BEACON: Cross-Domain Co-Training of Generative Robot Policies via Best-Effort Adaptation

Antong Zhang[1] **Han Qi**[2] **Heng Yang**[2]
[1]Department of Computer Science, Brown University
[2]School of Engineering and Applied Sciences, Harvard University
antong_zhang@brown.edu, hqi@g.harvard.edu
hankyang@seas.harvard.edu
https://computationalrobotics.seas.harvard.edu/BEACON/

Abstract

We introduce BEACON—Best-Effort Adaptation for Cross-Domain Co-Training—a theory-guided framework for training generative robot policies with abundant source demonstrations and limited target demonstrations. BEACON casts cross-domain co-training as a discrepancy-aware importance-reweighting problem, jointly learning a diffusion-based visuomotor policy and per-sample source weights through an objective inspired by the Best-Effort Adaptation (BEA) bound. To make this approach practical for high-dimensional sequence policies, we develop scalable instance-level discrepancy proxies, stochastic alternating updates for policy and weights, and a multi-source extension that balances heterogeneous source domains. Across sim-to-sim, sim-to-real, and multi-source manipulation settings, BEACON improves robustness and data efficiency over target-only, fixed-ratio co-training, and feature-alignment baselines. Without an explicit alignment objective, BEACON also brings source and target representations closer together as an implicit result of discrepancy-aware co-training.

1 Introduction

Behavior cloning (BC) has become a prominent route to learning generalizable robot manipulation policies because it reduces policy training to supervised imitation from expert demonstrations [1–4]. Recent sequence models, especially diffusion-based policies, further strengthen this paradigm by modeling temporally extended and multimodal action distributions [5, 6]. Despite this progress, policies trained in one domain often degrade in another even when task semantics are unchanged. Source and target domains can differ in appearance and camera viewpoint, while simulation-to-real transfer also involves differences in contact and dynamics [7–9]. Directly reusing source demonstrations can therefore be unreliable, while collecting large target-domain datasets is expensive and difficult to scale [10–12]. This raises a central question: *how can we leverage abundant but mismatched source demonstrations while using a minimal amount of target data?*

Classical methods use *domain randomization* to increase source diversity [8, 9, 13], while *domain adaptation* approaches explicitly align source and target representations to promote invariance to domain shifts [14–17]. More recent approaches show that *co-training* on source and target demonstrations can be effective when target data is scarce [18, 19]. However, these co-training approaches optimize empirical objectives through fixed mixing rules or auxiliary alignment losses. We focus instead on which source demonstrations should supervise the target policy. The question

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

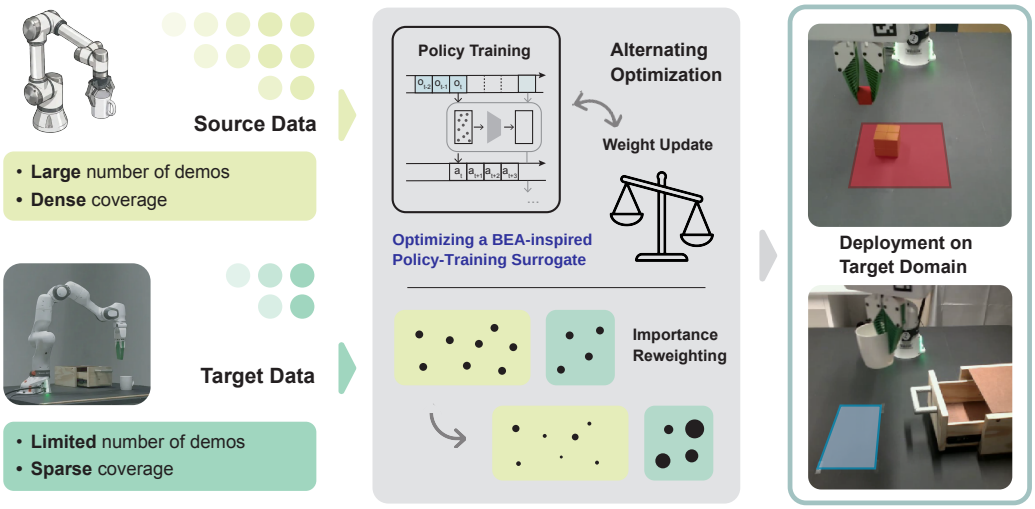

Figure 1: The BEACON framework. Given a large amount of (multi-)source demonstrations and a small amount of target demonstrations, BEACON jointly learns a diffusion-based visuomotor policy and a set of *per-sample weights* using a *discrepancy-aware objective inspired by BEA*.

then follows. *How can target-domain generalization bounds guide the design of training objectives that adaptively weight source demonstrations?*

Contribution. We take a theory-guided perspective and revisit the classical literature in domain adaptation, where target-domain generalization bounds guide the design of learning objectives. In particular, we build on the recent Best-Effort Adaptation (BEA) framework [20], and adapt its source-weighting principle to cross-domain diffusion-policy training for robot learning.

Concretely, we cast cross-domain co-training as an *importance-reweighting* problem with abundant source data and limited target data. The core idea is to reweight source samples using an instance-level discrepancy proxy to favor those more relevant to the target. This yields a *joint optimization* over *policy parameters* and *source weights*, guided by BEA's empirical-loss, discrepancy, and regularization terms. Applying this BEA-inspired formulation to modern manipulation tasks, however, requires practical adaptations for high-dimensional diffusion policies. We develop scalable discrepancy proxies and stochastic alternating updates tailored to sequence prediction [15, 21–23]. We also extend the method to multi-source settings motivated by realistic data pipelines, where demonstrations may come from different simulators, real-world scene configurations, or prior datasets, combining domain-level weights with weights on individual training windows. We call the method BEACON (Best-Effort Adaptation for Cross-Domain Co-Training, Figure 1) and evaluate it across sim-to-sim, sim-to-real, and multi-source manipulation, where it improves transfer over fixed-mixture co-training and feature-alignment baselines. Our analysis further shows that BEACON brings source and target representations closer together without an explicit feature-alignment loss.

Outline. The rest of the paper is organized as follows. Section 2 reviews related work. Section 3 introduces the theoretical foundations of BEA. Section 4 describes our BEACON method. Section 5 presents the experiments and analysis. Section 6 concludes the paper.

2 Related work

Behavior cloning for robotic manipulation. Behavior cloning reduces policy learning to supervised prediction from demonstration data [1, 2, 24]. Recent progress has come from both scaling real-robot datasets and improving sequence models for visuomotor control. Large multi-task and multi-domain robot datasets such as BridgeData and RT-1 show that diversity and scale materially improve generalization [10, 11]. Simulation benchmarks and datasets such as LIBERO and RoboCasa further expand the space of procedurally generated tasks and environments [25, 26]. In parallel, ACT predicts action chunks for precision manipulation, while Diffusion Policy models high-dimensional, multimodal action distributions [5, 6]. Recent analyses of transformer-based vision–language–action models report that action-aligned latent geometry emerges during training and tracks task performance [27]. At the same time, these successes accentuate the data bottleneck: collecting high-quality real demonstrations remains slow and expensive [18]. Systems such as MimicGen

partially alleviate this problem by synthesizing large simulation datasets from a small set of human demonstrations [12]. We build on this foundation of diffusion-based BC, focusing specifically on how to effectively leverage large-scale cross-domain source demonstrations (e.g., from simulation) alongside a limited dataset of target domain examples.

Sim-to-Real adaptation and Sim-and-Real co-training. Existing work addresses sim-to-real transfer primarily by reducing mismatch between simulated and real environments. Classical approaches such as domain randomization improve robustness by exposing policies to broader variation in simulation [8, 9, 13], while domain adaptation methods explicitly align the two domains at the pixel or feature level [14–17, 28]. Recent sim-and-real co-training methods further show that jointly training a single policy on mixed simulated and real demonstrations can be highly effective [18, 29]. Cheng et al. [19] augment co-training with task-aware Unbalanced Optimal Transport (UOT), using proprioceptive guidance and temporally aligned sampling to learn transferable representations. Its relaxed mass constraints accommodate unmatched portions of the two distributions. Although alignment-based co-training methods such as UOT show improvements, aligning feature distributions does not by itself ensure that the aligned features remain relevant to action under physical domain shifts. Their effectiveness also depends on which representations are aligned and how similarity is measured.

Our approach addresses the complementary question of which source examples should supervise the target policy. We learn their contributions to the behavior-cloning loss through importance reweighting, using policy features to estimate source–target mismatch without adding a feature-matching loss. This weighting formulation also extends naturally to multiple source domains.

Data weighting and curation for imitation learning. Recent methods select useful robot data at different levels. ReMix learns domain-level mixtures for robust imitation pretraining through distributionally robust optimization [30]. CUPID uses evaluation rollouts and influence estimates to rank demonstrations by their contribution to a policy's expected return [31]. BEACON instead learns target-specific weights on source observation windows jointly with the policy, using the available source and target demonstrations. These approaches provide complementary ways to decide how training data should influence policy learning.

Sample reweighting and multi-source domain adaptation. Classical work on distribution shift and domain adaptation characterized domain mismatch and motivated learning algorithms that account for differences between source and target distributions [32, 33]. Subsequent work developed task-aware discrepancy notions that depend on the loss and hypothesis class, yielding bounds and algorithms for regression, sample-bias correction, and generalized discrepancy minimization [21, 34, 35]. Classical importance-weighting approaches estimate weights from distribution mismatch before fitting the predictor [36, 37], with learning bounds characterizing the effect of those weights [38]. Multi-source domain adaptation studies the setting where labeled data are available from multiple source distributions [39, 40]. Prior work shows that naively combining sources can underperform the best individual source because domain shift exists not only between each source and the target, but also among sources [40–42]. Existing approaches use methods like source-specific alignment, domain aggregation, and learned source weighting to control negative transfer [41–43].

BEACON follows this discrepancy-based source-weighting direction, particularly the recent Best-Effort Adaptation framework [20], which explicitly assumes limited labeled target-domain data. Our key contribution is to adapt BEA to modern diffusion-based robot policy learning with practical enhancements for high-dimensional sequence prediction.

3 Problem setting and theoretical foundations

We study supervised learning over an input–output space $\mathcal{X} \times \mathcal{Y}$ with a source distribution Q and a target distribution P. We observe a labeled source dataset $D_{\mathrm{src}} = \{(x_i, y_i)\}_{i=1}^{m} \sim Q^m$ and a labeled target dataset $D_{\mathrm{tgt}} = \{(x_j, y_j)\}_{j=m+1}^{m+n} \sim P^n$, with typically $m \gg n$. Let $S = \{1, \ldots, m\}$ and $T = \{m+1, \ldots, m+n\}$ denote the source and target index sets. The goal is to learn a hypothesis $h \in \mathcal{H}$ that minimizes the target population loss

$$L(P, h) = \mathbb{E}_{(x,y)\sim P}\big[\ell(h(x), y)\big], \tag{1}$$

for a bounded loss $\ell : \mathcal{Y} \times \mathcal{Y} \to [0, 1]$. BEA [20] asks how to use abundant but potentially mismatched source data to improve performance on P, rather than naively mixing all samples equally.

Source reweighting. The key idea is to assign a nonnegative weight to each training example and learn both the predictor and the weights jointly. Concretely, we introduce a weight vector $\mathbf{q} = (q_1, \ldots, q_{m+n}) \in \mathbb{R}_+^{m+n}$, where each $q_i \geq 0$ controls the contribution of sample (x_i, y_i). Intuitively, source examples that are more relevant to the target domain should receive larger weight, while mismatched source examples should be down-weighted.

Discrepancy measure. To formalize when source data is helpful, BEA uses a discrepancy measure that quantifies domain mismatch in a way that is aligned with both the loss and the hypothesis class. For two labeled distributions P and Q, the one-sided labeled discrepancy is

$$\mathrm{dis}(P, Q) := \sup_{h \in \mathcal{H}} \left(\mathbb{E}_{(x,y)\sim P}[\ell(h(x), y)] - \mathbb{E}_{(x,y)\sim Q}[\ell(h(x), y)] \right). \tag{2}$$

Unlike a generic divergence between distributions, this quantity only measures differences that matter for predictors in $\mathcal{H}$ under the loss ℓ. At the sample level, BEA also introduces a discrepancy between the learned weighting $\mathbf{q}$ and a reference weighting $\mathbf{p}_0$. A natural choice for $\mathbf{p}_0$ is the target-only empirical distribution, $(\mathbf{p}_0)_i = \frac{1}{n}$ for $i \in T$ and $(\mathbf{p}_0)_i = 0$ for $i \in S$. This choice reflects the idealized baseline in which, absent any trustworthy source information, one would train only on the target sample. Given this reference, the weighted sample discrepancy is

$$\mathrm{dis}(\mathbf{q}, \mathbf{p}_0) := \sup_{h \in \mathcal{H}} \sum_{i=1}^{m+n} \left(q_i - (\mathbf{p}_0)_i \right) \ell(h(x_i), y_i). \tag{3}$$

This one-sided quantity measures an excess of the $\mathbf{q}$-weighted empirical loss over the target empirical loss, uniformly over $\mathcal{H}$.

Generalization bound. These quantities enter a discrepancy-based generalization bound due to [20, Corollary 4], which makes precise the trade-off between fitting the weighted training data and keeping the effective source contribution close to the target domain.

Theorem 1 (Discrepancy-based generalization bound [20])**.** *For any $\delta > 0$, with probability at least $1 - \delta$ over the choice of a source sample $D_{\mathrm{src}} = \{(x_i, y_i)\}_{i=1}^m \sim Q^m$ and a target sample $D_{\mathrm{tgt}} = \{(x_j, y_j)\}_{j=m+1}^{m+n} \sim P^n$, the following holds simultaneously for all $h \in \mathcal{H}$ and all weight vectors $\mathbf{q} \in [0,1]^{m+n}$ satisfying $\|\mathbf{q} - \mathbf{p}_0\|_1 < 1$:*

$$\begin{aligned} L(P, h) \leq & \sum_{i=1}^{m+n} q_i \, \ell(h(x_i), y_i) + q_{\mathrm{src}} \, \mathrm{dis}(P, Q) + \mathrm{dis}(\mathbf{q}, \mathbf{p}_0) + 2\mathfrak{R}_{\mathbf{q}}(\ell \circ \mathcal{H}) + 6\|\mathbf{q} - \mathbf{p}_0\|_1 \\ & + \left(\|\mathbf{q}\|_2 + 2\|\mathbf{q} - \mathbf{p}_0\|_1 \right) \left[\sqrt{\log \log_2 \frac{2}{1 - \|\mathbf{q} - \mathbf{p}_0\|_1}} + \sqrt{\frac{\log(2/\delta)}{2}} \right], \end{aligned} \tag{4}$$

where $q_{\mathrm{src}} := \sum_{i \in S} q_i$, $\mathrm{dis}(P, Q)$ is the labeled discrepancy (2) *between the source and target distributions, $\mathrm{dis}(\mathbf{q}, \mathbf{p}_0)$ is the weighted sample discrepancy* (3) *between $\mathbf{q}$ and the reference weights $\mathbf{p}_0$, and $\mathfrak{R}_{\mathbf{q}}(\ell \circ \mathcal{H})$ is the $\mathbf{q}$-weighted Rademacher complexity.*

The first term in Theorem 1 is the weighted empirical loss, so it encourages selecting examples that improve prediction. The second term shows that source data is only useful to the extent that it remains compatible with the target domain: large source mass is penalized when $\mathrm{dis}(P, Q)$ is large. The third term penalizes choices of $\mathbf{q}$ whose induced empirical objective deviates too far from the target-only reference $\mathbf{p}_0$. Finally, the remaining terms control complexity and suppress degenerate solutions that place too much mass on only a few examples.

The SBEST objective in BEA. Assuming the complexity term $\mathfrak{R}_{\mathbf{q}}(\ell \circ \mathcal{H})$ can be upper bounded by a norm penalty on h, Awasthi et al. [20] derive the SBEST learning objective from the BEA bound. Defining $d_i := \mathrm{dis}(P, Q)\, \mathbf{1}[i \in S]$ folds the source-discrepancy term into the weighted loss. Since $\ell \in [0, 1]$, the weighted sample discrepancy is upper bounded by $\|\mathbf{q} - \mathbf{p}_0\|_1$.[1] Absorbing the bound constants and concentration terms into regularization coefficients yields SBEST:

$$\min_{h \in \mathcal{H},\, \mathbf{q} \in [0,1]^{m+n}} \sum_{i=1}^{m+n} q_i \big(\ell(h(x_i), y_i) + d_i \big) + \lambda_\infty \|\mathbf{q}\|_\infty \|h\|^2 + \lambda_1 \|\mathbf{q} - \mathbf{p}_0\|_1 + \lambda_2 \|\mathbf{q}\|_2^2, \tag{5}$$

[1] BEA [20] also presents a variant that retains the weighted sample discrepancy term explicitly, yielding a more complex optimization problem.

where $\lambda_\infty, \lambda_1, \lambda_2 \geq 0$ are tunable coefficients. In BEA's setting, this bound-derived objective jointly learns a predictor and sample weights that fit the weighted data, limit the influence of discrepant source data, and remain anchored to the target empirical distribution. Section 4 describes our adaptation of this objective to diffusion-policy training.

4 Best-Effort Adaptation for cross-domain policy co-training

We now instantiate the BEA framework for learning visuomotor robot manipulation policies. We consider behavior cloning from demonstration trajectories collected in a source domain and a target domain. At each decision time t, the policy receives a window x_t of recent visual observations and proprioceptive features, and the desired output is an action chunk y_t. Each such window forms one training example (x_i, y_i); we use m, n for source and target window counts, and each window has its own weight. We learn a policy π_θ with the architecture of a Diffusion Policy [6], trained by behavior cloning. Accordingly, the hypothesis $h \in \mathcal{H}$ is replaced by π_θ, and the generic loss $\ell(h(x_i), y_i)$ is instantiated as the per-example diffusion behavior cloning loss $\ell_{\mathrm{BC}}(\theta; x_i, y_i)$. We adapt the BEA objective (5) to policy learning as follows:

$$\min_{\theta,\, \mathbf{q}\in\mathcal{C}_b} \sum_{i=1}^{m+n} q_i\big(\ell_{\mathrm{BC}}(\theta; x_i, y_i) + \lambda_d d_i\big) + \gamma\|\mathbf{q}\|_\infty R(\theta) + \lambda_1\|\mathbf{q}-\mathbf{r}\|_1 + \lambda_2\|\mathbf{q}\|_2^2, \tag{6}$$

where $R(\theta)$ is a policy regularizer and $\mathbf{r}$ is a uniform target-only reference, with r_i equal to one over the number of target demonstrations for target windows and 0 for source windows. The feasible set $\mathcal{C}_b = \{\mathbf{q} : \underline{q}_i \leq q_i \leq q_{\max},\ \sum_i q_i = b\}$ uses a target floor $\underline{q}_i = q_T^{\min}$ for $i \in T$, zero source floor, and total budget $b = n + \alpha m$. Target weights remain adaptive, so α controls the total budget rather than fixing source mass. The default discrepancy proxy d_i is nonnegative; we set $d_i = 0$ for target windows and tune $\lambda_d \geq 0$ to control the source penalty.

Challenges in applying BEA to visuomotor policy learning. It is nontrivial to carry the original BEA framework over to visuomotor policy learning. First, the original BEA analysis is developed for bounded supervised losses and hypothesis classes whose complexity can be controlled by relatively simple norm-based quantities. Diffusion Policies, however, are high-capacity sequence models trained with denoising-style objectives, and their optimization and generalization behavior are not well captured by the settings in which BEA was developed and evaluated. Second, the original BEA framework solves (5) through full-batch alternating minimization over h and $\mathbf{q}$. In image-conditioned robot policy learning, however, even a single loss evaluation requires a forward pass through a large diffusion model on high-dimensional observation histories. At the scales relevant to our setting, both policy updates and weight updates must therefore be carried out stochastically over minibatches rather than through full-batch optimization. Third, the discrepancy term d_i in BEA relies on optimization over relatively simple hypothesis classes with full access to labeled data. In our setting, the inputs are RGB observation windows, the outputs are action chunks, and the predictor is a deep generative policy. Solving the original discrepancy estimation problem directly in this space would be computationally prohibitive.

These mismatches motivate a practical adaptation of BEA for high-dimensional robot policy learning that preserves the structure of the bound-informed objective while replacing the parts of the original framework that do not scale to diffusion-based visuomotor control. Relative to (5), BEACON replaces the bounded loss with the diffusion behavior-cloning loss, the domain-level discrepancy $\mathrm{dis}(P, Q)$ with an instance-level proxy d_i, the norm-based capacity term with weight decay scaled by $\|\mathbf{q}\|_\infty$, BEA's admissible weights with the feasible set $\mathcal{C}_b$, and full-batch alternating minimization with stochastic alternating updates.

4.1 The BEACON algorithm

Complexity regularization via optimizer-compatible weight decay. In the original BEA objective (5), the capacity control term has the form $\lambda_\infty\|\mathbf{q}\|_\infty\|h\|^2$. For diffusion policies, we replace this by $\gamma\|\mathbf{q}\|_\infty R(\theta)$ with $R(\theta) := \frac{1}{2}\|\theta\|_2^2$, where γ sets the strength of weight decay. In the policy update, we scale the AdamW decay by $\|\mathbf{q}\|_\infty$, preserving the intuition that larger maximal sample weights should incur stronger regularization.

Instance-level discrepancy instead of domain-level discrepancy. A direct instantiation of BEA would assign the same discrepancy penalty $d_i = \mathrm{dis}(P, Q)\mathbf{1}[i \in S]$ to every source sample. In robot

manipulation, however, source demonstrations are highly heterogeneous: even within one simulator, some trajectories are visually and behaviorally close to the target domain, while others are clear outliers. We therefore replace the domain-level penalty by an instance-level discrepancy score d_i.

Our default proxy computes d_i in the policy's observation-encoder embedding space using k-nearest neighbors. Specifically, for each source sample x_i, we measure the average Euclidean distance from its embedding to its k nearest target embeddings, normalized by the mean within-target k-NN distance. Since the encoder is trained by behavior cloning on action chunks, d_i is not a pixel distance. The discrepancy scores are refreshed before each weight-update phase and held fixed during the subsequent updates.

We also consider two alternatives as ablations. The first is a localized labeled discrepancy estimator closer in spirit to BEA, which maximizes empirical discrepancy over a local policy class around a target-trained reference model. The second is a domain-classifier-based estimator that fits a binary discriminator on the encoder features and uses its per-source posterior as an instance-level mismatch signal. Implementation details are provided in the appendix. These variants help understand the role of discrepancy estimation, while the default k-NN estimator provides window-level scores without a separate reference policy or inner optimization.

Stochastic alternating optimization over policy and weights. We optimize (6) by alternating stochastic updates of θ and $\mathbf{q}$. For a minibatch $B \subseteq \{1, \dots, m+n\}$, we first compute the per-example loss $\ell_i(\theta) := \ell_{\mathrm{BC}}(\theta; x_i, y_i)$ and then update the weights through projected subgradient:

$$g_i = \ell_i(\theta) + \lambda_d d_i + \gamma R(\theta)\, v_i + \lambda_1 \operatorname{sign}(q_i - r_i) + 2\lambda_2 q_i, \qquad i \in B, \tag{7}$$

where v_i splits the capacity subgradient equally among minibatch entries attaining $\|\mathbf{q}\|_\infty$ and is zero otherwise. We update $q_i \leftarrow \mathrm{clip}_{[\underline{q}_i, q_{\max}]}(q_i - \eta_q g_i)$ and project onto $\mathcal{C}_b$ after the full weight sweep. The policy is then updated with AdamW on the weighted minibatch loss $\frac{1}{|B|}\sum_{i\in B} q_i \ell_i(\theta)$. This stochastic alternation is the practical analogue of BEA's joint optimization.

As an ablation, we also implement a convex inner update, closer to BEA's alternating minimization, that solves the weight subproblem at fixed θ and auxiliary variable $\|\mathbf{q}\|_\infty$. We found that the projected-subgradient update is simpler and better suited to large-scale training. More details are provided in the appendix.

Extension to multi-source co-training. In many realistic pipelines, the source data does not come from a single homogeneous domain, but from multiple simulators, scene configurations, or prior datasets. Pooling them into one source distribution can amplify negative transfer because mismatch exists both between each source and the target and among the source domains themselves.

To address this, we factor the source weight into two levels: a domain-level weight $\mathbf{w} \in \Delta^{K-1}$ over K source domains with index sets S_s, and a within-domain nonnegative weight $\widetilde{\mathbf{q}}$. If sample $i \in S$ comes from source domain $s(i)$, we define $q_i = w_{s(i)}\widetilde{q}_i, i \in S$, and $q_j = \widetilde{q}_j, j \in T$. This yields a multi-source BEACON objective in which $\mathbf{w}$ controls which source domains are useful globally, while $\widetilde{\mathbf{q}}$ suppresses mismatched training windows within each source. The multi-source objective evaluates (6) on these composite weights and adds $\rho_1\|\mathbf{w} - \mathbf{w}_0\|_1 + \rho_2\|\mathbf{w}\|_2^2$, with $\mathbf{w}_0 = \mathbf{1}/K$. Training alternates updates of the domain weights, within-domain weights, and policy. Full method details are provided in the appendix.

5 Experiments

The experiments are designed to substantiate five claims about BEACON. **(i)** Learned source weighting improves cross-domain policy performance relative to fixed-mixture co-training and feature-alignment baselines in the evaluated sim-to-sim and sim-to-real settings. **(ii)** BEACON learns policies that generalize to out-of-distribution task instances. **(iii)** Multi-source BEACON effectively combines heterogeneous source domains. **(iv)** Closer feature alignment does not necessarily yield better transfer. **(v)** BEACON shows an upward trend as the number of target demonstrations increases.

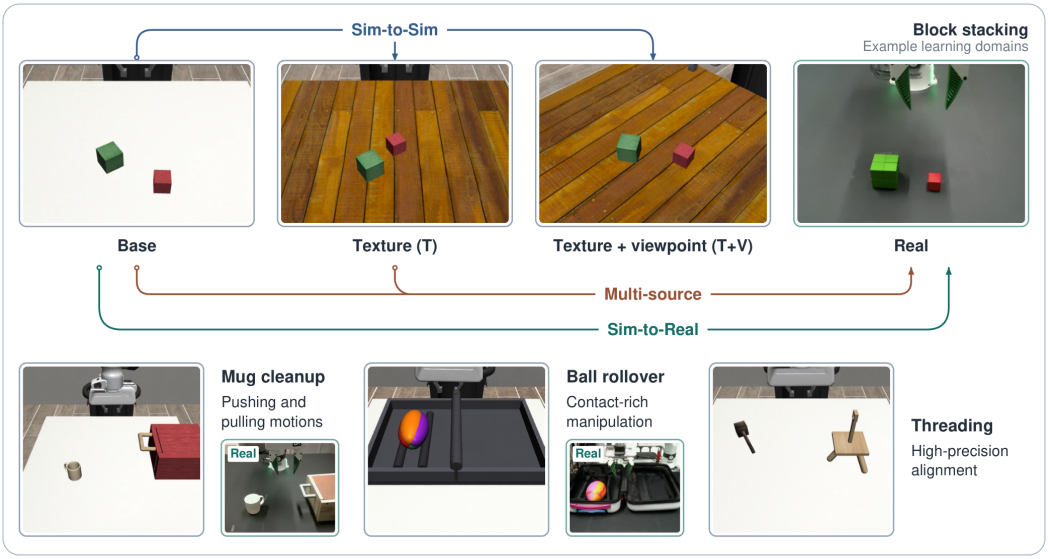

Figure 2: Experimental setup. Top shows block-stacking domains and transfer settings. Target-T changes the texture, and Target-T+V changes the texture and camera viewpoint relative to the base simulation. The real domain has both visual and physical gaps. Bottom shows demonstrations for other tasks that involve different motion patterns.

5.1 Experimental setup

We study image-based visuomotor manipulation on a Franka Emika Panda robot arm across four tasks. Our sim-to-sim benchmarks are *block stacking*, *mug cleanup*, and *threading*. Sim-to-real experiments use stacking, mug cleanup, and the contact-rich task of *ball rollover*, in which the robot rolls a football to a suitcase's central ridge and levers it into the opposite compartment (Figure 2). Tasks are evaluated in two cross-domain regimes: a controlled *sim-to-sim* setting, in which the source and target are both simulated but differ in their visual observation distributions, and a *sim-to-real* setting, in which the target is collected on the physical robot with both visual and physical gaps.

Visual gaps. The policy receives RGB observations resized to 96×96 together with end-effector proprioception, and the visual stream is encoded by a ResNet-18 backbone shared across domains. In simulation we instantiate two visual gaps between source and target: a *texture* gap, in which scene and object textures differ while the camera pose is held fixed, and a *texture + viewpoint* gap, which additionally perturbs the camera extrinsics. The sim-to-real gap is the natural mismatch between rendered simulation imagery and real RGB frames captured by a fixed camera observing the physical workspace, which subsumes background and texture differences.

Learning domains. All simulated demonstrations are generated in robosuite [44] via MimicGen [12], which expands a small set of human seed demonstrations into a large source dataset. For every task we use a source set of 500 demonstrations. The target set has 20 demonstrations in the sim-to-sim and 40 in the sim-to-real tasks. The environment with *texture gap* is used as the second source domain in the multi-source setting, where each source constitutes 50% of the total source data.

Tasks. For *block stacking*, both cubes are independently randomized over a square region of the workspace. For *mug cleanup*, the mug's position and orientation are randomized over a rectangular region. For *threading*, the thread's position and orientation are randomized over a rectangular region. For *ball rollover*, the ball starts at a randomized position on one side of the suitcase. To probe extrapolation, we additionally run an out-of-distribution (OOD) evaluation on *block stacking*, *mug cleanup*, and *ball rollover*: target demonstrations are collected with objects randomized over designated regions, while evaluation initial states are sampled exclusively from areas different from the target demonstrations. More details on task objectives and evaluation criteria are in the appendix.

Baselines. All methods use the same Diffusion Policy [6] backbone and observation encoder. *Source-Only* trains on $\mathcal{D}_{\text{src}}$ alone, and *Target-Only* trains on $\mathcal{D}_{\text{tgt}}$ alone. The remaining baselines use both datasets. *Co-Training* performs joint training on $\mathcal{D}_{\text{src}} \cup \mathcal{D}_{\text{tgt}}$ with a fixed source–target mixing

ratio [18]. *Tuned Uniform* assigns one common weight to all source losses, setting $q_i = c$ for every source window $i \in S$ instead of adapting q_i individually. We select this weight on block stacking and reuse it across all sim-to-sim tasks and gaps (Appendix D). *MMD* [16] augments co-training with a maximum mean discrepancy feature-alignment loss between source and target encoder activations. *UOT* [19] replaces the alignment loss with unbalanced optimal transport and temporally aligned sampling.

5.2 Generalization results

Table 1: Sim-to-sim mean task scores. T denotes the texture gap and T+V the texture + viewpoint gap. Stacking reports relaxed (R) and strict (S) success as defined in Appendix A; mug cleanup includes half credit for partial completion. Each Average is the equal-weight mean over stacking (R), mug cleanup, and threading.

	Block Stacking				**Mug Cleanup**		**Threading**		**Average**	
Method	T (R)	T (S)	T+V (R)	T+V (S)	T	T+V	T	T+V	T	T+V
Source-Only	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Target-Only	0.16	0.10	0.23	0.20	0.16	0.12	0.15	0.16	0.16	0.17
Co-Training	0.33	0.25	0.26	0.16	0.29	0.15	0.33	0.14	0.32	0.18
Tuned Uniform	0.62	0.45	0.47	0.37	0.34	0.17	0.38	0.18	0.45	0.27
MMD	0.87	0.72	0.67	0.54	0.47	**0.29**	0.47	0.28	0.60	0.41
UOT	0.68	0.48	0.52	0.38	0.46	0.23	0.41	0.22	0.52	0.32
BEACON	**0.89**	**0.77**	**0.73**	**0.60**	**0.51**	**0.29**	**0.53**	**0.33**	**0.64**	**0.45**

Table 2: Sim-to-real mean task scores. P denotes the real target domain, which includes visual and physical gaps. P-OOD uses initial states outside the target-demonstration regions. Each Average is the equal-weight mean over stacking (R), mug cleanup, and ball rollover; mug cleanup includes half credit.

	Block Stacking				**Mug Cleanup**		**Ball Rollover**		**Average**	
	P		P-OOD							
Method	R	S	R	S	P	P-OOD	P	P-OOD	P	P-OOD
Source-Only	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Target-Only	0.05	0.00	0.00	0.00	0.15	0.00	0.05	0.00	0.08	0.00
Co-Training	0.40	0.25	0.15	0.15	0.58	0.18	0.50	0.40	0.49	0.24
MMD	0.55	0.40	0.15	0.05	0.68	0.20	0.60	0.30	0.61	0.22
UOT	0.45	0.30	0.25	0.20	0.70	0.33	0.70	0.50	0.62	0.36
BEACON	0.75	**0.55**	**0.45**	0.35	0.78	0.53	**0.85**	**0.70**	0.79	0.56
MS-BEACON	**0.85**	**0.55**	**0.45**	**0.45**	**0.80**	**0.60**	**0.85**	**0.70**	**0.83**	**0.58**

BEACON improves sim-to-sim transfer. Table 1 shows that BEACON achieves the highest average task score of 0.64 in the *texture gap* and 0.45 in the *texture + viewpoint gap* over tasks that involve different motion patterns. Notably, in *block stacking*, BEACON shows more robust performance when switched to the stricter S criterion; the mean relative reduction from R to S across the two visual gaps is 15.6%, compared to 18.3% and 28.2% for *MMD* and *UOT*, respectively. This pattern is important because feature-alignment baselines already perform well; BEACON still improves over them by selecting source samples that better shape the target policy.

Adaptive source weights improve over a tuned uniform weight. Tuning a common source weight improves over fixed-mixture co-training, but BEACON remains stronger across the sim-to-sim tasks and gaps (Table 1). On stacking under T+V, strict success increases from 0.37 with Tuned Uniform to 0.60 with BEACON. A controlled ablation in Appendix C holds the k-NN estimator and stochastic update fixed and replaces each source discrepancy score by the mean source score. Strict success then decreases from 0.60 to 0.51 (Table 3). Together, these comparisons support the value of instance-level source weighting guided by local discrepancy information.

BEACON delivers the strongest sim-to-real transfer. Table 2 shows that *Target-Only* policies nearly fail, and hence the limited real demonstrations are insufficient by themselves. *MMD* and *UOT*

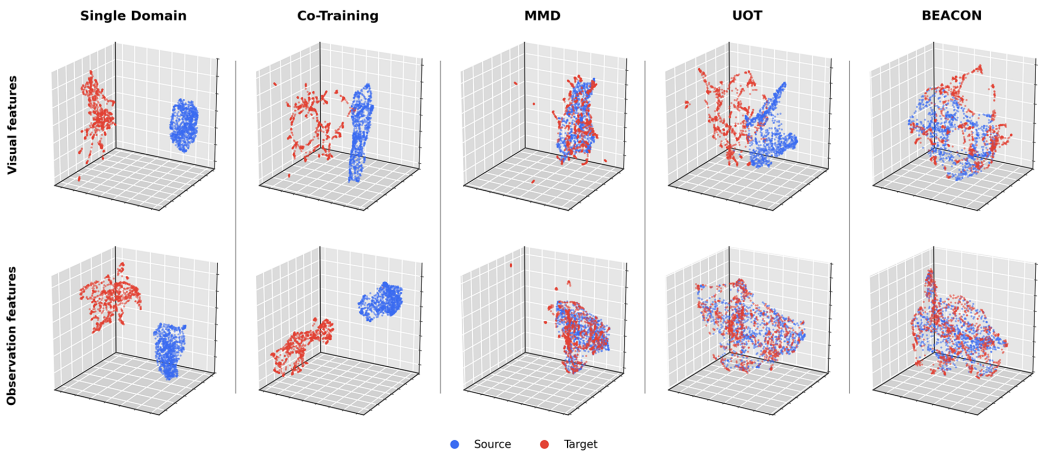

Figure 3: UMAP visualization of latent features (*block stacking*, sim-to-sim texture gap). We visualize latent features of the image alone as well as all observations (visual + proprioception) after the encoder trunk. Source and target features come closer together in BEACON as a byproduct of discrepancy-aware co-training.

improve over Target-Only, but their scores drop under OOD initial states. BEACON reaches the strongest single-source performance across all three tasks, with average scores of 0.79 on P and 0.56 on P-OOD. On contact-rich ball rollover, which requires sustained gripper–ball contact and coordinated rolling and levering over a ridge, BEACON reaches 0.85 success on P and 0.70 on P-OOD, compared with 0.70 and 0.50 for UOT. Hyperparameters selected on stacking are reused for this task. These results support adaptive source weighting for transfer across combined visual and physical mismatch.

BEACON generalizes beyond the target states seen during training. BEACON remains robust when evaluation states move outside the region covered by the target demonstrations. Under the strict real-world stacking criterion in Table 2, BEACON retains 0.35 OOD success from 0.55 in-distribution, whereas *MMD* drops from 0.40 to 0.05, the lowest OOD score among the co-training methods and below plain *Co-Training* (0.15). This contrast shows that an explicit feature-alignment objective alone does not ensure robustness to unseen task states.

MS-BEACON further strengthens transfer with heterogeneous sources. With the same total source-demonstration budget, MS-BEACON raises the three-task averages to 0.83 on P and 0.58 on P-OOD (Table 2). It improves relaxed stacking success and mug-cleanup scores while matching BEACON on ball rollover, indicating that heterogeneous simulation data can potentially make the learned policy both more accurate and more robust. The learned domain weights favor the source with lower measured discrepancy (Appendix B.3). These results support adapting how each domain and observation window contributes to policy learning.



Figure 4: Target data scaling. The target performance shows improvement with additional target demonstrations in the sim-to-sim *block stacking* setting. Starred curves show strict (S) success.

Closer feature alignment does not necessarily yield better transfer. Figure 3 visualizes the source and target features learned by each method. Without an explicit feature-alignment loss, BEACON brings source and target features closer together than plain co-training, reducing RBF-MMD from 0.534 to 0.168 on stacking and from 0.460 to 0.045 on mug cleanup (Appendix E, Table 4); this closeness arises implicitly through discrepancy-aware training. Yet the *MMD* baseline is the most aligned method on both tasks, and closer alignment does not consistently translate into higher task success. Together with the OOD results, this suggests that choosing which source

samples supervise the policy is more useful for transfer in our settings than minimizing feature mismatch alone.

BEACON shows improvement as the number of target demonstrations increases. We test the effect of scaling up the number of target demonstrations on BEACON's performance by using 20, 40, 60, and 80 target demonstrations in the sim-to-sim *block stacking* setting. Figure 4 shows that performance generally improves with additional target demonstrations.

6 Conclusion and limitations

We presented BEACON, a theory-guided framework that recasts cross-domain co-training of generative robot policies as a discrepancy-aware importance-reweighting problem, jointly learning a diffusion policy and per-sample source weights through an objective inspired by BEA. We introduced an instance-level discrepancy proxy computed in the policy's own embedding space, stochastic alternating optimization over policy parameters and weights, and a two-level extension that combines heterogeneous source domains. Across sim-to-sim, sim-to-real, and multi-source experiments, BEACON consistently outperforms target-only, fixed-ratio co-training, and feature-alignment baselines while bringing source and target representations closer together as an implicit result of discrepancy-aware co-training. Our results suggest that choosing which source samples supervise the policy can matter more for transfer than achieving closer feature alignment. The MMD baseline attained the lowest measured RBF-MMD on both analyzed tasks but did not transfer best, and suffered a particularly sharp drop under OOD initial states in real-world stacking. This supports adaptive source weighting as a useful principle for cross-domain policy co-training.

Although we chose tasks with different motion patterns and action dynamics, our task coverage is still limited. Also, our work builds on the assumption that real data is scarce, while in reality, there might be heterogeneous but considerably larger real data available (e.g. data collected with different real-world setups, lighting conditions), which can be used as a source domain in MS-BEACON. On the theoretical side, the link between our feature-distance proxy and BEA's loss- and hypothesis-dependent labeled discrepancy remains to be established.

Acknowledgments and Disclosure of Funding

We acknowledge funding from the Office of Naval Research grant N00014-25-1-2322.

References

[1] Dean A. Pomerleau. ALVINN: An autonomous land vehicle in a neural network. In *Advances in Neural Information Processing Systems*, volume 1, pages 305–313, 1989.

[2] Aude Billard, Sylvain Calinon, Rüdiger Dillmann, and Stefan Schaal. Robot programming by demonstration. In *Springer Handbook of Robotics*, pages 1371–1394. Springer, 2008.

[3] Han Qi, Changhe Chen, and Heng Yang. Compose by focus: Scene graph-based atomic skills. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2026.

[4] Han Qi, Haocheng Yin, Aris Zhu, Yilun Du, and Heng Yang. Inference-time enhancement of generative robot policies via predictive world modeling. *IEEE Robotics and Automation Letters*, 2026.

[5] Tony Z. Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning fine-grained bimanual manipulation with low-cost hardware. In *Robotics: Science and Systems*, 2023.

[6] Cheng Chi, Siyuan Feng, Yilun Du, Zhenjia Xu, Eric Cousineau, Benjamin Burchfiel, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. In *Robotics: Science and Systems*, 2023. doi: 10.15607/RSS.2023.XIX.026.

[7] Andrei A. Rusu, Matej Večerík, Thomas Rothörl, Nicolas Heess, Razvan Pascanu, and Raia Hadsell. Sim-to-real robot learning from pixels with progressive nets. In *Conference on Robot Learning*, pages 262–270. PMLR, 2017.

[8] Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 23–30, 2017.

[9] Xue Bin Peng, Marcin Andrychowicz, Wojciech Zaremba, and Pieter Abbeel. Sim-to-real transfer of robotic control with dynamics randomization. In *IEEE International Conference on Robotics and Automation*, pages 3803–3810, 2018.

[10] Frederik Ebert, Yanlai Yang, Karl Schmeckpeper, Bernadette Bucher, Georgios Georgakis, Kostas Daniilidis, Chelsea Finn, and Sergey Levine. Bridge data: Boosting generalization of robotic skills with cross-domain datasets. *arXiv preprint arXiv:2109.13396*, 2021.

[11] Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Jasmine Hsu, Julian Ibarz, Brian Ichter, Alex Irpan, Tomas Jackson, Sally Jesmonth, Nikhil J. Joshi, Ryan Julian, Dmitry Kalashnikov, Yuheng Kuang, Isabel Leal, Kuang-Huei Lee, Sergey Levine, Yao Lu, Utsav Malla, Deeksha Manjunath, Igor Mordatch, Ofir Nachum, Carolina Parada, Jodilyn Peralta, Emily Perez, Karl Pertsch, Jornell Quiambao, Kanishka Rao, Michael Ryoo, Grecia Salazar, Pannag Sanketi, Kevin Sayed, Jaspiar Singh, Sumedh Sontakke, Austin Stone, Clayton Tan, Huong Tran, Vincent Vanhoucke, Steve Vega, Quan Vuong, Fei Xia, Ted Xiao, Peng Xu, Sichun Xu, Tianhe Yu, and Brianna Zitkovich. RT-1: Robotics transformer for real-world control at scale. In *Robotics: Science and Systems*, 2023.

[12] Ajay Mandlekar, Soroush Nasiriany, Bowen Wen, Iretiayo Akinola, Yashraj Narang, Linxi Fan, Yuke Zhu, and Dieter Fox. MimicGen: A data generation system for scalable robot learning using human demonstrations. In *Conference on Robot Learning*, volume 229 of *Proceedings of Machine Learning Research*, pages 1820–1864. PMLR, 2023.

[13] Fereshteh Sadeghi and Sergey Levine. CAD2RL: Real single-image flight without a single real image. In *Robotics: Science and Systems*, 2017.

[14] Konstantinos Bousmalis, Nathan Silberman, David Dohan, Dumitru Erhan, and Dilip Krishnan. Unsupervised pixel-level domain adaptation with generative adversarial networks. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 3722–3731, 2017.

[15] Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. Domain-adversarial training of neural networks. *Journal of Machine Learning Research*, 17(59):1–35, 2016.

[16] Mingsheng Long, Yue Cao, Jianmin Wang, and Michael I. Jordan. Learning transferable features with deep adaptation networks. In *International Conference on Machine Learning*, pages 97–105. PMLR, 2015.

[17] Nicolas Courty, Rémi Flamary, Devis Tuia, and Alain Rakotomamonjy. Optimal transport for domain adaptation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(9): 1853–1865, 2017.

[18] Abhiram Maddukuri, Zhenyu Jiang, Lawrence Yunliang Chen, Soroush Nasiriany, Yuqi Xie, Yu Fang, Wenqi Huang, Zu Wang, Zhenjia Xu, Nikita Chernyadev, Scott Reed, Ken Goldberg, Ajay Mandlekar, Linxi Fan, and Yuke Zhu. Sim-and-real co-training: A simple recipe for vision-based robotic manipulation. *Robotics: Science and Systems*, 2025.

[19] Shuo Cheng, Liqian Ma, Zhenyang Chen, Ajay Mandlekar, Caelan Garrett, and Danfei Xu. Generalizable domain adaptation for sim-and-real policy co-training. In *Advances in Neural Information Processing Systems*, 2025.

[20] Pranjal Awasthi, Corinna Cortes, and Mehryar Mohri. Best-effort adaptation. *arXiv preprint arXiv:2305.05816*, 2023.

[21] Corinna Cortes, Mehryar Mohri, and Andrés Muñoz Medina. Adaptation based on generalized discrepancy. *Journal of Machine Learning Research*, 20(1):1–30, 2019.

[22] Yuchen Zhang, Mingsheng Long, Jianmin Wang, and Michael I. Jordan. On localized discrepancy for domain adaptation. *arXiv preprint arXiv:2008.06242*, 2020.

[23] Thomas M. Cover and Peter E. Hart. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, 13(1):21–27, 1967.

[24] Han Qi, Haocheng Yin, and Heng Yang. Control-oriented clustering of visual latent representation. In *International Conference on Learning Representations (ICLR)*, 2025.

[25] Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. LIBERO: Benchmarking knowledge transfer for lifelong robot learning. *arXiv preprint arXiv:2306.03310*, 2023.

[26] Soroush Nasiriany, Abhiram Maddukuri, Lance Zhang, Adeet Parikh, Aaron Lo, Abhishek Joshi, Ajay Mandlekar, and Yuke Zhu. RoboCasa: Large-scale simulation of household tasks for generalist robots. In *Robotics: Science and Systems*, 2024.

[27] Han Qi, Huangyuan Su, Liuyixin Shao, Na Li, and Heng Yang. Action-aligned representation geometry in Vision–Language–Action models. Research Square preprint, August 2026. URL `https://www.researchsquare.com/article/rs-10286824/v1`. Version 1, posted August 28, 2026.

[28] Judy Hoffman, Eric Tzeng, Taesung Park, Jun-Yan Zhu, Phillip Isola, Kate Saenko, Alexei A. Efros, and Trevor Darrell. CyCADA: Cycle-consistent adversarial domain adaptation. In *International Conference on Machine Learning*, pages 1989–1998. PMLR, 2018.

[29] Yu Lei, Minghuan Liu, Abhiram Maddukuri, Zhenyu Jiang, and Yuke Zhu. A mechanistic analysis of sim-and-real co-training in generative robot policies. *arXiv preprint arXiv:2604.13645*, 2026.

[30] Joey Hejna, Chethan Anand Bhateja, Yichen Jiang, Karl Pertsch, and Dorsa Sadigh. ReMix: Optimizing data mixtures for large scale imitation learning. In *Proceedings of the 8th Conference on Robot Learning*, volume 270 of *Proceedings of Machine Learning Research*, pages 145–164. PMLR, 2025. URL `https://proceedings.mlr.press/v270/hejna25a.html`.

[31] Christopher Agia, Rohan Sinha, Jingyun Yang, Rika Antonova, Marco Pavone, Haruki Nishimura, Masha Itkina, and Jeannette Bohg. CUPID: Curating data your robot loves with influence functions. In *Proceedings of the 9th Conference on Robot Learning*, volume 305 of *Proceedings of Machine Learning Research*, pages 2907–2932. PMLR, 2025. URL `https://proceedings.mlr.press/v305/agia25a.html`.

[32] Daniel Kifer, Shai Ben-David, and Johannes Gehrke. Detecting change in data streams. In *International Conference on Very Large Data Bases*, pages 180–191, 2004.

[33] Shai Ben-David, John Blitzer, Koby Crammer, Alex Kulesza, Fernando Pereira, and Jennifer Wortman Vaughan. A theory of learning from different domains. *Machine Learning*, 79 (1–2):151–175, 2010.

[34] Yishay Mansour, Mehryar Mohri, and Afshin Rostamizadeh. Domain adaptation: Learning bounds and algorithms. In *Proceedings of the 22nd Annual Conference on Learning Theory*. Omnipress, 2009. URL `https://arxiv.org/abs/0902.3430`.

[35] Corinna Cortes and Mehryar Mohri. Domain adaptation and sample bias correction theory and algorithm for regression. *Theoretical Computer Science*, 519:103–126, 2014.

[36] Jiayuan Huang, Alex J. Smola, Arthur Gretton, Karsten M. Borgwardt, and Bernhard Schölkopf. Correcting sample selection bias by unlabeled data. In *Advances in Neural Information Processing Systems*, volume 19, 2007.

[37] Masashi Sugiyama, Shinichi Nakajima, Hisashi Kashima, Paul von Bünau, and Motoaki Kawanabe. Direct importance estimation with model selection and its application to covariate shift adaptation. In *Advances in Neural Information Processing Systems*, volume 20, 2007.

[38] Corinna Cortes, Yishay Mansour, and Mehryar Mohri. Learning bounds for importance weighting. In *Advances in Neural Information Processing Systems*, volume 23, 2010.

[39] Yishay Mansour, Mehryar Mohri, and Afshin Rostamizadeh. Domain adaptation with multiple sources. In *Advances in Neural Information Processing Systems*, volume 21, 2009.

[40] Sicheng Zhao, Hui Chen, Hu Huang, Pengfei Xu, and Guiguang Ding. More is better: Deep domain adaptation with multiple sources. In *International Joint Conference on Artificial Intelligence*, pages 8354–8362, 2024.

[41] Han Zhao, Shanghang Zhang, Guanhang Wu, José M. F. Moura, João P. Costeira, and Geoffrey J. Gordon. Adversarial multiple source domain adaptation. In *Advances in Neural Information Processing Systems*, volume 31, 2018.

[42] Junfeng Wen, Russell Greiner, and Dale Schuurmans. Domain aggregation networks for multi-source domain adaptation. In *International Conference on Machine Learning*, pages 10214–10224. PMLR, 2020.

[43] Xingchao Peng, Qinxun Bai, Xide Xia, Zijun Huang, Kate Saenko, and Bo Wang. Moment matching for multi-source domain adaptation. In *IEEE/CVF International Conference on Computer Vision*, pages 1406–1415, 2019.

[44] Yuke Zhu, Josiah Wong, Ajay Mandlekar, Roberto Martín-Martín, Abhishek Joshi, Kevin Lin, Soroush Nasiriany, and Yifeng Zhu. robosuite: A modular simulation framework and benchmark for robot learning. *arXiv preprint arXiv:2009.12293*, 2020.

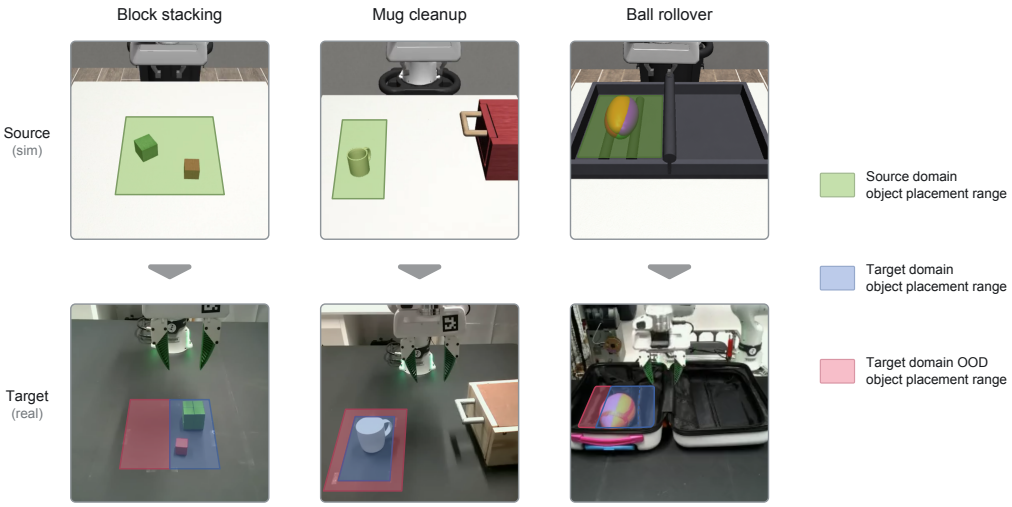

Figure A1: Object placement range for OOD evaluation. The green region shows the placement range for source demonstrations; the blue region shows the placement range for target demonstrations, and the red regions are the OOD evaluation regions. Each column shows one task, with simulation above real-world experiments. For ball rollover, placement regions are restricted to the left luggage compartment: simulation covers its full interior, while real demonstrations cover only the middle and right sections; the uncollected left section is held out as OOD and requires an extra left-to-middle motion before moving to the right section.

A Task setups

We conduct comprehensive sim-to-sim, sim-to-real, and multi-source experiments on tasks that involve different motion patterns.

Evaluation and scoring. Each simulated policy is evaluated on 100 rollouts and each real-robot policy on 20 rollouts. Initial states are sampled independently across rollouts and matched across methods for each task. Each simulated rollout uses a distinct evaluation seed, separate from the training seed. Mug cleanup awards 1 for complete success and 0.5 for partial completion; the reported values are therefore mean task scores. Stacking, threading, and ball rollover use binary success. The task objectives and success criteria are defined below.

Block stacking: grasp the smaller cube (A) and stack it on the larger one (B). This task involves simple pick-and-place motions. Both cubes' (x, y) positions are independently randomized over a square region; OOD evaluation uses a disjoint region shown in Figure A1. Relaxed (R) success requires cube A to be stacked on cube B at any frame; strict (S) success requires the stack to persist for at least five consecutive frames or occur at the final frame.

Mug cleanup: drag and open the drawer, pick up a mug and place it upright inside the drawer, and then close the drawer. This task combines pulling and pushing the drawer with picking and placing the mug. The mug's (x, y) positions are randomized over a rectangular region; OOD evaluation uses a spatially disjoint region. The task is considered successful when the mug is inside the drawer, upright, and the drawer is closed. From a full score of 1, a mug that is not upright and a drawer that is not fully closed each incur a 0.5-point deduction. A failed attempt scores zero.

Threading: grasp a thin rod/thread and insert it through a small receptacle. This task requires precise insertion with sub-centimeter alignment. The rod's (x, y) and yaw are randomized over a rectangular region. The task is considered successful when the rod/thread is inserted through the receptacle.

Ball rollover: a suitcase lies flat on the table, with a football at a randomized position on one side. The gripper rolls the ball to the central ridge, reaches its base, and levers it into the opposite compartment. This contact-rich task requires sustained gripper–ball contact through the rolling and levering sequence, with the ridge nearly as tall as the ball. Success requires moving the ball across

the ridge into the opposite compartment. The multi-source experiment uses the same two simulation domains and total source-demonstration budget as the other sim-to-real tasks.

B Algorithm and training details

This section describes how BEACON is implemented and trained in practice. We start with the single-source training loop (Section B.1), then detail the k-nearest-neighbor discrepancy estimator that supplies the per-sample d_i values (Section B.2), and finally describe the multi-source extension MS-BEACON (Section B.3). Implementation-level choices and default hyperparameters appear in Section B.4.

B.1 Single-source training loop

The objective in (6) is jointly nonconvex in $(\theta, \mathbf{q})$: it is convex in $\mathbf{q}$ for fixed θ but nonconvex in θ for fixed $\mathbf{q}$ because ℓ_{BC} is the diffusion behavior-cloning loss of a deep policy. We therefore optimize it by alternating a weight-refresh phase and a policy-training phase (Algorithm A1). On refresh epochs, the $\mathbf{q}$-phase fixes θ and takes one sweep of projected subgradient steps over $\mathbf{q}$, followed by a single global projection that enforces the box and sum constraints on the weights. In every epoch, the h-phase fixes $\mathbf{q}$ and takes a configurable number of minibatch steps on the $\mathbf{q}$-weighted policy loss. Decoupled weight decay is rescaled to $\gamma \|\mathbf{q}\|_\infty$ during the h-phase, which serves as an optimizer-level analogue of the capacity term $\gamma \|\mathbf{q}\|_\infty R(\theta)$ in (6), without an explicit gradient through $R(\theta)$.

Algorithm A1 BEACON: single-source training loop

Require: Combined dataset $\mathcal{D}$ with source/target indices S, T; policy π_θ; reference $\mathbf{r}$; coefficients $(\lambda_d, \lambda_1, \lambda_2, \gamma, \alpha)$; step sizes (η_q, η_θ); bounds $[\underline{q}_i, q_{\max}]$; q-refresh period K_q; h-steps per epoch S_h; epochs E.

Initialize θ and $q_i \leftarrow r_i$ for all $i \in S \cup T$.
for $e = 1, \ldots, E$ **do**
  if $e = 1$ **or** $e \bmod K_q = 0$ **then**
    Refresh $\{d_i\}_{i \in S}$ via Algorithm A2.
    for each minibatch $\mathcal{B} \subset \mathcal{D}$ in a sequential sweep **do**
      Evaluate $\{\ell_i(\theta)\}_{i \in \mathcal{B}}$ (no backprop).
      for $i \in \mathcal{B}$ **do**
        $g_i \leftarrow \ell_i + \lambda_d d_i + \gamma R(\theta)\, v_i + \lambda_1 \operatorname{sign}(q_i - r_i) + 2\lambda_2 q_i$ ▷ cf. (7)
        $q_i \leftarrow \mathrm{clip}_{[\underline{q}_i,\, q_{\max}]}\big(q_i - \eta_q g_i\big)$
      end for
    end for
    Project $\mathbf{q}$ onto $\{q_i \in [\underline{q}_i, q_{\max}],\ \sum_i q_i = n + \alpha m\}$ by bisection.
  end if

  for $s = 1, \ldots, S_h$ **do**
    Sample minibatch $\mathcal{B}$ from the source–target sampling mixture.
    $\theta \leftarrow \mathrm{AdamW}\big(\theta,\ \nabla_\theta \frac{1}{|\mathcal{B}|} \sum_{i \in \mathcal{B}} q_i\, \ell_i(\theta);\ \eta_\theta\big)$.
  end for
end for
return θ.

B.2 Per-sample k-NN discrepancy estimator

We instantiate the source–target discrepancy proxy d_i used in (6) as a per-sample k-nearest-neighbor distance in the embedding space of the policy's observation encoder ϕ_θ, with self-normalization by the within-target spread (Algorithm A2). The embedding concatenates visual and proprioceptive features over the observation window. The normalization makes the scale of d_i approximately invariant to a common rescaling of the embedding space. Because ϕ_θ co-evolves with the policy, we recompute $\{d_i\}_{i \in S}$ at the beginning of every $\mathbf{q}$-update phase (line 4 of Algorithm A1); when the target sample is small ($n < k$), the effective neighborhood size collapses to $k_{\mathrm{eff}} = \min(k, n)$.

Algorithm A2 Per-sample k-NN discrepancy in policy embedding space

Require: Inputs $\{x_i\}_{i\in S\cup T}$; encoder ϕ_θ; neighbors k (with $k_{\text{eff}}{=}\min(k,n)$).
$z_i \leftarrow \phi_\theta(x_i)$ for all $i \in S \cup T$ ▷ single forward pass through ϕ_θ
$Z \leftarrow$ mean k_{eff}-NN distance among target embeddings $\{z_j\}_{j\in T}$, excluding self-matches.
for $i \in S$ **do**
    $d_i \leftarrow \big($mean distance from z_i to its k_{eff} nearest target embeddings$\big) \,/\, Z$.
end for
return $\{d_i\}_{i\in S}$.

B.3 Multi-source training loop

For multi-source training we factor the per-sample weight as $q_i = w_{s(i)}\widetilde{q}_i$ for $i \in S$ and $q_j = \widetilde{q}_j$ for $j \in T$, where $\mathbf{w} \in \Delta^{K-1}$ is the simplex-constrained vector of source-domain weights and $\widetilde{\mathbf{q}}$ is the nonnegative within-source/target weight vector. The multi-source objective augments (6) with an additional L_1/L_2 regularizer on $\mathbf{w}$ around a reference $\mathbf{w}_0$:

$$\min_{\theta,\,(\mathbf{w},\widetilde{\mathbf{q}})\in\mathcal{M}_b} \; \mathcal{F}_{\text{single}}\big(\theta,\, \mathbf{q}(\mathbf{w},\widetilde{\mathbf{q}})\big) \;+\; \rho_1\|\mathbf{w}-\mathbf{w}_0\|_1 \;+\; \rho_2\|\mathbf{w}\|_2^2, \tag{8}$$

where $\mathcal{F}_{\text{single}}$ is the single-source BEACON objective (6) evaluated on the composite weights $\mathbf{q}(\mathbf{w},\widetilde{\mathbf{q}})$. The feasible set $\mathcal{M}_b$ requires $\mathbf{w} \in \Delta^{K-1}$, $0 \le \widetilde{q}_i \le q_{\max}$ for $i \in S$, $q_T^{\min} \le \widetilde{q}_j \le q_{\max}$ for $j \in T$, and $\sum_{i\in S} w_{s(i)}\widetilde{q}_i + \sum_{j\in T}\widetilde{q}_j = b$. This induces three blocks of variables (Algorithm A3). On refresh epochs, we perform a *w-step* that uses a single full-dataset gradient, scaled by $1/\bar{m}$ with $\bar{m} = m/K$, followed by clipping to $[0,1]$ and projection onto Δ^{K-1}; a *$\widetilde{q}$-step* that takes one minibatch sweep of projected subgradient on the within-source weights and a separate update on the target weights; and a final *joint projection* of the composite vector $\mathbf{q}$ onto the box-and-sum constraint set with source bounds $0 \le q_i \le w_{s(i)}q_{\max}$ and target bounds $q_T^{\min} \le q_j \le q_{\max}$, which is then split back into $\widetilde{\mathbf{q}}$ via the change of variables above. The h-step runs in every epoch and is identical to the single-source case but uses the composite weights $q_i = w_{s(i)}\widetilde{q}_i$ when forming the weighted minibatch loss. The h-step uses the same optimizer-level weight decay as Algorithm A1. The multi-source w- and $\widetilde{q}$-updates omit the capacity subgradient and use their respective regularizers (ρ_1,ρ_2) and (λ_1,λ_2). Because the two sources in our experiments contain nearly equal numbers of observation windows (within 0.31%), the common scale matches normalizing each domain's term by its own window count to within 0.31%, so the learned domain weights are not driven by source size.

Algorithm A3 MS-BEACON: multi-source training loop

Require: K source datasets $\{\mathcal{D}_{\text{src},k}\}$ with assignment $s: S \to \{1,\dots,K\}$; target $\mathcal{D}_{\text{tgt}}$; reference $\mathbf{w}_0 = (1/K)\mathbf{1}$; domain regularizers (ρ_1,ρ_2); step sizes (η_w,η_T); remaining hyperparameters as in Algorithm A1.
Initialize θ, $w_k \leftarrow 1/K$, and $\widetilde{\mathbf{q}}$ from $\mathbf{r}$; let $q_i = w_{s(i)}\widetilde{q}_i$ for $i \in S$ and $q_j = \widetilde{q}_j$ for $j \in T$.
for $e = 1,\dots,E$ **do**
    if $e = 1$ **or** $e \bmod K_q = 0$ **then**
        Refresh $\{d_i\}_{i\in S}$ via Algorithm A2; collect $\{\ell_i(\theta)\}_{i\in S\cup T}$ in one sequential sweep of $\mathcal{D}$.
        w-step: $\mathbf{w} \leftarrow \Pi_{\Delta^{K-1}}\big(\text{clip}_{[0,1]}(\mathbf{w} - (\eta_w/\bar{m})\,\nabla_{\mathbf{w}}\mathcal{F})\big)$, where $\nabla_{\mathbf{w}}\mathcal{F}$ is the closed-form gradient of (8) in $\mathbf{w}$ (with the $\arg\max$ capacity term excluded).
        $\widetilde{q}$-step: for each minibatch $\mathcal{B}$ in a second sequential sweep of $\mathcal{D}$, take a projected-subgradient step on $\widetilde{q}_i$ ($i \in \mathcal{B}\cap S$, step size η_q) and $\widetilde{q}_j$ ($j \in \mathcal{B}\cap T$, step size η_T) using the per-sample subgradients of (8) under the change of variables $q = w\widetilde{q}$.
        Joint projection: form $\mathbf{q}$ and project onto the box-and-sum set using source upper bounds $w_{s(i)}q_{\max}$ and target bounds $[q_T^{\min}, q_{\max}]$, then read back $\widetilde{q}_i = q_i/w_{s(i)}$ for $i \in S$ and $\widetilde{q}_j = q_j$ for $j \in T$.
    end if
    h-step: run the h-update of Algorithm A1 with composite weights $q_i = w_{s(i)}\widetilde{q}_i$ ($i \in S$) and $q_j = \widetilde{q}_j$ ($j \in T$).
end for
return θ.

Domain-weight diagnostics. In the stacking and mug-cleanup multi-source runs, each source contributes 250 demonstrations and receives equal sampling mass, so the learned domain weights are not explained by source size or sampling. On block stacking, the weights select the source with the lower mean discrepancy almost exclusively (0.98 versus 0.02). On mug cleanup, the two sources are nearly equidistant from the target (mean discrepancies 4.276 and 4.281), and the preference is correspondingly soft (0.70 versus 0.30, again favoring the closer source). With two sources, solving the domain-weight subproblem exactly reproduces the learned selections, and none of 129 tested settings of $(\rho_1, \rho_2, \lambda_1, \lambda_2, \lambda_d)$ reverses them. Within each source, the learned weights $\widetilde{q}_i$ correlate strongly with the discrepancy proxy (-0.91 to -0.99) but only weakly with the imitation loss (-0.04 to -0.29), suggesting that within-domain weighting mainly suppresses mismatched windows rather than difficult ones.

B.4 Implementation notes and essential hyperparameters

The policy backbone, optimizer, and learning-rate schedule follow the public Diffusion Policy implementation [6] and are kept identical across all baselines and BEACON variants. Training windows use stride one within each demonstration, repeating endpoints for padding, with observation, prediction, and execution horizons of 2, 16, and 8. Policy minibatches are drawn from a fixed source–target sampling mixture, uniformly within each domain, and the learned weights multiply the sampled losses. The BEACON-specific knobs that matter most in practice are the loss-coefficient triple $(\lambda_1, \lambda_2, \lambda_d) \approx (10^{-2}, 10^{-2}, 10^{-1})$, the per-sample upper bound $q_{\max} \approx 5$ together with the target floor $q_T^{\min} \in \{0.05, 0.10\}$ that prevents target weights from collapsing, the sum budget $\alpha \in [0.4, 0.5]$ in $\sum_i q_i = n + \alpha m$, the k-NN neighborhood $k \in \{4, 5, 6\}$, and the q-step size $\eta_q = 10^{-2}$. We use $K_q = 1$ for sim-to-sim and $K_q = 2$ for sim-to-real, with an initial refresh before policy training. For MS-BEACON the additional knobs are the source-level regularizers $(\rho_1, \rho_2) \approx (10^{-2}, 10^{-2})$ and the w-step size $\eta_w \in [10^{-2}, 10^{-1}]$. Sensitivity around these defaults is reported in Section D.

C Alternative q-update and discrepancy estimators

The default BEACON loop in Algorithm A1 couples a minibatch projected subgradient on $\mathbf{q}$ with a k-NN discrepancy estimator. We retain one alternative for $\mathbf{q}$ and two alternatives for the discrepancy term, all of which we ablate against the defaults.

Full-convex inner solve for q. A solver closer to Awasthi et al. [20]'s alternating minimization solves the $\mathbf{q}$ subproblem in closed form for fixed θ. Treating $t = \|\mathbf{q}\|_\infty$ as auxiliary, the inner problem at fixed (θ, t) becomes separable across coordinates and admits the closed-form solution

$$q_i^\star(t) \;=\; \mathrm{clip}_{[\underline{q}_i, t]}\Big(r_i + \mathcal{S}_{\lambda_1/(2\lambda_2)}\big(z_i(\theta) - r_i\big)\Big), \qquad z_i(\theta) := \frac{\mu - \ell_i(\theta) - \lambda_d\, d_i}{2\lambda_2}, \tag{9}$$

where $\mathcal{S}_\tau(u) = \mathrm{sign}(u)\max(|u| - \tau, 0)$ is the soft-threshold operator and μ is the Lagrange multiplier of the sum constraint. The convex outer problem $\min_{t \geq 0} [\,\mathrm{inner}(t) + \gamma\, R(\theta)\, t\,]$ is one-dimensional and convex, and we solve it by ternary search to floating-point precision. The per-sample sum constraint $\sum_i q_i = n + \alpha m$ is incorporated via a second bisection on μ, exactly as in line 12 of Algorithm A1. In this ablation, the costs $\ell_i(\theta) + \lambda_d d_i$ are divided by their maximum whenever it exceeds one.

Localized labeled discrepancy. The first BEA-style alternative is the most faithful to the original derivation of $\mathrm{dis}(P, Q)$ in Awasthi et al. [20], with the local-class refinement of Cortes et al. [21], Zhang et al. [22]. As a one-time pre-processing step before the main BEACON loop, we train a target-only reference policy $\pi_{\theta_{\mathrm{ref}}}$ on $\mathcal{D}_{\mathrm{tgt}}$, partition its parameters into the (frozen) observation encoder θ_{enc} and the (free) noise-prediction parameters θ_{free}, and define a parameter-space ball over the free part,

$$\mathcal{H}_{\mathrm{loc}} \;=\; \big\{\pi_{\theta'} : \theta'_{\mathrm{enc}} = \theta_{\mathrm{enc,ref}},\;\; \|\theta'_{\mathrm{free}} - \theta_{\mathrm{free,ref}}\|_2 \leq \rho\big\},$$

with radius $\rho = 0.05\|\theta_{\mathrm{free,ref}}\|_2$. We then run projected gradient ascent on $\theta' \in \mathcal{H}_{\mathrm{loc}}$ to maximize the empirical labeled discrepancy

$$\widehat{\mathrm{dis}}(P, Q; \theta') \;=\; \frac{1}{n}\sum_{j \in T} \ell_{\mathrm{BC}}(\theta'; x_j, y_j) \;-\; \frac{1}{m}\sum_{i \in S} \ell_{\mathrm{BC}}(\theta'; x_i, y_i), \tag{10}$$

Table 3: Average target success rate of eight BEACON variants on sim-to-sim *block stacking* with *texture + viewpoint gap*, evaluated on the strict S criterion. The two k-NN columns hold the estimator fixed while varying discrepancy granularity. The bolded entry is the default configuration used in Section 5.

	Localized	**k-NN**		**Classifier**
q-update	**labeled**	**Source mean**	**Instance-level**	
Full-convex inner solver	0.58	0.48	0.52	0.54
Stochastic update (default)	0.58	0.51	**0.60**	0.58

augmented with a soft proximity penalty $-\beta \left\|\theta'_{\text{free}} - \theta_{\text{free,ref}}\right\|_2^2$ ($\beta > 0$) that prevents gradient steps from concentrating mass on the boundary of the ball; the hard ball constraint is enforced by Euclidean projection of θ'_{free} after each step. Every few steps we evaluate $\widehat{\text{dis}}(P, Q; \theta')$ on the full labeled data, averaging over a small number of independent diffusion-noise/timestep draws to reduce the upward bias of taking a maximum over noisy estimates, and track the running maximum. We set $\hat{d}$ to this maximum, fix $d_i = \hat{d}\,\mathbf{1}[i \in S]$ and $\lambda_d = 1$, and freeze them for the entire BEACON run, so line 4 of Algorithm A1 becomes a no-op.

Domain-classifier discrepancy. The second alternative replaces the domain-level loss difference (10) with a per-source instance-level mismatch score derived from a binary domain classifier on the policy's observation-encoder features, an instance-level analogue of the $d_{\mathcal{A}}$-style discriminator estimators of Ganin et al. [15], Ben-David et al. [33]. At each refresh we (i) extract the current embeddings $\{\phi_\theta(x_i)\}_{i \in S \cup T}$ in a single forward pass through ϕ_θ, (ii) draw a class-balanced training set by random source undersampling so that source and target contribute equally (since typically $m \gg n$), (iii) fit a logistic-regression discriminator $g_\psi : \phi_\theta(x) \mapsto [0, 1]$ with target labeled 1 and source labeled 0, and (iv) apply g_ψ to *all* m source embeddings (not just the balanced subset) to obtain

$$d_i \;=\; 1 - g_\psi\big(\phi_\theta(x_i)\big) \;=\; \widehat{\Pr}\big(\text{source} \mid x_i\big), \qquad i \in S, \qquad d_j = 0 \text{ for } j \in T, \tag{11}$$

so that source samples that look target-like ($g_\psi \to 1$) get $d_i \to 0$ and clearly-source samples ($g_\psi \to 0$) get $d_i \to 1$. We additionally log $\text{AUC}(g_\psi)$ on the full pooled $S \cup T$ as a global indicator of domain separability, but use only the per-instance d_i from (11) inside the BEACON objective. Unlike the localized estimator, both g_ψ and $\{d_i\}_{i \in S}$ here are recomputed in place of line 4 of Algorithm A1 so that the score tracks the encoder ϕ_θ as it evolves.

Empirical comparison. Table 3 compares eight variants across two $\mathbf{q}$-update rules and three estimator families, with two score granularities for k-NN. To isolate discrepancy granularity, the source-mean variant replaces each source score d_i by $\bar{d} = \frac{1}{m}\sum_{i \in S} d_i$, while retaining the same estimator and weight-update rule. The sample weights remain adaptive; only the discrepancy scores are shared across source samples. With the stochastic update, instance-level scores achieve 0.60 strict stacking success, compared with 0.51 for source-mean scores; with the full-convex update, the corresponding scores are 0.52 and 0.48. The localized and classifier columns separately compare estimator families. The default stochastic update with instance-level k-NN scores achieves the highest score.

D Hyperparameter sensitivity analysis

We conduct a sensitivity analysis over the training hyperparameters, including the L_1 deviation coefficient λ_1, the L_2 shrinkage λ_2, the discrepancy weight λ_d, and the sum budget α. We sweep one hyperparameter at a time, retaining the value with the highest target success rate before moving to the next. A single sweep on sim-to-sim *block stacking* is reused unchanged for *mug cleanup* and *threading*; a separate sim-to-real stacking sweep is reused for *mug cleanup* and *ball rollover*. OOD evaluations are never used for selection. The coefficient λ_2 discourages concentrated weights, λ_d scales the discrepancy penalty, and α sets the total-weight budget through the projection in line 12 of Algorithm A1. Under the stated target floors and fixed total-weight budget, the inherited L_1 term is constant over the feasible set.

The sim-to-sim sweep identifies a stable region with $\lambda_1 = 0.01$, $\lambda_2 \in \{0.01, 0.001\}$, $\lambda_d \in \{0.1, 0.05\}$, and $\alpha \in [0.3, 0.5]$. Sim-to-real uses smaller λ_d to accommodate larger discrepancy scores and smaller

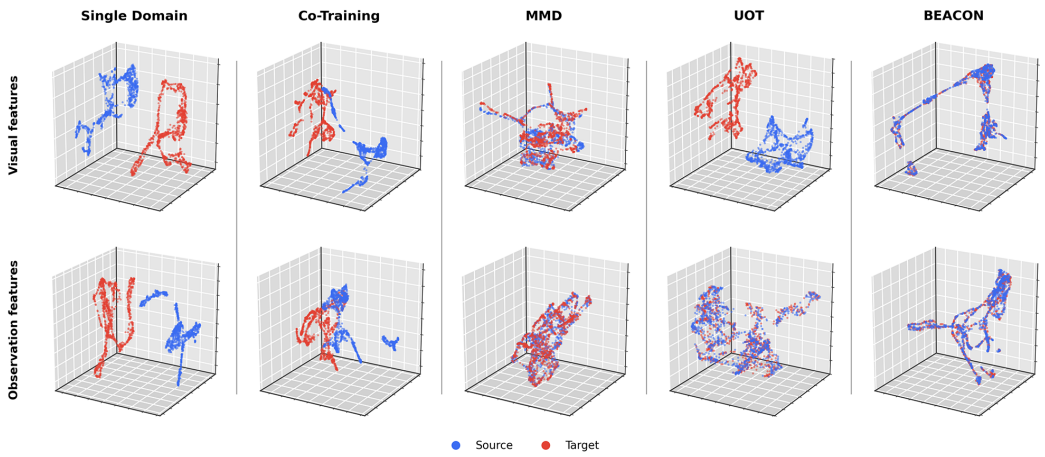

Figure A2: UMAP visualization of latent features (*mug cleanup*, sim-to-sim texture gap). As in Figure 3, the top row shows features of the image alone and the bottom row all observations (visual + proprioception) after the encoder trunk.

α to reduce the total-weight budget. Stored-checkpoint audits of the default runs found no oscillation or zeroed source weights.

For a new transfer setting, the self-normalized discrepancy scores make the two active knobs readable from the data. Since $d_i \approx 1$ means that a source window lies within the typical target spread, α can be set from the empirical histogram of d_i (roughly the fraction of source windows with d_i near or below 1), and λ_d at the scale that makes $\lambda_d d_i$ comparable to a typical target behavior-cloning loss; both quantities follow from one forward pass of a target-only policy. Where target data can be spared, a held-out target split selected on validation behavior-cloning loss serves the same purpose. We offer this as a heuristic starting point; the reported runs used the sweep protocol above.

For the Tuned Uniform baseline, we sweep the common source weight over $\{0.1, 0.5, 0.9, 2, 8, 16\}$ on block stacking, select 8, and reuse it across all sim-to-sim tasks and visual gaps.

E Feature alignment

Figure A2 complements the stacking visualization with latent features from *mug cleanup*. We quantify source–target alignment on the frozen policy encoders from the sim-to-sim texture-gap experiments, using the full observation features (visual and proprioceptive), with the RBF-MMD between source and target features and the held-out error of a linear domain classifier (Table 4). Without an explicit alignment loss, BEACON reduces RBF-MMD relative to co-training by about $3\times$ on block stacking and $10\times$ on mug cleanup, and its features are harder to separate by domain (probe error 0.13 and 0.41, versus 0.00 for co-training). The MMD baseline, which minimizes this discrepancy directly, attains the lowest values on both tasks without the highest task success. Holding BEACON's trained encoder fixed, reweighting the source features by $\mathbf{q}$ changes the RBF-MMD by less than 0.01 on both tasks (paired bootstrap intervals include zero), so the closer alignment resides in the trained encoder and does not depend on reweighting at evaluation time.

F Source weight evolution during training

The $\mathbf{q}$-evolution during training and weight distributions for *block stacking* task are shown in Figure A3. The target mean weight rises steeply during training from ~ 0.5 to ~ 1.9 over 600 epochs, while the source mean weight drifts from 0.51 to 0.46. At the end of training, target weights are tightly concentrated (mean 1.93, std 0.07), well below the cap $q_{\max} = 5$. Their increase reflects redistribution under the total-weight budget. Across all 12 reported runs, no weight reaches either edge of its box, and the largest weight is 11–63% of $q_{\max}$. After normalization by $\sum_i q_i$, all weights lie in $[0, 1]$; because the admitted source mass exceeds the target mass, however, the normalized

Table 4: Source–target feature alignment on frozen policy encoders (sim-to-sim texture gap, full observation features, sampled source windows unweighted). RBF-MMD: lower values indicate closer alignment. ϵ: held-out error of a linear domain classifier (0.5 is chance); $\hat{A} = \max\{0, 2(1 - 2\epsilon)\}$ is the proxy $\mathcal{A}$-distance, with lower values indicating less separable features.

	Co-Training	MMD	UOT	BEACON
RBF-MMD, block stacking	0.534	**0.082**	0.124	0.168
RBF-MMD, mug cleanup	0.460	**0.028**	0.041	0.045
ϵ / $\hat{A}$, block stacking	0.00 / 2.00	0.58 / 0.00	0.00 / 1.99	0.13 / 1.49
ϵ / $\hat{A}$, mug cleanup	0.00 / 2.00	0.52 / 0.00	0.00 / 2.00	0.41 / 0.37

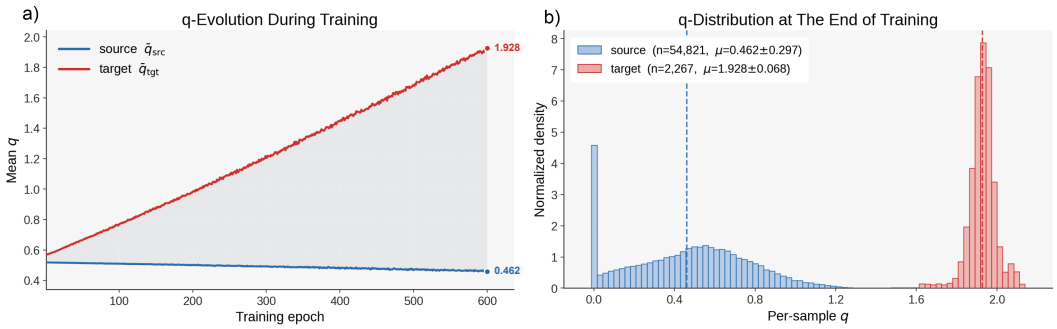

Figure A3: a) $\mathbf{q}$-evolution during training. The average $\mathbf{q}$ values for source windows (blue) decrease over time, while the average $\mathbf{q}$ values for target windows (red) increase at a steeper rate. b) $\mathbf{q}$-distribution at the end of training. Density is normalized to the total number of samples in each domain so the two very different sample counts are comparable. Both plots are for the *block stacking* task.

weights do not satisfy BEA's condition $\|\mathbf{q} - \mathbf{p}_0\|_1 < 1$ (Theorem 1). Source weights are more dispersed (mean 0.46, std 0.30), with a near-zero peak and a broader mode around 0.5.

Additional audits show that weighting acts mainly within demonstrations, with 78–96% of the source-weight variance lying within individual trajectories. Stretches with $q_i < 0.05$ have a median length of five to six frames. In sim-to-real stacking, high-weight windows tend to occur during slow placement near the end of the episode, while low-weight windows occur during faster transport. This pattern shows how the learned weights distinguish task phases within a demonstration.

G Compute resources

Experiments use a single NVIDIA L40S GPU with 48 GB of memory per run, and a typical BEACON run completes within 24 hours. In the reported block-stacking profile, a BEACON epoch takes 40.3 s, compared with 38.7 s for co-training. Across profiled settings, the per-epoch ratio ranges from $1.04\times$ to $4.2\times$, with the largest overhead on sim-to-real mug cleanup at approximately 180,000 observation windows. Additional cost comes mainly from the no-gradient passes used to extract features and evaluate per-sample losses, together with data loading. Multi-source weight refresh takes 1.1–$1.4\times$ the single-source refresh time.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The paper's contributions and scope are accurately and thoroughly described in the abstract and introduction.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: The discussion of limitations is present in the paper's Conclusion section.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

Justification: The only theoretical result in the paper, Theorem 1 in Section 3, is a restatement of Corollary 4 of Awasthi et al. [20] (the original BEA paper) specialized to our notation; we state its full set of assumptions directly in the theorem statement, and refer the reader to the cited paper for the proof rather than reproducing it, since the bound is invoked as prior work rather than claimed as a new theoretical contribution.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: All implementation details to reproduce the experiments are provided in the Appendix.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: We have released the code at the project page (`https://computationalrobotics.seas.harvard.edu/BEACON/`). The Appendix additionally provides detailed pseudocode, hyperparameter settings, and experimental settings.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: Training and test details are clearly specified in the Method and Appendix sections.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No]

   Justification: The primary experimental results to report are the average success rates of the learned policies in simulation and real-world scenarios. To make the average success rates reliable, we performed 100 evaluations in simulation and 20 evaluations in real-world scenarios, due to the high cost of real-world evaluations.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.

- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: The computer resources required to reproduce the experiments are detailed in the Appendix.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: Our work is in compliance, in every respect, with the NeurIPS Code of Ethics.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [N/A]

Justification: We propose a framework that improves cross-domain robot policy co-training, hence we do not expect any immediate societal impacts.

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The paper poses no such risks.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We build upon existing open-source code bases, and we credit the original authors with appropriate citations and direct mentions of their work.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.

- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A]

    Justification: The paper does not release new assets.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: The paper does not involve LLMs as any important, original, or non-standard components.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.