed line is exact recovery. The variants are defined in Appendix E.1.

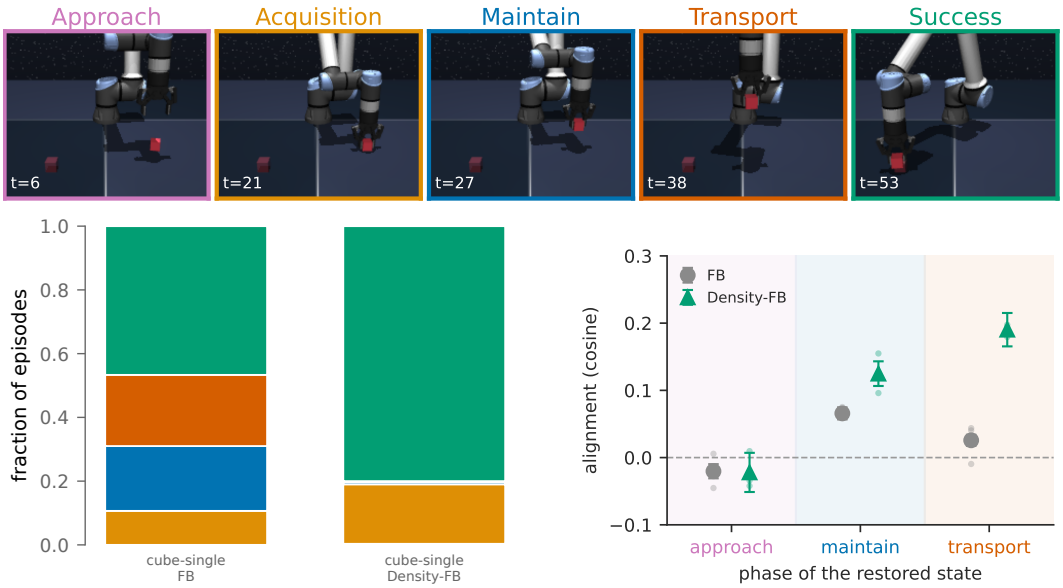


Figure 5: **Cube manipulation failures and critic alignment.** Top: the stages in one successful Density-FB episode on `cube-single` (translucent cube: goal). (a) First failed stage: approach (reach), acquisition (grasp), maintain (hold), or transport (carry to goal); success denotes completion. (b) Cosine similarity between predicted and rollout-estimated action differences in discounted future cube visits; zero indicates no directional alignment. Large markers and error bars show mean ± SE over evaluated states; small markers show the three seed means.

From tabular bottlenecks to manipulation tasks. We next examine where FB fails in manipulation tasks and whether the critic captures how changing an action changes the cube's future in the `cube-single` from OGBench. Following Mohan et al. (2026), we classify each evaluation episode as successful or by its first failed stage (Figure 5, Top). Density-FB fails less often at both stages (Fig. 5a). Most FB failures occur after grasping, while holding or carrying the cube. Density-FB has fewer failures at both stages (Figure 5a). We next examine the corresponding critic predictions by restoring states from each phase and comparing predicted differences between two actions' discounted future cube occupancies with rollout estimates under the critic's continuation policy. Both methods have near-zero cosine alignment during approach, but Density-FB aligns more closely during holding and carrying, with the largest separation during transport (Figure 5b).

Table 1: **Performance across benchmarks.** Results are mean ± SE over five seeds. Benchmark rows average tasks, with propagated SEs. **Bold**: best mean and others within its SE. Δ: Density-FB's relative improvement over the best baseline (%), colored from red ($\leq -25\%$) through gray to green ($\geq +25\%$). Benchmark-specific configurations, task conditioning, and sampling are detailed in Appendix C.7.

Task/Domain	FB	HILP	RLDP	GCIQL	CRL	QRL	Density-FB	Δ next best
OGBench (success rate, %)	18.7 ±1.3	38.3 ±2.0	14.0 ±0.7	53.0 ±1.0	16.0 ±0.4	3.7 ±0.2	**78.8** ±2.0	(+48.7%)
`cube-single`	45.1 ±2.9	64.4 ±4.8	27.5 ±1.2	68.0 ±2.1	19.0 ±0.7	5.0 ±0.4	**93.1** ±0.6	(+36.9%)
`cube-double`	1.7 ±1.1	9.2 ±1.7	2.8 ±0.4	40.0 ±1.8	10.0 ±0.7	1.0 ±0.0	**69.5** ±1.2	(+73.8%)
`scene`	9.3 ±2.2	41.3 ±3.1	11.6 ±1.7	51.0 ±1.4	19.0 ±0.7	5.0 ±0.4	**73.7** ±6.0	(+44.5%)
MetaWorld (success rate, %)	52.5 ±5.9	33.1 ±3.5	39.7 ±5.3	10.0 ±3.3	16.9 ±2.7	**87.7** ±0.8	85.2 ±2.8	(−2.9%)
`push`	29.6 ±3.4	17.2 ±3.8	57.6 ±1.7	4.0 ±1.5	18.4 ±2.6	80.8 ±1.1	**91.2** ±1.9	(+12.9%)
`bin-picking`	62.4 ±17.4	49.2 ±9.1	19.6 ±15.4	0.0 ±0.0	0.0 ±0.0	**98.0** ±1.0	92.4 ±2.4	(−5.7%)
`box-close`	65.6 ±1.8	32.8 ±3.4	42.0 ±3.8	26.0 ±9.7	32.4 ±7.8	**84.4** ±1.8	72.0 ±7.7	(−14.7%)
ManiSkill (success rate, %)	63.3 ±1.7	44.4 ±2.4	56.0 ±1.6	36.5 ±1.0	13.4 ±1.8	65.8 ±0.9	**77.3** ±1.0	(+17.5%)
`push-cube`	88.7 ±3.4	49.2 ±6.3	40.8 ±3.5	68.4 ±1.9	29.6 ±6.9	**96.4** ±0.4	**97.2** ±1.2	(+0.8%)
`pick-cube`	4.4 ±0.4	6.4 ±1.3	**39.6** ±2.9	2.8 ±0.4	4.4 ±0.7	7.2 ±1.3	**40.8** ±3.2	(+3.0%)
`stack-cube`	**70.4** ±3.0	56.8 ±5.7	57.2 ±3.7	28.0 ±3.4	2.4 ±1.0	**67.6** ±1.5	**70.8** ±3.3	(+0.6%)
`poke-cube`	90.0 ±1.9	51.2 ±5.9	58.4 ±5.3	73.6 ±1.7	10.8 ±1.2	**92.0** ±2.3	**92.8** ±1.6	(+0.9%)
`lift-peg`	63.2 ±6.8	58.4 ±6.0	**84.0** ±1.7	9.6 ±2.6	20.0 ±5.3	66.0 ±3.4	**84.7** ±1.1	(+0.8%)

Table 2: **ExoRL (DMC).** Results are mean ± SE. **Bold**: best mean and others within its SE. Δ: Density-FB's relative improvement over the next-best baseline (%).

Task/Domain	FB	HILP	RLDP	Density-FB	Δ next best
ExoRL (DMC returns)	648.2 ±2.6	620.1 ±4.5	610.2 ±7.2	**700.5** ±10.3	(+8.1%)
`walker`	**811.5** ±2.6	796.4 ±3.4	723.9 ±8.2	750.2 ±13.3	(−7.6%)
`quadruped`	595.6 ±4.1	694.8 ±4.9	665.0 ±6.2	**760.5** ±9.1	(+9.5%)
`point-mass`	513.0 ±8.9	371.0 ±16.6	476.3 ±17.6	**751.5** ±33.1	(+46.5%)
`cheetah`	**672.7** ±2.2	618.3 ±2.6	575.6 ±20.1	539.9 ±18.8	(−19.7%)

6.2 Performance across benchmarks

We next evaluate Density-FB on 11 manipulation tasks from OGBench, MetaWorld, and ManiSkill, and locomotion tasks in DMC (Table 1). Dataset details are provided in Appendix D. Density-FB achieves higher mean success than FB on every manipulation task, with average gains of 60.1, 32.7, and 14.0 percentage points on OGBench, MetaWorld, and ManiSkill, respectively. It also exceeds HILP and RLDP on all 11 tasks. Against the goal-conditioned baselines, Density-FB achieves the highest mean success on nine tasks, including all three OGBench and all five ManiSkill tasks. QRL has higher mean success on `bin-picking` and `box-close`, and on the MetaWorld average. On OGBench, Density-FB is conditioned on the task's goal state, as are the goal-conditioned baselines.

Beyond manipulation. We evaluate Density-FB on DMC using the ExoRL RND datasets (Table 2) to test zero-shot reward inference, using reward samples to determine the task vectors. Density-FB achieves the highest mean DMC return under zero-shot reward inference (700.5 vs. 648.2 for FB), leading on quadruped and point-mass but trailing FB on walker and cheetah. It therefore remains competitive beyond manipulation, with its most consistent gains in the manipulation settings motivating our bottleneck analysis.´

7 Conclusion

We studied how SM approximations can erase action-value differences at control bottlenecks. Our analysis characterizes this failure under least-squares fitting and establishes sufficient accuracy conditions for preserving action ordering. Motivated by these findings, Density-FB learns normalized future-state distributions from bilinear scores using a categorical Bellman objective, then evaluates actions by averaging rewards under those distributions. Across eleven manipulation tasks Density-FB outperforms FB on every task. Together, these findings emphasize that reusable occupancy predictions must retain the consequences of action choices to support precise control.

AI USAGE STATEMENT

In this work, we used generative AI tools to refine mathematical claims, experimental design and interpretation of results. We have not used generative AI tools for generating synthetic datasets, implementing methods, translating content, or cleaning and reformatting datasets, and qualitative and thematic data analysis are not applicable to this work. Additionally, we used generative AI tools to edit the manuscript for readability. We have reviewed all AI-assisted work. Specifically, we independently checked mathematical claims and proofs, verified literature references against their original sources, and checked interpretations against experimental results. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REFERENCES

S. Agarwal, H. Sikchi, P. Stone, and A. Zhang. Proto successor measure: Representing the behavior space of an rl agent. In *Proceedings of the 42nd International Conference on Machine Learning (ICML'25)*, 2025.

S. Azad, J. Hoffmann, I. Nematollahi, H. Zhu, A. Valada, and J. Boedecker. Spectral alignment in forward-backward representations via temporal abstraction. *arXiv preprint arXiv:2603.20103*, 2026.

M. Bagatella, M. Pirotta, A. Touati, A. Lazaric, and A. Tirinzoni. TD-JEPA: Latent-predictive representations for zero-shot reinforcement learning. In *The Fourteenth International Conference on Learning Representations*, 2026a.

M. Bagatella, T. Rupf, G. Martius, and A. Krause. Soft forward-backward representations for zero-shot reinforcement learning with general utilities. *arXiv preprint arXiv:2602.06769*, 2026b.

A. Barreto, W. Dabney, R. Munos, J. Hunt, T. Schaul, H. Van Hasselt, and D. Silver. Successor features for transfer in reinforcement learning. In I. Guyon, U. von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (eds.), *Proceedings of the 31st International Conference on Advances in Neural Information Processing Systems (NeurIPS'17)*. Curran Associates, 2017.

M. G. Bellemare, W. Dabney, and Rémi R. Munos. A distributional perspective on reinforcement learning. In D. Precup and Y. Teh (eds.), *Proceedings of the 34th International Conference on Machine Learning (ICML'17)*, volume 70, pp. 449–458. Proceedings of Machine Learning Research, 2017.

E. Cetin, A. Touati, and Y. Ollivier. Finer behavioral foundation models via auto-regressive features and advantage weighting. *Reinforcement Learning Journal*, 2024.

C. Chuck, S. Chockchowwat, and S. Niekum. Hypothesis-driven skill discovery for hierarchical deep reinforcement learning. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 5572–5579. IEEE, 2020.

C. Chuck, K. Black, A. Arjun, Y. Zhu, and S. Niekum. Granger causal interaction skill chains. *Transactions on Machine Learning Research*, 2024. ISSN 2835-8856. URL https://openreview.net/forum?id=iA2KQyoun1.

C. Chuck, F. Feng, C. Qi, C. Shi, S. Agarwal, A. Zhang, and S. Niekum. Null counterfactual factor interactions for goal-conditioned reinforcement learning. In *International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=2uPZ4aX1VV.

P. Dayan. Improving generalization for temporal difference learning: The successor representation. *Neural Computation*, 1993.

B. Dubail, S. Stojanovic, and A. Proutiere. Shift before you learn: Enabling low-rank representations in reinforcement learning. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen (eds.), *Proceedings of the 39th International Conference on Advances in Neural Information Processing Systems (NeurIPS'25)*. Curran Associates, 2025.

B. Eysenbach, R. Salakhutdinov, and S. Levine. C-learning: Learning to achieve goals via recursive classification. In *Proceedings of the Ninth International Conference on Learning Representations*, 2021.

B. Eysenbach, T. Zhang, S. Levine, and R. Salakhutdinov. Contrastive learning as goal-conditioned reinforcement learning. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh (eds.), *Proceedings of the 36th International Conference on Advances in Neural Information Processing Systems (NeurIPS'22)*. Curran Associates, 2022.

J. Farebrother, J. Orbay, Q. Vuong, A. Taiga, Y. Chebotar, T. Xiao, A. Irpan, S. Levine, P. Castro, A. Faust, A. Kumar, and R. Agarwal. Stop regressing: Training value functions via classification for scalable deep RL. In R. Salakhutdinov, Z. Kolter, K. Heller, A. Weller, N. Oliver, J. Scarlett, and F. Berkenkamp (eds.), *Proceedings of the 41st International Conference on Machine Learning (ICML'24)*, volume 251 of *Proceedings of Machine Learning Research*. PMLR, 2024.

J. Farebrother, M. Pirotta, A. Tirinzoni, R. Munos, A. Lazaric, and A. Touati. Temporal difference flows. In *Forty-second International Conference on Machine Learning*, 2025.

N. Hansen, H. Su, and X. Wang. Learning massively multitask world models for continuous control. In *Proceedings of the Fourteenth International Conference on Learning Representations*, 2026.

P. Jajoo, H. Sikchi, S. Agarwal, A. Zhang, S. Niekum, and M. White. Regularized latent dynamics prediction is a strong baseline for behavioral foundation models. In *The Fourteenth International Conference on Learning Representations*, 2026.

M. Janner, I. Mordatch, and S. Levine. Gamma-models: Generative temporal difference learning for infinite-horizon prediction. In H. Larochelle, M. Ranzato, R. Hadsell, M.-F. Balcan, and H. Lin (eds.), *Proceedings of the 34th International Conference on Advances in Neural Information Processing Systems (NeurIPS'20)*. Curran Associates, 2020.

S. Jeen, T. Bewley, and J. Cullen. Zero-shot reinforcement learning from low quality data. In *Proceedings of the 38th International Conference on Advances in Neural Information Processing Systems (NeurIPS'24)*. Curran Associates, 2024.

I. Kostrikov, A. Nair, and S. Levine. Offline reinforcement learning with implicit q-learning. In *Proceedings of the Tenth International Conference on Learning Representations*, 2022.

R. McLean, E. Chatzaroulas, L. McCutcheon, F. Röder, T. Yu, Z. He, K. Zentner, R. Julian, J. Terry, I. Woungang, N. Farsad, and P. Castro. Meta-world+: An improved, standardized, RL benchmark. In *NeurIPS Datasets and Benchmarks Track*, 2025.

A. Mohan, A. Zhang, and M. Lindauer. Structure in deep reinforcement learning: A survey and open problems. *Journal of Artificial Intelligence Research*, 79, 2024.

A. Mohan, C. Chuck, S. Agarwal, and A. Zhang. Is manipulation hard for successor measures? In *Third Reinforcement Learning Beyond Rewards Workshop at the Reinforcement Learning Conference*, 2026.

S. Park, T. Kreiman, and S. Levine. Foundation policies with hilbert representations. In R. Salakhutdinov, Z. Kolter, K. Heller, A. Weller, N. Oliver, J. Scarlett, and F. Berkenkamp (eds.), *Proceedings of the 41st International Conference on Machine Learning (ICML'24)*, volume 251 of *Proceedings of Machine Learning Research*. PMLR, 2024.

S. Park, K. Frans, B. Eysenbach, and S. Levine. Ogbench: Benchmarking offline goal-conditioned rl. In *Proceedings of the Thirteenth International Conference on Learning Representations (ICLR'25).*, 2025a.

S. Park, Q. Li, and S. Levine. Flow q-learning. In *Forty-second International Conference on Machine Learning*, 2025b.

T. Rupf, M. Bagatella, M. Vlastelica, and A. Krause. Optimistic task inference for behavior foundation models. In *The Fourteenth International Conference on Learning Representations*, 2026.

M. Seitzer, B. Schölkopf, and G. Martius. Causal influence detection for improving efficiency in reinforcement learning. *Advances in neural information processing systems*, 34:22905–22918, 2021.

T. Shen, C. Chuck, F. Feng, and B. Huang. Learning object manipulation from scratch via contrastive interaction. *arXiv preprint arXiv:2606.11525*, 2026.

S. Tao, F. Xiang, A. Shukla, Y. Qin, X. Hinrichsen, X. Yuan, C. Bao, X. Lin, Y. Liu, T. Chan, Y. Gao, X. Li, T. Mu, N. Xiao, A. Gurha, V. Rajesh, Y. Choi, Y. Chen, Z. Huang, R. Calandra, R. Chen, S. Luo, and H. Su. Maniskill3: Gpu parallelized robotics simulation and rendering for generalizable embodied ai. In *Robotics: Science and Systems*, 2025.

A. Touati and Y. Ollivier. Learning one representation to optimize all rewards. In M. Ranzato, A. Beygelzimer, K. Nguyen, P. Liang, J. Vaughan, and Y. Dauphin (eds.), *Proceedings of the 35th International Conference on Advances in Neural Information Processing Systems (NeurIPS'21)*. Curran Associates, 2021.

A. Touati, J. Rapin, and Y. Ollivier. Does zero-shot reinforcement learning exist? In *Proceedings of the Eleventh International Conference on Learning Representations (ICLR'23).*, 2023.

T. Wang, A. Torralba, P. Isola, and A. Zhang. Optimal goal-reaching reinforcement learning via quasimetric learning. In A. Krause, E. Brunskill, K. Cho, B. Engelhardt, S. Sabato, and J. Scarlett (eds.), *Proceedings of the 40th International Conference on Machine Learning (ICML'23)*, volume 202 of *Proceedings of Machine Learning Research*. PMLR, 2023a.

Z. Wang, J. Hu, P. Stone, and R. Martín-Martín. ELDEN: Exploration via local dependencies. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Proceedings of the 37th International Conference on Advances in Neural Information Processing Systems (NeurIPS'23)*. Curran Associates, 2023b. URL https://openreview.net/forum?id=sL4pJBXkxu.

Z. Wang, J. Hu, C. Chuck, S. Chen, R. Martín-Martín, A. Zhang, S. Niekum, and P. Stone. Skild: Unsupervised skill discovery guided by factor interactions. *Advances in Neural Information Processing Systems*, 37:77748–77776, 2024.

D. Yarats, D. Brandfonbrener, H. Liu, M. Laskin, P. Abbeel, A. Lazaric, and L. Pinto. Don't change the algorithm, change the data: Exploratory data for offline reinforcement learning. *arXiv preprint arXiv:2201.13425*, 2022.

C. Zheng, R. Salakhutdinov, and B. Eysenbach. Contrastive difference predictive coding. In *Proceedings of the Twelfth International Conference on Learning Representations*, 2024.

C. Zheng, S. Park, S. Levine, and B. Eysenbach. Intention-conditioned flow occupancy models. In *Proceedings of the Fourteenth International Conference on Learning Representations*, 2026.

A Extended related Work

Successor representations for zero-shot control. Successor features separate discounted feature occupancies from reward weights, enabling transfer across tasks with shared dynamics (Dayan, 1993; Barreto et al., 2017). Forward-Backward representations (FB) learn a low-rank factorization of SMs and a family of task-conditioned policies (Touati & Ollivier, 2021; Touati et al., 2023), while Proto-Successor Measures learn a collection of SMs spanning different behaviors (Agarwal et al., 2025). Subsequent work improves offline learning (Jeen et al., 2024), representation and policy parameterizations (Cetin et al., 2024), temporal abstraction (Azad et al., 2026), or the successor quantity (Dubail et al., 2025). These methods improve convergence, stability, or expressiveness, but do not address the failure we study: a difference between actions that matter for control can receive little weight under an objective that fits the entire SM using least squares. Density-FB changes how the SM is estimated and learns it as a distribution in order to mitigate this failure.

Normalized and alternative models of future occupancy. Several methods avoid direct regression of future occupancy. C-learning estimates discounted future-state occupancy through recursive classification (Eysenbach et al., 2021). Contrastive RL learns goal-conditioned values through contrastive estimation (Eysenbach et al., 2022). TD-InfoNCE represents future occupancy using normalized exponentiated scores trained with a categorical temporal-difference objective (Zheng et al., 2024), and is closest to our construction. However, our setting is reward-free pretraining of a family of policies and their corresponding SM approximations for rewards specified after training, which requires both a successor representation and a corresponding downstream value-extraction procedure. Other approaches instead change the predictive object by learning geometric or latent predictive representations (Park et al., 2024; Jajoo et al., 2026; Bagatella et al., 2026a) or by directly generating discounted future states (Janner et al., 2020; Farebrother et al., 2025; Zheng et al., 2026).

Categorical prediction and downstream task inference. Categorical objectives improve value learning by replacing scalar regression with prediction over a discrete support (Bellemare et al., 2017; Farebrother et al., 2024). In Density-FB, the categories are instead future states sampled from the offline distribution, so their probabilities define a normalized SM. This normalization breaks the standard FB identity that maps rewards directly to task vectors. Methods such as Soft FB (Bagatella et al., 2026b) and optimistic task inference (Rupf et al., 2026) modify how downstream tasks are selected from a pretrained FB representation. Density-FB changes the successor representation, evaluates rewards under the learned future-state distribution, and infer the policy-conditioning vector separately.

Control-relevant interactions and bottlenecks. Work on RL for environemtns with interaction-rich structure (Mohan et al., 2024) highlights the importance of decisions through which an agent can influence its environment. HyPE, COInS, and SkiLD use object or factor interactions to discover reusable skills (Chuck et al., 2020; 2024; Wang et al., 2024). Causal influence detection and ELDEN use state-dependent influence and uncertainty about local dependencies to guide learning and exploration (Seitzer et al., 2021; Wang et al., 2023b). HInt uses inferred interactions to guide hindsight relabeling, addressing the dominance of trajectories in which objects remain unchanged (Chuck et al., 2025). These approaches motivate attention to consequential interactions, such as establishing contact with an object before manipulating it. Our control-bottleneck analysis examines whether an SM approximation preserves the differences in long-term outcomes between actions at such decisions. Even when relevant transitions are present in the offline data, a low-rank least-squares fit can erase the occupancy distinction needed to rank the actions. Density-FB addresses this approximation problem by learning normalized future-state distributions, without requiring explicit interaction detection or a skill hierarchy.

B Theoretical Analysis

B.1 Proof of Lemma 1

Setup. Fix z and write $\pi = \pi_z$. Let ρ be the reference probability distribution, r a bounded measurable reward, and $B \in L^2(\rho; \mathbb{R}^d)$ fixed. At a bottleneck entry state s, choose $a_+ \in \mathcal{A}_+(s)$ and

$a_- \in \mathcal{A}_-(s)$ such that $Q_r^\pi(s, a_+) > Q_r^\pi(s, a_-)$. For each $a \in \{a_+, a_-\}$, write

$$M_a := M^\pi(s, a, \cdot), \qquad M_a(ds^+) = m_a(s^+)\,\rho(ds^+), \qquad m_a \in L^2(\rho).$$

The forward vectors minimize their squared errors independently:

$$F^*(s, a, z) \in \operatorname*{arg\,min}_{f \in \mathbb{R}^d} \int_{\mathcal{S}} \big[m_a(s^+) - f^\top B(s^+)\big]^2 \,\rho(ds^+).$$

Define the fitted densities and values by

$$\widehat{m}_a(s^+) := F^*(s, a, z)^\top B(s^+), \qquad \widehat{Q}_r^\pi(s, a) := \frac{1}{1-\gamma} \int_{\mathcal{S}} r(s^+)\widehat{m}_a(s^+)\,\rho(ds^+).$$

Proof. **Step 1: Characterize the optimal fit.** With B fixed, fitting retains the component of each successor density that lies in the span of the backward features. To see this, differentiating the squared-error objective gives

$$G_B F^*(s, a, z) = \int_{\mathcal{S}} B(s^+)\, M_a(ds^+), \qquad G_B := \int_{\mathcal{S}} B(s^+)B(s^+)^\top \,\rho(ds^+).$$

Equivalently, for every $j = 1, \ldots, d$,

$$\int_{\mathcal{S}} B_j(s^+)\big[m_a(s^+) - \widehat{m}_a(s^+)\big]\,\rho(ds^+) = 0.$$

Thus the residual is orthogonal to the backward feature span. Writing Π_B for the $L^2(\rho)$ orthogonal projection onto $\operatorname{span}\{B_1, \ldots, B_d\}$, we obtain

$$\widehat{m}_a = \Pi_B m_a \qquad \rho\text{-almost everywhere.}$$

The fitted density is therefore unique, even if G_B is singular and the minimizing forward vector is not unique.

Step 2: Identify when the action distinction is lost. The two fits coincide exactly when the backward features have the same expectations under the two SMs. Let

$$v := F^*(s, a_+, z) - F^*(s, a_-, z).$$

Subtracting the optimality equations yields

$$G_B v = \int_{\mathcal{S}} B(s^+)\, M_{a_+}(ds^+) - \int_{\mathcal{S}} B(s^+)\, M_{a_-}(ds^+).$$

If these feature expectations agree, then $G_B v = 0$, and hence

$$\int_{\mathcal{S}} \big[\widehat{m}_{a_+}(s^+) - \widehat{m}_{a_-}(s^+)\big]^2 \,\rho(ds^+) = v^\top G_B v = 0.$$

The fitted densities therefore coincide ρ-almost everywhere. Conversely, if they coincide, then

$$G_B v = \int_{\mathcal{S}} B(s^+)\big[\widehat{m}_{a_+}(s^+) - \widehat{m}_{a_-}(s^+)\big]\,\rho(ds^+) = 0,$$

so their feature expectations agree. This proves

$$\mathbb{E}_{M_{a_+}}[B] = \mathbb{E}_{M_{a_-}}[B] \quad \Longleftrightarrow \quad \widehat{m}_{a_+} = \widehat{m}_{a_-} \quad \rho\text{-almost everywhere.}$$

Step 3: Relate the lost distinction to action values. The same projection can be applied to the reward: evaluating the fitted density against r is equivalent to evaluating the exact density against $\Pi_B r$. Indeed, orthogonality gives

$$\begin{aligned}(1-\gamma)\widehat{Q}_r^\pi(s, a) &= \int_{\mathcal{S}} r(s^+)\widehat{m}_a(s^+)\,\rho(ds^+) \\ &= \int_{\mathcal{S}} (\Pi_B r)(s^+)\widehat{m}_a(s^+)\,\rho(ds^+) \\ &= \int_{\mathcal{S}} (\Pi_B r)(s^+) m_a(s^+)\,\rho(ds^+).\end{aligned}$$

The second equality uses that $\widehat{m}_a$ lies in the backward feature span; the third uses that $\Pi_B r$ does.

Under the equivalent conditions of Step 2, the fitted densities coincide, so

$$\widehat{Q}_r^\pi(s,a_+) - \widehat{Q}_r^\pi(s,a_-) = \frac{1}{1-\gamma}\int_{\mathcal{S}} r(s^+)\big[\widehat{m}_{a_+}(s^+) - \widehat{m}_{a_-}(s^+)\big]\,\rho(ds^+) = 0.$$

For the chosen action pair, however, (2) gives

$$Q_r^\pi(s,a_+) - Q_r^\pi(s,a_-) = \frac{1}{1-\gamma}\int_{\mathcal{S}} r(s^+)\big[m_{a_+}(s^+) - m_{a_-}(s^+)\big]\,\rho(ds^+) > 0.$$

Thus fitting discards the component of the successor densities that carries their true action-value distinction. □

B.2 Proof of Proposition 1

Setup. Both the normalized SMs $M^\pi(s,a,\cdot)$ and the Density-FB predictions $\widehat{M}_\theta(s,a,z,\cdot)$ are probability measures. Define the fitted values by

$$\widehat{Q}_r^\pi(s,a) := \frac{1}{1-\gamma}\int_{\mathcal{S}} r(s^+)\,\widehat{M}_\theta(s,a,z,ds^+).$$

The assumed positive gap implies that r is non-constant, so $0 < R < \infty$.

Proof. **Step 1: Convert distribution error into value error.** For bounded rewards, closeness of the successor distributions controls the error in their predicted returns. The bounded-reward expectation bound and Pinsker's inequality give, for each $a \in \{a_+, a_-\}$,

$$\left|\widehat{Q}_r^\pi(s,a) - Q_r^\pi(s,a)\right| \le \frac{R}{1-\gamma}\sqrt{\frac{1}{2}D_{\mathrm{KL}}\Big(M^\pi(s,a,\cdot)\,\Big\|\,\widehat{M}_\theta(s,a,z,\cdot)\Big)}.$$

Here KL divergence uses natural-log.

Step 2: Compare the errors with the true gap. The stated KL threshold makes each value error smaller than half the true action-value gap:

$$\left|\widehat{Q}_r^\pi(s,a) - Q_r^\pi(s,a)\right| < \frac{R}{1-\gamma}\sqrt{\frac{(1-\gamma)^2\delta^2}{4R^2}} = \frac{\delta}{2}, \qquad a \in \{a_+, a_-\}.$$

Step 3: Preserve the action ordering. Even if both errors reduce the predicted gap, their combined magnitude is insufficient to erase it:

$$\begin{aligned}
&\widehat{Q}_r^\pi(s,a_+) - \widehat{Q}_r^\pi(s,a_-) \\
&\quad \ge \delta - \left|\widehat{Q}_r^\pi(s,a_+) - Q_r^\pi(s,a_+)\right| - \left|\widehat{Q}_r^\pi(s,a_-) - Q_r^\pi(s,a_-)\right| \\
&\quad > \delta - \frac{\delta}{2} - \frac{\delta}{2} = 0.
\end{aligned}$$

□

B.3 Convergence under exact fitting

We adapt the exact-expectation convergence argument of TD InfoNCE (Zheng et al., 2024, Appendix A) and establish a KL contraction that connects the fitted updates to Proposition 1.

Setup. Assume finite state and action spaces, $0 < \gamma < 1$, a reference distribution ρ with positive mass on every state, and a fixed continuation policy $\pi = \pi_z$. All expectations and normalizing constants are evaluated exactly. Write

$$\widehat{M}_k(s,a,\cdot) := \widehat{M}_{\theta_k}(s,a,z,\cdot),$$

and initialize $\widehat{M}_0(s,a,\cdot)$ with positive mass on every state for every (s,a).

At iteration k, form the Bellman targets

$$T_k(s,a,\cdot) := (1-\gamma)P(\cdot \mid s,a) + \gamma \mathbb{E}_{\substack{s'\sim P(\cdot|s,a)\\ a'\sim\pi(\cdot|s')}} \left[\widehat{M}_k(s',a',\cdot)\right].$$

Assume that these targets are jointly representable by Density-FB at the chosen feature dimension. With the targets held fixed, globally minimize the weighted sum of categorical fitting losses, assigning positive weight to every state-action pair. Both the forward and backward parameters may change during fitting. Then

$$\max_{s,a} D_{\mathrm{KL}}\Big(M^\pi(s,a,\cdot) \,\Big\|\, \widehat{M}_k(s,a,\cdot)\Big) \longrightarrow 0.$$

Consequently, for any fixed bounded reward and any bottleneck action pair with a strictly positive true value gap, the fitted ordering is correct at every sufficiently large iteration.

Proof. **Step 1: Recover the Bellman target.** Exact cross-entropy fitting recovers a representable target distribution. Writing

$$\widehat{M}_\theta(s,a,z,ds^+) = h_\theta(s,a,z,s^+)\,\rho(ds^+),$$

the loss for each state-action pair satisfies

$$\begin{aligned} &-\int_{\mathcal{S}} \log h_\theta(s,a,z,s^+)\, T_k(s,a,ds^+) \\ &\qquad = D_{\mathrm{KL}}\Big(T_k(s,a,\cdot) \,\Big\|\, \widehat{M}_\theta(s,a,z,\cdot)\Big) + \text{constant}, \end{aligned}$$

where the constant is independent of θ. Joint realizability allows all KL terms to vanish simultaneously. Since every fitting weight is positive, every global minimizer therefore satisfies

$$\widehat{M}_{k+1}(s,a,\cdot) = T_k(s,a,\cdot) \qquad \text{for every } (s,a).$$

Step 2: Contract the KL error. The true SM and the fitted update have the same immediate-transition component. Their discrepancy comes only from the continuation predictions. Define

$$K_k := \max_{s,a} D_{\mathrm{KL}}\Big(M^\pi(s,a,\cdot) \,\Big\|\, \widehat{M}_k(s,a,\cdot)\Big).$$

Using (3), Step 1, and joint convexity of KL,

$$\begin{aligned} &D_{\mathrm{KL}}\Big(M^\pi(s,a,\cdot) \,\Big\|\, \widehat{M}_{k+1}(s,a,\cdot)\Big) \\ &\quad \le \gamma \mathbb{E}_{\substack{s'\sim P(\cdot|s,a)\\ a'\sim\pi(\cdot|s')}} \left[D_{\mathrm{KL}}\Big(M^\pi(s',a',\cdot) \,\Big\|\, \widehat{M}_k(s',a',\cdot)\Big)\right] \\ &\quad \le \gamma K_k. \end{aligned}$$

Taking the maximum over (s,a) gives

$$K_{k+1} \le \gamma K_k.$$

Step 3: Reach the preservation threshold. Positive initialization and finite state and action spaces ensure $K_0 < \infty$. Hence

$$K_k \le \gamma^k K_0 \longrightarrow 0.$$

Fix a bounded reward r and a bottleneck action pair $a_+ \in \mathcal{A}_+(s)$, $a_- \in \mathcal{A}_-(s)$ with

$$\delta := Q_r^\pi(s,a_+) - Q_r^\pi(s,a_-) > 0.$$

Let

$$R := \sup_{\mathcal{S}} r - \inf_{\mathcal{S}} r, \qquad \varepsilon := \frac{(1-\gamma)^2\delta^2}{2R^2}.$$

There exists k_0 such that, for every $k \ge k_0$,

$$D_{\mathrm{KL}}\Big(M^\pi(s,a,\cdot) \,\Big\|\, \widehat{M}_k(s,a,\cdot)\Big) \le K_k < \varepsilon, \qquad a \in \{a_+, a_-\}.$$

Thus Proposition 1 gives

$$\widehat{Q}^\pi_{k,r}(s,a_+) > \widehat{Q}^\pi_{k,r}(s,a_-), \qquad k \ge k_0,$$

where $\widehat{Q}^\pi_{k,r}$ integrates r against $\widehat{M}_k$ as in (2). □

B.4 SUCCESSOR-FEATURE INTERPRETATION OF THE TRAINING REWARD

Setup. Fix z, $\pi = \pi_z$, and a bounded measurable feature map $B_\theta : \mathcal{S} \to \mathbb{R}^d$. Consider the training reward

$$r_z(s^+) := \frac{B_\theta(s^+)^\top z}{\sqrt{d}}.$$

With these reward features held fixed, the value of r_z has the standard successor-feature representation (Barreto et al., 2017).

Proof. **Step 1: Express successor features using the SM.** Successor features record the expected discounted sum of future features. By the definition of the normalized SM,

$$\begin{aligned}\psi^\pi(s,a) &:= \mathbb{E}_\pi\left[\sum_{t\geq 0}\gamma^t \frac{B_\theta(s_{t+1})}{\sqrt{d}} \,\middle|\, s_0 = s, a_0 = a\right] \\ &= \frac{1}{1-\gamma}\int_{\mathcal{S}} \frac{B_\theta(s^+)}{\sqrt{d}}\, M^\pi(s,a,ds^+).\end{aligned}$$

Boundedness of B_θ justifies exchanging the expectation and the discounted sum.

Step 2: Recover the reward value. Since the reward is linear in these features, its value is linear in their discounted occupancies:

$$\begin{aligned}Q^\pi_{r_z}(s,a) &= \frac{1}{1-\gamma}\int_{\mathcal{S}} \frac{B_\theta(s^+)^\top z}{\sqrt{d}}\, M^\pi(s,a,ds^+) \\ &= \psi^\pi(s,a)^\top z.\end{aligned}$$

Step 3: Identify the features estimated by the readout. For candidate future states s_j^+, the probabilities $p_j(s,a,z)$ in (6) define the estimate

$$\widehat{\psi}_\theta(s,a,z) := \frac{1}{1-\gamma}\sum_j p_j(s,a,z)\frac{B_\theta(s_j^+)}{\sqrt{d}}.$$

Substituting r_z into (8) gives

$$\begin{aligned}\widehat{Q}_{\theta,r_z}(s,a,z) &= \frac{1}{1-\gamma}\sum_j p_j(s,a,z)\, r_z(s_j^+) \\ &= \widehat{\psi}_\theta(s,a,z)^\top z.\end{aligned}$$

Thus the readout estimates successor features by averaging future-state features under the predicted probabilities, then applies the reward weights z. □

For the implemented centered reward, the same identities hold with B_θ replaced by $B_\xi - \hat{\mu}$, holding B_ξ and $\hat{\mu}$ fixed.

C IMPLEMENTATION DETAILS

This appendix describes task inference, successor networks, policy extraction, and zero-shot evaluation. Benchmark-specific task conditioning and sampling are specified in Appendix C.7, with hyperparameters in Table 4 and Appendix C.7.

C.1 INFERRING A TASK INDEX FROM REWARD SAMPLES

Given N_r states $s_j^+ \sim \rho$ and rewards $r_j = r(s_j^+)$, write $b_j = B_\xi(s_j^+)$ and $\bar{r} = N_r^{-1}\sum_j r_j$, where B_ξ is the slowly updated backward encoder used to define the training rewards. Let $\widehat{b} = \hat{\mu}$ and $\widehat{C}$ be the running mean and covariance of its features, maintained during pretraining. Set $v = \mathrm{tr}(\widehat{C})/d$

and $\lambda = 0.01v$. Form $\widetilde{C}$ by replacing eigenvalues of $\widehat{C}$ below $10^{-6}v$ by that threshold. The encoder solves

$$(\widetilde{C} + \lambda I)c_r = \frac{1}{N_r}\sum_{j=1}^{N_r}(b_j - \widehat{b})(r_j - \bar{r}), \qquad z_r = \sqrt{d}\,\frac{c_r}{\|c_r\|}. \tag{11}$$

If $\|c_r\| < 10^{-6}$, we return a fixed task vector of norm $\sqrt{d}$ instead.

C.2 Successor networks

The default successor networks are MLPs. Each hidden layer uses Linear→GELU→LayerNorm, followed by a linear output layer. The backward network B_θ has one output head and no output normalization or orthogonality regularizer. For task-vector conditioning, the forward network receives $[s, a, z]$ and has two heads,

$$F_\theta(s, a, z) \in \mathbb{R}^{2\times d}. \tag{12}$$

Direct goal conditioning replaces z with the goal state g. The ExoRL architecture and backward-output normalization differ as specified in Appendix C.7. Each forward head defines its own normalized successor distribution using Equation (5). The representation loss in Equation (7) is computed independently for the two heads and then averaged. Target networks $\bar{F}$ and $\bar{B}$ are updated after every gradient step using an exponential moving average with rate η. The bootstrap action is produced by the current actor and treated as a stop-gradient when constructing the successor target.

Combining the two heads. Each forward head $h \in \{1, 2\}$ defines its own normalized distribution $p^{(h)}(s, a, z)$ over the candidate next states, and each head is trained against its own target. The two heads are combined only after a reward has been integrated: for a reward r the per-head values are $q^{(h)}(s, a, z) = \sum_j p_j^{(h)}(s, a, z)\, r(s'_j)$ and the value used by the actor is

$$q(s, a, z) := \min\left\{q^{(1)}(s, a, z),\, q^{(2)}(s, a, z)\right\}. \tag{13}$$

Because each head remains a probability distribution, adding a constant to the reward adds the same constant to both heads' values and hence to q, so action rankings are invariant to reward shifts; taking the minimum of the two probability vectors instead would not have this property, since its total mass depends on the action. The normalization and fixed-point results of Section 5 apply to each head.

C.3 Policy extraction

We write the actor objectives for task-vector conditioning. For direct goal conditioning, replace z with g and r_z with the goal-reaching reward specified in Appendix C.7. For each forward head h, we compute

$$q^{(h)}(s, a, z) := \sum_{j=1}^{n} p_j^{(h)}(s, a, z) r_z(s'_j). \tag{14}$$

The value used by the actor is the minimum of the two heads' values, $q := \min_h q^{(h)}$, as in Equation (13). We normalize its scale using

$$s_q := \mathrm{sg}\left[\mathbb{E}\,|q - \mathbb{E}[q]| + \varepsilon\right], \qquad \varepsilon = 10^{-6}. \tag{15}$$

This normalization controls the scale of the value term relative to the behavior-regularization term.

Flow-based actor. For OGBench, MetaWorld, and ManiSkill, we use the flow-based policy extractor from TD-JEPA (Bagatella et al., 2026a), based on Flow Q-Learning (Park et al., 2025b). A goal-agnostic velocity field $v_\psi(s, a, t)$ models the behavior policy. Given

$$x_0 \sim \mathcal{N}(0, I), \qquad x_1 = a, \qquad t \sim \mathcal{U}[0, 1], \tag{16}$$

we define

$$x_t := (1 - t)x_0 + tx_1, \tag{17}$$

and train the velocity field with

$$\mathcal{L}_{\text{flow}} := \mathbb{E}\left[\|v_\psi(s, x_t, t) - (x_1 - x_0)\|^2\right]. \tag{18}$$

Let

$$a_{\text{BC}} := \text{ODE}_\psi(s, \eta), \qquad \eta \sim \mathcal{N}(0, I), \tag{19}$$

denote the action obtained by integrating the learned velocity field for K Euler steps.

The one-step actor $\pi_\omega(s, z; \eta)$ is trained using

$$\mathcal{L}_\pi := -\frac{\mathbb{E}\left[q\left(s, \pi_\omega(s, z; \eta), z\right)\right]}{s_q} + \lambda_{\text{BC}}\mathbb{E}\left[\|\pi_\omega(s, z; \eta) - a_{\text{BC}}\|^2\right]. \tag{20}$$

The velocity field and the actor are optimized separately using Equations (18) and (20). Removing the value term from Equation (20) gives the corresponding behavior-cloning policy.

During the actor update, the successor networks are fixed. Gradients pass through the action input of F_θ to the actor, but the parameters of F_θ and B_θ are not updated.

DDPG+BC actor. For ExoRL, we use the DDPG+BC extractor with a Gaussian policy

$$\pi_\omega(\cdot \mid s, z) = \mathcal{N}\left(\mu_\omega(s, z), \sigma^2 I\right), \tag{21}$$

with fixed standard deviation σ.

Its objective is

$$\mathcal{L}_\pi = -\frac{\mathbb{E}\left[q(s, \mu_\omega(s, z), z)\right]}{s_q} - \alpha\mathbb{E}\left[\log \pi_\omega(a \mid s, z)\right]. \tag{22}$$

C.4 Ablation: the actor's value readout

We compare the actor's distribution-weighted reward readout with the linear readout used by FB. The critic architecture and cross-entropy objective are unchanged; only the value readout used to train the actor is replaced. For the goal-conditioned OGBench experiments, the critic remains conditioned on g, and the alternative readout is $F(s, a, g)^\top B(g)$. Here $B(g)$ supplies the readout weights, while g remains the conditioning input.

Table 3 reports success averaged over evaluations at 3, 4, and 5×10^5 training steps. The linear readout achieves near-zero success on all three environments. Softmax probabilities are invariant to adding a constant to all candidate scores for a given state, action, and task, so the cross-entropy objective does not calibrate absolute scores as action values. In these runs, the linear readout grows from approximately 1 to 100, while the distribution-weighted value remains near 0.03.

Table 3: **The actor needs the distribution-weighted value.** Success (%) of Density-FB with the actor trained on the reward averaged under the predicted successor distribution (as in table 1) or on the linear readout, with the same critic. Mean ± SE over seeds of the mean over the evaluations at 3, 4 and 5×10^5 steps.

	Goal-conditioned		Reward inference	
Environment	distribution-weighted	$F(s, a, g)^\top B(g)$	distribution-weighted	$F(s, a, z)^\top z$
`cube-single`	93.6 ±1.6	0.3 ±0.2	82.4 ±2.7	55.1 ±6.9
`cube-double`	61.5 ±2.8	0.0 ±0.0	20.5 ±0.8	12.9 ±0.4
`scene`	62.2 ±1.4	0.1 ±0.1	48.8 ±2.2	31.1 ±3.6

C.5 Reward inference and zero-shot evaluation

All evaluations use frozen successor and policy networks. No gradient updates or additional environment interaction for task adaptation are performed.

For reward inference, we evaluate the downstream reward on states sampled from the offline dataset and infer z_r using the regression in Equation (11). We use $N_r = 10^4$ states for MetaWorld and ManiSkill and $N_r = 5 \times 10^4$ for ExoRL. Task inference uses the same encoder and feature statistics as the goal-derived task vectors during pretraining (Appendix C.7). The agent then executes $\pi_\omega(\cdot \mid s, z_r)$. For OGBench, the agent instead executes the goal-conditioned policy $\pi_\omega(\cdot \mid s, g)$ using the goal supplied by the environment.

Value diagnostics. To evaluate the critic, we sample a fixed pool $U_P = \{u_1, \ldots, u_P\}$ from the dataset next-state marginal ρ and precompute $B_\theta(u_j)$. Each forward head defines probabilities $p_j^{(h)}$ over this pool. For a reward r, the finite-pool value estimate is

$$\widehat{q}_{\theta,r}(s, a, z) := \min_h \sum_{j=1}^{P} p_j^{(h)}(s, a, z)\, r(u_j). \tag{23}$$

Thus heads are combined after reward integration, as in Equation (13). For direct goal conditioning, g replaces z. The pool contains $P = 4096$ states, is sampled once with a fixed seed, and is reused throughout evaluation. These values are used for critic diagnostics; policy execution requires only the frozen actor and does not compute a value.

C.6 Hyperparameters

C.7 Task conditioning and sampling

Density-FB supports both direct goal conditioning and conditioning on latent task vectors. Based on empirical performance, the reported OGBench experiments use goal conditioning, while MetaWorld, ManiSkill, and ExoRL use task vectors inferred from rewards.

Task-vector conditioning. Each critic and actor training row carries a task vector $z \in \mathbb{R}^d$ with $\|z\| = \sqrt{d}$. With probability $1 - p_{\text{rand}}$, we sample a future state from the row's trajectory as a hindsight goal and evaluate its goal-reaching reward on the batch's candidate next states. We encode these reward samples using the regression in Appendix C.7, with the slow backward encoder B_ξ and its running feature statistics. With probability p_{rand}, we instead sample z uniformly from the sphere of radius $\sqrt{d}$. Both branches are treated as constants: gradients do not flow through the sampled task vector. The same reward-to-task encoding is used for downstream reward inference.

The actor uses the centered latent reward

$$r_z(s^+) := \frac{(B_\xi(s^+) - \hat{\mu})^\top z}{\sqrt{d}},$$

with the same encoder and mean used for task inference. In preliminary task-vector-conditioned experiments on `cube-single`, using the fast target encoder to compute this reward substantially reduced success. Replacing the latent reward with the exact goal kernel on goal-derived rows did not improve success. We therefore retain the latent reward throughout task-vector-conditioned training.

Direct goal conditioning. The OGBench experiments condition the critic and actor directly on the goal state g. Actor goals are future states drawn uniformly from the corresponding trajectory. For a candidate next state s_j^+, the actor's reward is the Gaussian kernel

$$\exp\left(-\frac{1}{2}\sum_k \left\|(s_{j,k}^+ - g_k)/\epsilon_k\right\|^2\right),$$

normalized over the candidates. Here k indexes object-coordinate blocks and ϵ_k specifies their widths. The actor maximizes this reward averaged under the predicted successor distribution.

Critic goals are sampled from the current state, a future state of the same trajectory, or a random dataset state with probabilities 0.2, 0.5, and 0.3, respectively. The object blocks are the cube position for `cube-single` and both cube positions for `cube-double`, with $\epsilon = 0.4$. For `scene`, the reward additionally includes the drawer, window, and two button blocks, with widths 0.5, 0.5, 0.72, and 0.6. At evaluation, the policy receives the goal supplied by the environment.

Task-prior fraction. The task-vector-conditioned MetaWorld and ManiSkill experiments use $p_{\text{rand}} = 1/2$. ExoRL uses domain-specific fractions, specified below.

ExoRL configuration. We use $p_{\text{rand}} = 1$ for `walker` and `cheetah`, and $p_{\text{rand}} = 1/2$ for `quadruped` and `point-mass`. All ExoRL experiments use RND datasets, state observations, 10^6 gradient steps, five training seeds, and reward inference from 5×10^4 offline states. They use the DDPG+BC extractor with $\alpha = 0$. The settings in Table 4 are modified by the domain-specific choices below. The base settings, overridden below where specified, are: successor towers $(512, 512, 512)$, backward towers $(256, 256)$ with the output projected onto the sphere, target rate 0.005, learning rate 10^{-4}, actor learning rate 10^{-4}, discount 0.98, and a hit kernel over the full observation with the radius calibrated to the hit rate of the manipulation kernel (`walker` $0:24$ at 15.9, `quadruped` $0:78$ at 21.1). Per domain: `point-mass` uses discount 0.99, actor learning rate 10^{-6} and a kernel over the two position coordinates at radius 0.03, as every reference implementation does on this domain; `walker` uses task vectors drawn only from the sphere during training, the actor's value normalizer, a single successor tower of width 1024 and target rate 0.001, which raised its return from 724.6 to 750.2 and lowered `quadruped` and `point-mass`, so those two keep the shared base recipe; `quadruped` additionally restricts the hit kernel to the three torso-velocity coordinates (columns 44–46, radius 0.11) and the torso-upright coordinate (column 47, radius 0.001), which raised its return from 655.9 to 760.5; `cheetah` uses the `walker` settings with the hit kernel restricted to the nine velocity coordinates of the observation (columns 8–16, radius 7.26 at the same hit rate), which raised its return from 431 to 540, against 532 for a kernel over the forward-speed coordinate alone.

D METAWORLD AND MANISKILL DATASETS

This appendix describes how the offline datasets used in Section 6.2 were collected. The same environment wrapper is used for data collection and for evaluation, so the observation and action spaces, the control frequency, and the success criterion are identical in both.

Common protocol. Each dataset consists of N fixed-length trajectories of T environment steps, giving $N \times T \geq 10^6$ transitions per task (Table 5). Every trajectory pursues a single goal drawn once at reset from the environment's own goal distribution; the goal is never re-drawn within a trajectory. The behavior policy typically completes the task early and then holds its final configuration, so the remainder of the trajectory records the object at the goal. The fraction of transitions in which the object remains at the goal (the at-goal fraction) ranges from 0.76 to 0.95 across tasks; this is the persistent-occupancy structure discussed in Section 4. Each dataset is stored as one packed array file per task with fields `observation`, `action`, `next_observation`, `reward`, `terminal` (all zero; the horizon is fixed), `timeout` (set at $t = T - 1$), and `success`; MetaWorld additionally stores `achieved_goal` and `goal_switch`. Actions are clipped to $[-1, 1]$.

MetaWorld (`push`, `bin-picking`, `box-close`). We use the goal-observable V2 environments behind a wrapper replicating the collection setup of Hansen et al. (2026): action repeat 2 with summed rewards, a reset consisting of `env.reset()` followed by one zero-action step (which fills the previous-frame half of the 39-dimensional state), and 4-dimensional actions. The environment's internal path-length truncation is disabled so that every trajectory runs for the full $T = 500$ wrapper steps (1000 simulator steps). Each task has $N = 2010$ trajectories (1,005,000 transitions), collected in 30 shards of 67 episodes with per-episode seeding of the global random number generator, from which MetaWorld draws its reset randomization.

The behavior policies are the scripted oracles shipped with MetaWorld. For `push` we use `SawyerPushV2Policy`, which follows the goal channel of the observation. For the two fixture tasks, the transport policies (`SawyerPickPlaceV2Policy` for `bin-picking`, `SawyerBoxCloseV2Policy` for `box-close`) are augmented with a deterministic grasp primitive that repairs the scripted grasp under action repeat 2: the gripper is aligned above the object to within 1.5–2 cm with hysteresis, descends open to the grasp depth (the mocap floor at $z \approx 0.065$ for the cube; 5 cm above the lid for the handle), closes, and hands control back to the scripted policy once the finger separation enters the measured carry band (0.35–0.60 for the cube, 0.25–0.40 for the lid). Per-episode Gaussian action noise is added with standard deviation $\sigma \sim \mathcal{U}[0, \sigma_{\max}]$, $\sigma_{\max} = 0.02$ for `push` and `bin-picking` and 0.01 for `box-close`.

The goal protocol is task-specific. `push`: one target per episode from the environment's own reset distribution (goal-space xy with z drawn from the empirical pool of reset heights). `bin-picking`: one placement into the goal bin, after which the object is held in the closed gripper. `box-close`: one placement of the lid onto the box, after which the lid is released and the hand hovers above it (holding the lid with the gripper closed makes the scripted controller oscillate). Fixtures are restored to their canonical pose at every reset and clamped to a wall-safe region (box $y \in [0.68, 0.78]$). Because the environment clips the goal dimensions of the observation (indices 36–38) to its goal-space box, which is degenerate for the fixture tasks, the true target is written into those dimensions on every stored frame. Success is $\|\text{obj} - \text{target}\| < r$ with $r = 0.06$, 0.06, and 0.08 m for the three tasks. Trajectories in which the object leaves the workspace (`push`: $|x| > 0.30$; `bin-picking`: $|x| > 0.45$; `box-close`: $|x| > 0.38$ or $y \notin [0.35, 0.90]$) are rejected at the first offending step and re-rolled, with at most 8 retries. The measured at-goal fractions are 0.921, 0.895, and 0.812, and no rejected trajectory was released into the datasets.

ManiSkill (`push-cube`, `pick-cube`, `stack-cube`, `poke-cube`, `lift-peg`). We use the ManiSkill3 (version 3.0.1) state-observation environments with the per-task control modes of Hansen et al. (2026) (`pd_ee_delta_pos` for `pick-cube`, `pd_ee_delta_pose` otherwise), wrapped with `CPUGymWrapper(ignore_terminations=True)`. The horizon is overridden to $T = 125$ steps, five times the 25-step evaluation horizon, so that the post-completion hold is recorded. Each task has $N = 8000$ trajectories (1,000,000 transitions), collected in 8 shards of 1000 episodes with `env.reset(seed)` seeding.

The behavior policies are the released multitask TD-MPC2 checkpoints of Hansen et al. (2026) (model size B), with observations zero-padded to 128 dimensions and the 16-dimensional action head truncated to the task's action dimension. Actions are drawn from the stochastic policy head (`eval_mode=False`), with planning disabled (`mpc=False`) and no additional exploration noise or ϵ-mixing. The environment's own goal distribution is used unchanged. Tasks complete in roughly 10–20 steps and the policy then holds. The measured episode success rates are 0.997, 0.990, 0.940, 0.948, and 0.921, and the at-goal fractions are 0.949, 0.906, 0.756, 0.858, and 0.783.

E Bottleneck experiment details

Process and policies. All experiments use the nine-state transition graph in Figure 2, with two deterministic actions per state, $\gamma = 0.8$, and a uniform reference distribution $\rho(s^+) = 1/9$. Entering A (state 3) gives reward 1, every tail state (5–8) gives 0.5, and all other states give 0. The main example, P9, uses a continuation policy that chooses each action with probability $1/2$. For five additional settings, R1–R5, we keep the transitions fixed and independently draw the probability of the first action at each state from $\mathcal{U}(0.25, 0.75)$. The sampling procedure retains a draw only if the true action-value difference is at least 0.5 and the best rank-2 linear fit retains less than half of it. All five initial draws passed these conditions. Because every tail state carries the same reward, the value of the tail does not depend on the policy, so the true action-value difference at state 2 is exactly 1 in all six settings. Table 6 lists the six policy settings.

Training and the fixed-feature check. We use exact expectations and bootstrapped target networks, with ten training runs per setting and feature dimensions $d = 1, \ldots, 8$. For the check of Lemma 1, we freeze the trained backward features and refit the two bottleneck forward vectors to the exact successor densities by ρ-weighted least squares. We evaluate the fitted action-value difference using (2).

Objectives of the ablation arms. Let M be the exact successor matrix (one row per state-action pair, mass $1/(1-\gamma)$), $\bar{M} = (1-\gamma)M$ its normalized rows, $\rho = 1/n$, and $X = nM$ the density ratios FB fits. All rows carry equal weight in every objective. FB fits $F_i B^\top$ to X_i by squared error; its optimum is the rank-d truncated singular value decomposition of X, and the value is read as $F_i^\top z$ with $z = \rho\, B^\top r$. *FB, normalized target* fits $F_i B^\top$ to $\bar{M}_i$ with the same squared error and readout rescaled by $n/(1-\gamma)$, so its optimum coincides with FB's. *Softmax + squared error* predicts $p_i = \mathrm{softmax}(F_i B^\top)$ and minimizes $\frac{1}{2}\sum_j (p_{ij} - \bar{M}_{ij})^2$; Density-FB minimizes the cross-entropy $-\sum_j \bar{M}_{ij} \log p_{ij}$; both read the value as $\sum_j p_{ij} r_j/(1-\gamma)$. *Density-FB, linear readout* keeps Density-FB's trained F and B but reads $F_i^\top z$ as FB does. *Density-FB, soft immediate target* replaces

the one-hot immediate term by a kernel over graph distance, $\exp(-\mathrm{dist}^2/(2 \cdot 0.5^2))$ row-normalized, so its fixed point is $\bar{M}$ smoothed by that kernel. TD-trained arms use the bootstrapped targets of these objectives, $(1-\gamma)P_i + \gamma\, P_i\, \bar{p}^{\pi}_{\text{target}}$ for the softmax arms and the corresponding squared TD error for the linear arms (with FB's orthogonality penalty at coefficient 1), a target network with rate 0.005, Adam, 5×10^4 steps, five seeds and two learning rates (10^{-3}, 10^{-2}), reported as the mean over the ten runs.

Results across policies. Table 7 summarizes the feature dimension needed to reach 90% of the true action-value difference in each setting. Both softmax variants reach this threshold with three or four features in every setting, while FB needs eight, the largest dimension below full rank, and does not reach it at all on R4 and R5. FB reads less than 10% of the difference with up to six features in every setting.

Features learned by FB. Figure 6 shows the features of a trained FB critic with two features, where they can be drawn in the plane. FB gives the rewarded state A (3) exactly the same features as the unrewarded state B (4), to numerical precision in all ten runs. The average feature vector of the future after a_+ therefore equals that after a_-, which is the condition of Lemma 1, and FB predicts the same future and the same value (2.14) for both actions, against exact values of 3 and 2.

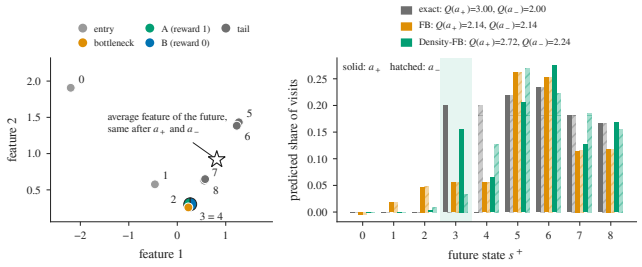


Figure 6: **FB's features cannot separate the two branches.** Backward features $B(s)$ of one trained FB critic with two features; the half-green, half-blue marker is state A (3) lying exactly on state B (4). The star is the average feature vector of the true future after a_+ and after a_-, which coincide.

E.1 Ablation of Density-FB's components

Figure 4 compares Density-FB and FB with three variants, on the same process and with the same training; their objectives are defined in Appendix E. Normalizing FB's target (*FB, normalized target*) leaves its optimum unchanged and, like FB, reads none of the difference up to six features. Applying a softmax to the bilinear scores and fitting it by squared error (*softmax + squared error*) recovers the full difference from four features, like Density-FB; the softmax parameterization, not the cross-entropy loss, is what keeps the difference with few features. Replacing the one-hot immediate target with a kernel over graph distance (*Density-FB, soft immediate target*) caps recovery at 71%, because its fixed point is a smoothed successor measure that spreads mass from state A onto neighbouring states.

Table 4: Density-FB hyperparameters for OGBench, MetaWorld, and ManiSkill, read from the configurations of the reported runs. ExoRL departs from these values as listed in the last block and in Appendix C.7.

	Symbol	Value
Successor model		
Latent dimension	d	50
Score temperature	τ	1.0
Tower hidden sizes	–	$(512, 512, 512)$
Hidden activation	–	GELU + LayerNorm
Forward heads	–	2
Backward output normalization	–	none
Orthogonality regularization	–	none
Immediate Bellman target	–	one-hot
Optimization		
Discount	γ	0.99
Batch size (candidate next states)	n	1024
Gradient steps	–	10^6
Optimizer	–	Adam
Learning rate (successor networks)	–	3×10^{-4}
Target EMA rate	η	0.005
Policy extraction		
Main extractor	–	flow-based actor
Actor MLP	–	2×512
Velocity-field MLP	–	4×512
Actor learning rate	–	10^{-4}
Velocity-field learning rate	–	3×10^{-4}
Euler steps	K	10
Distillation weight	λ_{BC}	3.0
Zero-shot evaluation		
Default value pool	P	4096
Pool sweep	P	$256, 1024, 4096, 16384$
Episodes per task and evaluation	–	20 (OGBench); 50 (MetaWorld, ManiSkill); 10 (ExoRL)
Reported score	–	mean over the last three evaluations
Goal-conditioned model (OGBench cells)		
Conditioning input	–	goal state g
Actor goals	–	future state of the same trajectory
Bootstrap goals (current/future/random)	–	0.2/0.5/0.3
Actor reward	–	Gaussian kernel on object coordinates
Kernel widths	ϵ_k	cube 0.4; other `scene` blocks 0.5–0.72
Evaluation goals	–	supplied by the environment
Reward-inference model (task vectors, Appendices C.1 and C.7)		
Random task-vector fraction	p_{rand}	0.5
Actor reward	–	centered latent reward r_z
Slow backward copy EMA rate	ξ	5×10^{-4}
Feature mean and covariance EMA rate	–	5×10^{-3}
Ridge coefficient	λ	$0.01\ \mathrm{tr}(\widehat{C})/d$
Eigenvalue floor	–	$10^{-6}\ \mathrm{tr}(\widehat{C})/d$
Reward-inference states	N_r	10^4
ExoRL (reward inference, Appendix C.7)		
Policy extractor	–	DDPG+BC, $\alpha = 0$ (no BC term)
Actor MLP	–	1×1024
Target-policy noise (std., clip)	–	0.2, 0.3
Backward MLP	–	$(256, 256)$
Backward output normalization	–	onto the sphere of radius $\sqrt{d}$
Learning rate (successor, actor)	–	10^{-4}, 10^{-4} (`point-mass`: 10^{-6})
Discount	γ	0.98 (`point-mass` 0.99)
Reward-inference states	–	5×10^4

Table 5: Dataset sizes and behavior policies. N trajectories of T wrapper steps each; at-goal fraction = fraction of transitions with the object held at the goal; episode success = fraction of collected trajectories that reach the goal.

Suite	Task	N	T	At-goal fraction	Episode success
MetaWorld	`push`	2010	500	0.921	—
MetaWorld	`bin-picking`	2010	500	0.895	—
MetaWorld	`box-close`	2010	500	0.812	—
ManiSkill	`push-cube`	8000	125	0.949	0.997
ManiSkill	`pick-cube`	8000	125	0.906	0.990
ManiSkill	`stack-cube`	8000	125	0.756	0.940
ManiSkill	`poke-cube`	8000	125	0.858	0.948
ManiSkill	`lift-peg`	8000	125	0.783	0.921

Table 6: **One transition graph (Figure 2), six continuation policies.** Probability of the first action (solid arrows in Figure 2) at each state; the second action has the complementary probability. P9 chooses each action with equal probability; R1–R5 are the sampled policies.

Setting	0	1	2	3	4	5	6	7	8
P9	0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.50
R1	0.46	0.35	0.49	0.73	0.47	0.42	0.75	0.64	0.36
R2	0.69	0.73	0.58	0.33	0.60	0.33	0.41	0.34	0.70
R3	0.40	0.69	0.58	0.75	0.74	0.32	0.35	0.37	0.38
R4	0.57	0.40	0.71	0.58	0.53	0.28	0.54	0.52	0.34
R5	0.61	0.49	0.65	0.70	0.26	0.55	0.50	0.59	0.25

Table 7: **Features needed to retain the action-value difference.** Smallest tested feature dimension for which the TD-trained model reaches $\widehat{G}_r/G_r \geq 0.9$. P9 uses the uniform policy; R1–R5 use sampled policies. A dash indicates that the threshold was not reached by $d = 8$.

Method	P9	R1	R2	R3	R4	R5
FB	8	8	8	8	–	–
FB, normalized target	7	7	8	8	–	–
Softmax + squared error	4	3	4	3	3	3
Density-FB	4	3	3	3	3	3
Density-FB, soft immediate target	–	–	–	–	–	–