PLASDecoupledCVAE: A Decoupled and Adaptive CVAE Framework for Offline Reinforcement Learning

M. Asif Hasan
Albert-Ludwigs-Universität
Freiburg, Germany
hasana@cs.uni-freiburg.de

André Biedenkapp
Albert-Ludwigs-Universität
Freiburg, Germany
biedenka@cs.uni-freiburg.de

Noor Awad
Albert-Ludwigs-Universität
Freiburg, Germany
awad@cs.uni-freiburg.de

ABSTRACT

The performance and computational efficiency of latent-space offline Reinforcement Learning (RL) methods critically depend on the quality of learned representations. In approaches such as Policy in Latent Action Space (PLAS), this representation is obtained via a Conditional Variational Autoencoder (CVAE) trained during a fixed warmup phase prior to policy optimization. We propose PLASDecoupledCVAE, a modular extension that explicitly decouples CVAE pre-training from policy learning. Our framework exposes the CVAE warmup as an independent, configurable stage equipped with adaptive convergence detection based on reconstruction-loss plateaus. This design enables a train-once, reuse-many paradigm, allowing a learned latent action space to be reused across multiple policy optimization runs and substantially reducing computational overhead. From an Automated Reinforcement Learning (AutoRL) perspective, we argue that the CVAE pre-training schedule should be treated as a tunable, data-dependent component rather than a fixed-step hyperparameter. We empirically evaluate fixed and adaptive warmup strategies on the Fetch robotic manipulation benchmark across multiple convergence thresholds. Our results show that adaptive warmup can yield performance gains of up to +21.92% over the standard PLAS baseline, although optimal configurations are task-dependent. Notably, we find that lower CVAE reconstruction loss does not guarantee better policy performance, highlighting the non-trivial relationship between generative model convergence and downstream task success. We release our implementation as a community package.[1]

KEYWORDS

Offline Reinforcement Learning, Latent Action Space, Representation Learning, Robotic Manipulation

1 INTRODUCTION

Offline RL introduces a fundamental challenge of distribution shift. The learned policy may query the value of actions that are out-of-distribution (OOD) with respect to the dataset, leading to erroneous Q-value estimation and catastrophic performance [6]. State-of-the-art algorithms address this by constraining the learned policy to remain "close" to the data's behavior policy.

Policy in Latent Action Space (PLAS) [16] employs a Conditional Variational Autoencoder (CVAE) [13] to model the distribution of actions conditioned on states. The policy outputs latent codes, which the CVAE decoder maps to executable actions. This architecture implicitly constrains the policy to actions plausible under the behavioral distribution, mitigating extrapolation errors without explicit pessimism penalties.

1.1 Motivation: The Need for Decoupled Training

Despite its effectiveness, standard PLAS implementation, such as that in d3rlpy [12], tightly couples CVAE training with policy optimization in a monolithic training loop. This design presents several practical limitations: (1) **Redundant computation:** The CVAE is retrained from scratch for every policy hyperparameter configuration, even though the optimal latent space depends primarily on the dataset, not the policy; (2) **Opaque convergence:** The CVAE warmup phase uses a fixed step count, with no mechanism to detect when the generative model has sufficiently converged; (3) **Limited reusability:** The standard d3rlpy [12] implementation treats the CVAE as an internal, non-serializable component of the training loop. This architectural limitation prevents practitioners from reusing a well-trained CVAE across experiments, hindering systematic hyperparameter exploration.

Fundamentally, the CVAE's role is to learn meaningful representations that capture the characteristics of the behavioral action distribution. This representation learning is general-purpose and does not require policy-specific training, enabling efficient reuse across different policy configurations and potential fine-tuning for related tasks. This strategy parallels 'predictive identification' in meta-reinforcement learning, where latent models are decoupled to capture global dynamics [2, 15], contrasting with coupled methods designed to capture behavior-specific contexts for zero-shot adaptation [7]. This property becomes particularly valuable in AutoRL and hyperparameter optimization workflows, where hundreds of policy configurations may need to be evaluated against the same learned representation. Retraining the CVAE for each trial introduces substantial redundant computation.

This motivates a fundamental question: **Can we refactor the PLAS workflow to make CVAE training a reusable, data-driven, and more efficient process?**

1.2 Contribution: PLASDecoupledCVAE

We introduce PLASDecoupledCVAE, a modular refactoring of the PLAS algorithm that addresses these limitations through architectural decoupling. Our framework (1) **Decouples CVAE pre-training:** The CVAE warmup phase is extracted into an independent stage that can be executed once and cached for reuse across multiple policy training runs; (2) **Introduces adaptive convergence detection:** Rather than fixed-step warmup, we implement

[1]Code available at https://github.com/somenomus/PLASDecoupledCVAE

Proc. of the Adaptive and Learning Agents Workshop (ALA 2026), Aydeniz, Delgrange, Mohammedalamen, Yang (eds.), May 25 – 26, 2026, Paphos, Cyprus, https://alaworkshop2026.github.io/. 2026.

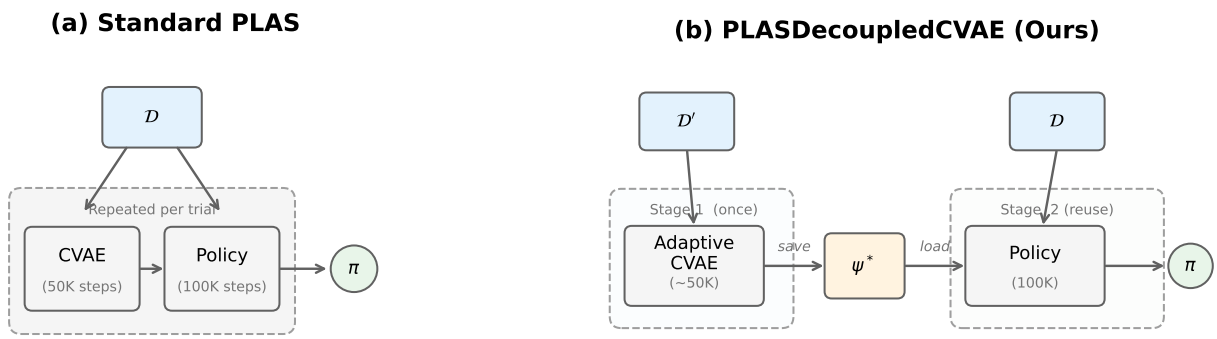


Figure 1: Comparison of training workflows. (a) Standard PLAS: CVAE and policy training are coupled. (b) PLASDecoupledCVAE (Ours): The CVAE is trained once on distilled data and frozen; multiple policy trials reuse the same latent space.

a plateau-based early stopping mechanism that monitors reconstruction loss and terminates pre-training when convergence is detected. (3) **Exposes tunable hyperparameters:** The framework introduces three new hyperparameters, convergence threshold (δ), window size (W), and patience (P), enabling practitioners to tune CVAE pre-training for their specific domains; (4) **Enables efficient exploration:** The train-once, reuse-many paradigm significantly reduces computational cost for hyperparameter optimization, as the CVAE does not require retraining during each trial of the policy.

Figure 1 illustrates the architectural difference between standard PLAS and our decoupled approach.

From an Automated Reinforcement Learning (AutoRL) perspective, our framework treats the CVAE pre-training as a distinct, reusable "artifact" and its training schedule as a tunable, data-dependent component of the RL pipeline. We empirically validate our framework on robotic manipulation benchmarks, demonstrating the adaptive mechanism.

2 BACKGROUND AND RELATED WORK

An effective approach to implicit policy constraint is to learn policies in a latent action space. Policy in Latent Action Space (PLAS) [16] constrains the policy by design through the following mechanism: a Conditional Variational Autoencoder (CVAE) is trained to model the behavior policy distribution $p(a|s)$, and the learned policy $\pi(z|s)$ operates over the latent variable z rather than directly in the action space. The decoder of the CVAE maps latent actions to environmental actions $a = D(z, s)$, ensuring that the policy remains implicitly constrained to the support of the learned latent distribution.

This structural constraint offers significant advantages over explicit regularization approaches used in competing methods. BCQ [4] uses explicit density estimation combined with perturbations to restrict actions, while BEAR [5] employs divergence penalties (specifically, Maximum Mean Discrepancy) to maintain proximity to the behavior policy. By contrast, PLAS constrains the policy implicitly by operating in the latent space of a CVAE fitted to the behavior actions, avoiding explicit density thresholds or divergence penalty tuning as in BCQ and BEAR, while achieving competitive empirical performance [16]. Recent theoretical and empirical work shows that offline RL can suffer from substantial error amplification under distribution shift, and that stability depends critically on the properties of the used representation [14]. Complementary lower-bound results demonstrate that sample-efficient offline RL requires coverage or representation conditions that go beyond those sufficient for standard supervised learning guarantees [3].

Building on the dataset evaluation framework of Schweighofer et al. [10], which shows offline RL performance depends on TQ and SACo, we extend these metrics to curate high-quality distilled datasets. This finding motivates a data-centric perspective on offline RL: rather than viewing the dataset as a fixed artifact, the framework explicitly considers dataset composition as part of the learning pipeline. Our approach operationalizes this principle.

From an AutoRL perspective, optimizing the RL pipeline extends beyond algorithm and hyperparameter selection to include the structure and composition of training stages, aligning with AutoRL goals of modular pipeline optimization [8]. Effective AutoRL requires identifying which components of the pipeline can be decoupled, reused, and made data-dependent rather than fixed.

These considerations motivate our work: we decouple the CVAE pre-training stage from policy optimization, enabling systematic exploration of convergence criteria while reducing computational overhead.

3 METHODOLOGY

We introduce a comprehensive framework that comprises two core components: (1) a dataset optimization framework, and (2) a decoupled CVAE pre-training and policy optimization pipeline called PLASDecoupledCVAE. Additionally, we propose an Adaptive Warmup mechanism that automatically determines CVAE convergence based on loss plateauing, transforming the fixed `warmup_steps` hyperparameter into a data-dependent, tunable component aligned with Automated Reinforcement Learning principles.

3.1 Dataset Optimization Framework

To identify and retain the most informative trajectories, we evaluate each dataset using two complementary metrics that jointly capture task performance and behavioral diversity. Building upon the evaluation framework of [10], we compute a Trajectory Quality (TQ)

score and a State–Action Coverage (SACo) score for every episode (Lines 5 of Algorithm 1). Our dataset optimization framework is detailed in the following paragraphs and outlined in Algorithm 1.

Trajectory Quality (TQ) Assessment: The Trajectory Quality score quantifies the task performance of individual episodes (Line 5). For episode i, we compute:

$$\mathrm{TQ}_i = \sum_{t=0}^{T_i-1} \gamma^t r_{i,t} \tag{1}$$

In our implementation, TQ is the undiscounted cumulative episode reward used for trajectory ranking during dataset curation. Because the Fetch dense tasks accumulate step-wise reward over the full trajectory, shorter successful episodes still tend to obtain better TQ values by incurring fewer low-reward steps.

State–Action Coverage (SACo) Analysis: Although the Trajectory Quality (TQ) metric captures task performance, it tends to favor a narrow set of near-optimal trajectories, which can reduce dataset diversity. The State–Action Coverage metric complements this by measuring how broadly an episode explores the environment.

We first extract all state-action pairs from the entire dataset:

$$SA_{\text{global}} = \bigcup_{i=1}^{M} \{(s_t^{(i)}, a_t^{(i)})\}_{t=1}^{T_i-1} \tag{2}$$

where indexing begins at $t = 1$ to ensure compatibility with the d3rlpy MDPDataset format during data conversion from Stable Baselines3, which requires proper alignment of terminal flags and transition indexing. Then, we apply MiniBatch K-Means clustering [11] to partition the global state-action space(Lines 3):

$$C = \{C_1, C_2, \ldots, C_k\} = \text{MiniBatchKMeans}(SA_{\text{global}}, k) \tag{3}$$

Where the number of clusters k is selected empirically as a compromise between representational fidelity, the degree to which clusters accurately reflect variations in the state–action space, and generalization, ensuring scalability with dataset size. A large k value improves representational fidelity by capturing fine-grained detail but risks overfitting and poor generalization. In contrast, a small k value is more general but risks creating a coarse representation that loses fidelity. Our choice of k therefore represents a compromise, tailored to the size and complexity of each environment's dataset. See Appendix C for a detailed sensitivity analysis validating this selection.

For each episode i, we compute its State–Action Coverage as:

$$\mathrm{SACo}_i = \frac{|V_i|}{k} \tag{4}$$

Where V_i is the set of unique cluster indices visited by episode i, specifically, we assign each state-action pair $(s_t^{(i)}, a_t^{(i)})$ to its nearest cluster and count the number of distinct clusters visited. The SACo score ranges from 0 to 1, where a score of 0 indicates an episode explores very few unique explorations, and a score of 1 signifies that it covers diverse regions of the environment.

Episodes with higher SACo scores contribute more behavioral diversity to the dataset, helping maintain good coverage of the environment, even with reduced data.

Episode Selection: To construct the optimized dataset $\mathcal{D}'$, we select the most valuable episodes from the original dataset $\mathcal{D}$. Since we aim to balance both trajectory quality and State–Action Coverage, we combine these two metrics into a single scalar score, allowing us to rank all episodes.

We first normalize both metrics to ensure equal contribution (Lines 10 - 11):

$$\mathrm{TQ}_i^{\text{norm}} = \frac{\mathrm{TQ}_i - \min_m(\mathrm{TQ}_m)}{\max_m(\mathrm{TQ}_m) - \min_m(\mathrm{TQ}_m)} \tag{5}$$

$$\mathrm{SACo}_i^{\text{norm}} = \frac{\mathrm{SACo}_i - \min_m(\mathrm{SACo}_m)}{\max_m(\mathrm{SACo}_m) - \min_m(\mathrm{SACo}_m)} \tag{6}$$

where the min and max are computed over all episodes m in the dataset.

We then define a combined score for each episode i as (Line 14):

$$\mathrm{Score}_i = w_{\mathrm{TQ}} \cdot \mathrm{TQ}_i^{\text{norm}} + w_{\mathrm{SACo}} \cdot \mathrm{SACo}_i^{\text{norm}} + \mathbf{1}[\text{success}_i] \tag{7}$$

where w_{TQ} and w_{SACo} are the weights for each metric. $\mathbf{1}[\text{success}_i]$ is an indicator function that equals 1 if episode i achieves the goal, and 0 otherwise. This success indicator ensures successful episodes rank higher than non-successful ones. Using this score, we rank all episodes and select the top episodes to form the optimized dataset $\mathcal{D}'$. During this selection process, we ensure the proper terminal flags and episode integrity, as the length of episodes is variable.

Algorithm 1 Dataset Optimization for Offline RL

Require: Dataset $\mathcal{D}$, retention rate α, weights $w_{\mathrm{TQ}}, w_{\mathrm{SACo}}$, cluster count k, success threshold τ

Ensure: Optimized dataset $\mathcal{D}'$

1: $\mathcal{D}_{\text{unique}} \leftarrow \text{DeduplicateEpisodes}(\mathcal{D})$ ▷ Filter unique episodes using hashing
2: $SA_{\text{global}} \leftarrow \bigcup_{e_i \in \mathcal{D}_{\text{unique}}} \{(s_t^{(i)}, a_t^{(i)}) \mid t \geq 1\}$ ▷ Collect transitions, excluding the first step
3: $C \leftarrow \text{MiniBatchKMeans}(SA_{\text{global}}, k)$ ▷ Train global state-action clusterer
4: **for** each episode $e_i \in \mathcal{D}_{\text{unique}}$ **do**
5: $\quad \mathrm{TQ}_i \leftarrow \sum_{t=0}^{T_i-1} r_t^{(i)}$
6: $\quad V_i \leftarrow \{\text{predict}(C, (s_t^{(i)}, a_t^{(i)})) \mid t \geq 1\}$
7: $\quad \mathrm{SACo}_i \leftarrow |\text{unique}(V_i)|/k$
8: $\quad \text{success}_i \leftarrow \mathbf{1}[\|s_{T_i}^{\text{achieved}} - s_{T_i}^{\text{desired}}\|_2 \leq \tau]$
9: **end for**
10: $\mathrm{TQ}^{\text{norm}} \leftarrow \text{MinMaxScale}(\mathrm{TQ})$
11: $\mathrm{SACo}^{\text{norm}} \leftarrow \text{MinMaxScale}(\mathrm{SACo})$
12: $\mathrm{Score}_i \leftarrow w_{\mathrm{TQ}} \cdot \mathrm{TQ}_i^{\text{norm}} + w_{\mathrm{SACo}} \cdot \mathrm{SACo}_i^{\text{norm}} + \text{success}_i$
13: $\mathcal{I}_{\text{success}} \leftarrow \{i \mid \text{success}_i = 1\}$
14: $\mathcal{I}_{\text{ranked}} \leftarrow \text{Argsort}(\{\mathrm{Score}_j \mid j \notin \mathcal{I}_{\text{success}}\})$ ▷ Sort non-successful episodes by score
15: $n \leftarrow \lfloor \alpha \cdot |\mathcal{D}_{\text{unique}}| \rfloor$
16: $\mathcal{I}_{\text{final}} \leftarrow \text{Unique}(\text{Concatenate}(\mathcal{I}_{\text{success}}, \mathcal{I}_{\text{ranked}}))[:n]$
17: $\mathcal{D}' \leftarrow \{e_i \mid i \in \mathcal{I}_{\text{final}}\}$
18: **return** $\mathcal{D}'$

3.2 PLASDecoupledCVAE Framework

Standard implementations of PLAS [12, 16] typically intertwine representation learning and policy optimization into a monolithic training loop governed by a fixed `warmup_steps` hyperparameter. This coupled design creates computational redundancy, particularly during hyperparameter optimization (HPO), as the generative model must be retrained for every new policy trial.

We introduce `PLASDecoupledCVAE`, a modular framework that refactors this process into two strictly distinct, serializable stages: (1) Adaptive CVAE Pre-training and (2) Policy Optimization with Frozen Latent Space. The complete training procedure is detailed in Algorithm 2.

Stage 1: Adaptive CVAE Pre-training. The first stage focuses exclusively on learning the conditional distribution $p(a|s)$ of the behavior policy. The objective is to minimize the reconstruction loss of the CVAE on the distilled dataset $\mathcal{D}'$ (defined in Section 3.1).

Unlike the standard PLAS fixed warmup, we implement a novel Adaptive Warmup mechanism. We define a sliding window of size W and monitor the moving average of the reconstruction loss $\mathcal{L}_{rec}$. At intervals of W steps (Line 5), we compute the relative improvement Δ_{imp} between the previous window mean μ_{prev} and the current window mean μ_{curr} (Line 7):

$$\Delta_{imp} = \frac{\mu_{prev} - \mu_{curr}}{\mu_{prev}} \tag{8}$$

The training duration is governed by two hyperparameters: a convergence threshold δ and a patience counter P. The stopping condition is triggered if the relative improvement falls below the threshold ($\Delta_{imp} < \delta$) for P consecutive checks (Lines 9–10). This mechanism ensures that training automatically terminates when the latent representation stabilizes, preventing unnecessary computation while adapting to the specific complexity of the distilled dataset. Upon convergence, the trained CVAE is saved for reuse in subsequent policy optimization (Line 14).

Stage 2: Policy Optimization with Frozen Latent Space. In the second stage, the pre-trained CVAE weights ψ^* are loaded and strictly frozen ($\nabla_\psi J = 0$) as shown in Line 15. The policy π_ϕ and critic Q_θ are then initialized and trained using the standard PLAS objective on the full dataset $\mathcal{D}_{full}$. The objective function for the policy explicitly relies on the fixed decoder D_{ψ^*} to map latent actions z to environmental actions a (Line 19):

$$J(\phi) = \mathbb{E}_{s\sim\mathcal{D}_{full}}\left[Q_\theta(s, D_{\psi^*}(\pi_\phi(s), s))\right] \tag{9}$$

By decoupling these stages, we ensure that the policy search is constrained by the high-quality latent space learned from the distilled data $\mathcal{D}'$, while the value estimation benefits from the dense reward signals available in the full dataset $\mathcal{D}_{full}$. Furthermore, because the latent space is fixed, HPO can be performed solely on the policy and critic hyperparameters, significantly reducing the computational cost of the search. Crucially, because the stages are independent, they can operate on different data distributions, a flexibility not available in coupled training schemes.

Algorithm 2 PLASDecoupledCVAE Training Framework

Require: Distilled Dataset $\mathcal{D}'$, Full Dataset $\mathcal{D}_{full}$
Require: CVAE params ψ, Policy params ϕ, Critic params θ
Require: Adaptive Hyperparams: Threshold δ, Window W, Patience P
Ensure: Optimized Policy π_ϕ

Stage 1: Adaptive CVAE Pre-training
1: Initialize CVAE weights ψ, loss history $\mathcal{L}_{hist}$, counter $c \leftarrow 0$
2: **while** not converged **do**
3: Sample batch $B \sim \mathcal{D}'$ ▷ Use Distilled Dataset
4: $\mathcal{L}_{rec} \leftarrow$ UpdateCVAE(ψ, B); Append to $\mathcal{L}_{hist}$
5: **if** $|\mathcal{L}_{hist}| \pmod W == 0$ **then** ▷ Check every W steps
6: $\mu_{curr} \leftarrow$ Mean($\mathcal{L}_{hist}[-W:]$); $\mu_{prev} \leftarrow$ Mean($\mathcal{L}_{hist}[-2W:-W]$)
7: $\Delta_{imp} \leftarrow (\mu_{prev} - \mu_{curr})/\mu_{prev}$
8: **if** $\Delta_{imp} < \delta$ **then** $c \leftarrow c + 1$ **else** $c \leftarrow 0$
9: **end if**
10: **if** $c \geq P$ **then break** ▷ Early Stopping
11: **end if**
12: **end if**
13: **end while**
14: $\psi^* \leftarrow \psi$ ▷ Serialize and Freeze weights

Stage 2: Policy Optimization
15: Load frozen weights ψ^*; Initialize ϕ, θ
16: **for** step $t = 1$ to N_{policy} **do**
17: Sample batch $B \sim \mathcal{D}_{full}$ ▷ Use Full Dataset
18: $z \leftarrow \pi_\phi(s)$
19: $a \leftarrow D_{\psi^*}(z, s)$ ▷ Decode using frozen decoder
20: Update Critic θ on (s, a, r, s')
21: Update Policy ϕ to maximize $Q_\theta(s, a)$
22: **end for**
23: **return** π_ϕ

4 EXPERIMENTAL SETUP

Our experiments are designed to answer two key questions: (1) Does decoupling the CVAE training workflow affect downstream policy performance? and (2) Can the adaptive warmup mechanism reduce computational costs without compromising task success?

4.1 Environments

We evaluated our method on four robotic manipulation tasks from the Fetch environment suite in Gymnasium-Robotics [1]. Fetch-ReachDense-*v*2, FetchPushDense-*v*2, FetchPickAndPlaceDense-*v*2 and FetchSlideDense-*v*2. These environments simulate a 7-DoF Fetch robotic arm with a two-finger parallel gripper operating in a tabletop setting. These environments differ mainly in their objectives, observation dimensionality, and manipulation complexity, as detailed in Appendix A.1.

4.2 Dataset Generation

To construct the offline datasets used for training and evaluation, we generated demonstrations using behavior policies as follows.

Behavior Policy Training: For each environment, we train two behavior policies using deep deterministic policy gradient (DDPG)

with hindsight experience replay (HER) using the Stable Baselines3 v2.3.2 [9] library with the default hyperparameters are as follows:

(1) Expert Policy: Trained until convergence (success rate > 90%)
(2) Medium Expert Policy: Saved model of 60-70% of the expert training time.

Training configuration: All policies were trained using the same default configuration unless otherwise specified:

- Algorithm: DDPG (MultiInputPolicy) with HER.
- Network Architecture: Actor and Critic with library default.
- Learning Rate: 0.0003
- HER Strategy: "future" with four sampled goals per transition.
- Environment Setup: Vectorized with four parallel environments.
- Noise: Ornstein Uhlenbeck Action Noise.

We trained DDPG-HER agents independently on four distinct environments using a consistent set of default hyperparameters. The total training steps and replay buffer capacity for each agent were varied to accommodate the differing difficulty levels among the environments. The Table 1 (Appendix A.2) contains the size of the final dataset for each environment.

Dataset collection Process: We collected offline datasets across all four Fetch environments using previously trained DDPG-HER policies. The collected trajectories were then converted into d3rlpy's MDPDataset format [12] for offline RL training. Adapting datasets generated with Stable Baselines3 [9] for use in d3rlpy v2.5.0 [12] required careful preprocessing to ensure compatibility and correctness. The following steps were applied during data conversion:

- We concatenated the observation dictionary into a single feature vector, combining `observation`, `achieved_goal`, and `desired_goal` into a flattened state representation whose dimensionality depends on the environment.
- The Fetch Dense environments never naturally terminate. Thus, needed to manually detect success from `info["is_success"]` with early stopping after a minimum of 2 steps. Upon success, we set `terminal=True` and broke the episode loop to prevent unnecessary steps.
- We properly distinguished between episode outcomes: successful episodes were marked with `terminal=True, timeout=False`, while episodes reaching `max_steps` without success were marked with `terminal= False, timeout=True`.
- We ensured data consistency by validating that terminals and timeouts are never simultaneously `True`.

Each dataset contains mixed-quality policies (50% expert, 50% medium) transitions.

Dataset optimization Process: Following the framework in Section 3.1, we apply the optimization procedure to each collected dataset. We first filter episodes shorter than a minimum length threshold to ensure adequate trajectory information for metric computation. To ensure dataset integrity, we implement hash-based deduplication (Line 1) using MD5 checksums computed from concatenated observations and actions. Episodes are then scored based on TQ, SACo, and success metrics (Lines 5–12), ranked by descending score, and the top α fraction is retained to form the optimized dataset (Lines 14).

4.3 Hybrid Training Strategy and Configurations

Leveraging the data flexibility of our decoupled architecture (Section 3.2), we implement a Hybrid Data Scheme that consists of two stages as follows:

Stage 1 (Representation): The CVAE is pre-trained on the distilled Dataset ($\mathcal{D}'$, $\alpha = 0.5$), retaining only the top 50% of trajectories based on the TQ-SACo metric (Section 3.1). This ensures the latent action space is constructed exclusively from high-quality, diverse behaviors, effectively filtering out noise and suboptimal actions from the manifold.

Stage 2 (Reinforcement): The Policy is trained on the Full Dataset ($\mathcal{D}_{full}$). This ensures the critic has access to the complete transition history, including suboptimal transitions to minimize extrapolation error and provide accurate value estimation across the entire state space.

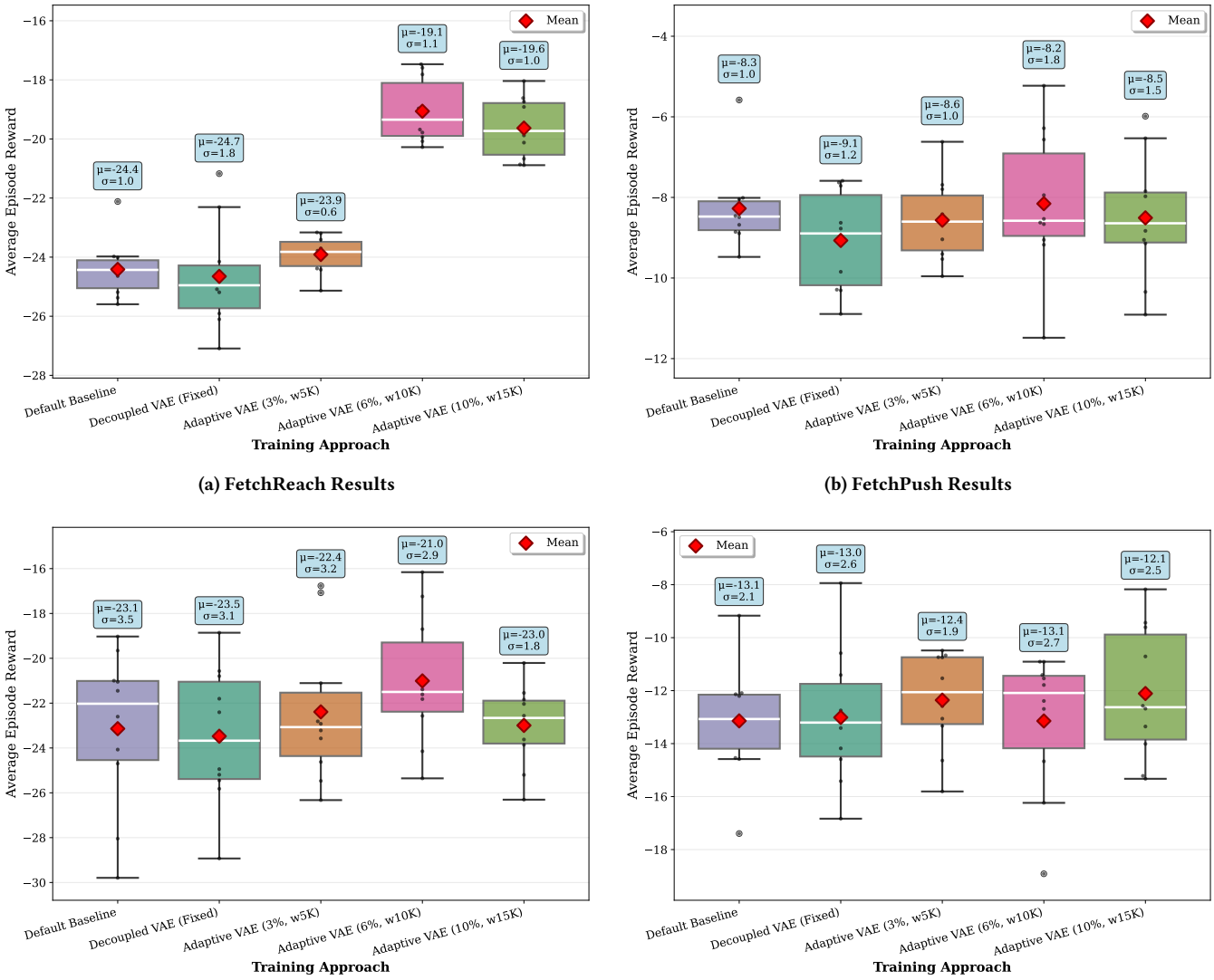
To evaluate the impact of this decoupled architecture and the adaptive mechanism, we define five specific training configurations:

(1) **PLAS Baseline (Coupled):** The standard monolithic training loop where the CVAE is trained on $\mathcal{D}_{full}$ with fixed warmup steps.
(2) **Decoupled (Fixed):** Our framework utilizing the Hybrid Data Scheme. The CVAE is pre-trained on $\mathcal{D}'$ for a fixed duration and then frozen.
(3) **Adaptive** (δ, W)**:** To demonstrate the framework's flexibility, we selected a range of convergence thresholds $\delta \in (0.03, 0.06, 0.10)$ spanning conservative to permissive early stopping criteria, paired with corresponding window sizes $W \in (5\text{k}, 10\text{k}, 15\text{k})$ that provide increasing smoothing for convergence detection. These values were chosen to illustrate the trade-off between training duration and convergence precision, rather than as optimized recommendations for any specific task.

The patience parameter was fixed at $P = 2$ across all experiments, requiring two consecutive below-threshold checks before early stopping. Full implementation details and hyperparameters are provided in Appendix A.3.

5 RESULTS

Figure 2 summarizes the complete results, with performance measured by mean cumulative reward. We observe that adaptive warmup yields larger performance gains on FetchReach (+21.92%) and FetchSlide (+9.20%), which primarily involve end-effector positioning and momentum transfer, compared to FetchPush and FetchPickAndPlace, which require precise contact dynamics and gripper control. This suggests that the adaptive convergence mechanism may be more beneficial when the action space has more direct geometric effects on the environment state. A detailed analysis of how the adaptive mechanism affects training duration and convergence behavior across different hyperparameter configurations is provided in Appendix B.



(a) FetchReach Results

(b) FetchPush Results

(c) FetchSlide Results

(d) FetchPickAndPlace Results

Figure 2: Performance comparison across four Fetch environments.

5.1 Decoupled Fixed-Schedule Training

The Decoupled (Fixed) configuration, where the CVAE is pre-trained on the distilled dataset $\mathcal{D}'$ for a fixed number of steps, shows mixed results compared to the baseline (Figure 2). While a marginal improvement is observed on FetchPickAndPlace (+0.99%), performance degrades on FetchPush (−9.67%), FetchReach (−0.98%), and FetchSlide (−1.47%).

Notably, the fixed-schedule approach exhibits reduced stability (higher standard deviation) in three out of four environments, with FetchReach showing a 79.59% increase in variance. This suggests that a fixed pre-training duration, while eliminating redundant CVAE training across policy trials, may not adequately adapt to the characteristics of distilled datasets. The only exception is FetchSlide, where stability improves by 12.99%.

These results indicate that simply decoupling the CVAE training without adaptive termination is insufficient to consistently improve performance, motivating the need for data-dependent convergence criteria.

5.2 Adaptive Warmup: Threshold and Window Size Analysis

The adaptive warmup mechanism demonstrates performance gains over both the baseline and the fixed-schedule approach, with effectiveness depending on the convergence threshold δ and window size W.

- Adaptive (3%, W=5K): This conservative configuration yields modest improvements on FetchPickAndPlace (+5.94%), Reach

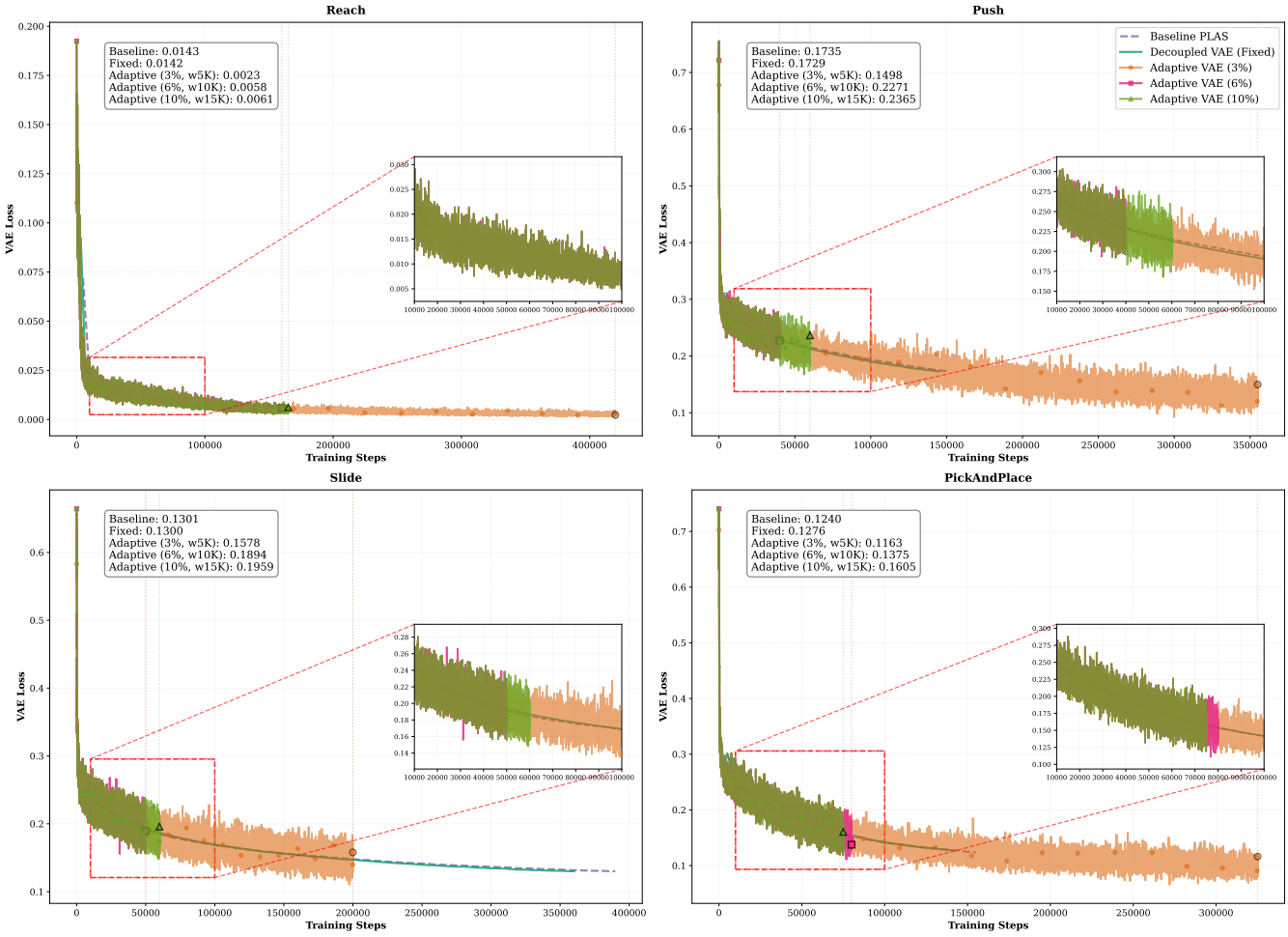


Figure 3: VAE reconstruction loss convergence across adaptive configurations.

(+2.05%), and Slide (+3.24%), while showing slight degradation on Push (−3.63%). Importantly, this configuration improves stability in three environments, with Reach achieving a 37.76% reduction in variance.

- Adaptive (6%, W=10K): This configuration achieves the strongest overall performance, with dramatic improvement on Reach (+21.92%) and Slide (+9.20%). FetchSlide also benefits from 18.36% stability improvement. However, Push exhibits significantly increased variance (−70.19% stability), suggesting task-dependent sensitivity to the convergence threshold.
- Adaptive (10%, W=15K): The most permissive threshold delivers strong gains on FetchReach (+19.58%) and PickAndPlace (+7.84%), with Slide achieving remarkable stability improvement (+48.87%). Performance on Push remains slightly below baseline (−2.90%).

Overall, the adaptive mechanism with $\delta = 6\%$ and $W = 10K$ emerges as the most effective configuration, achieving an average improvement of +8.11% across all environments.

Figure 3 illustrates the VAE reconstruction loss convergence across configurations. A key observation emerges: lower reconstruction loss does not guarantee better policy performance. The Adaptive (3%) configuration achieves the lowest final VAE loss yet does not yield the best downstream results, suggesting that moderate convergence may provide sufficient constraint without overly restricting the latent space.

6 CONCLUSION

We presented PLASDecoupledCVAE, a modular extension to the PLAS algorithm that decouples CVAE pre-training from policy optimization. By refactoring the monolithic training loop into a two-stage pipeline, our framework enables the latent action space to be learned once on high-quality distilled data and subsequently reused across multiple policy training runs without redundant computation.

Key Contributions: Our framework introduces three principal advances:

(1) **Architectural decoupling:** The CVAE warmup phase becomes an independent, serializable stage, eliminating redundant generative model training during hyperparameter optimization.
(2) **Adaptive convergence detection:** A plateau-based early stopping mechanism automatically terminates pre-training when the reconstruction loss stabilizes, replacing fixed-step schedules with data-dependent criteria.
(3) **Hybrid data utilization:** The decoupled architecture enables training the CVAE on curated, high-quality trajectories while the policy learns from the complete dataset, combining representation quality with comprehensive value estimation.

Empirical Findings: Our experiments on the Fetch robotic manipulation suite reveal several insights. First, adaptive warmup can yield performance improvements. Second, optimal configurations are task-dependent: no single threshold dominates across all environments, underscoring the value of exposing convergence criteria as tunable hyperparameters. Third, and perhaps most notably, lower CVAE reconstruction loss does not guarantee better downstream policy performance. This counterintuitive finding highlights the non-trivial relationship between generative model convergence and task success, suggesting that moderate convergence may provide sufficient behavioral constraint without overly restricting the latent action space. |

ACKNOWLEDGMENTS

The authors acknowledge funding by the German Research Foundation (DFG) under SFB 1597 (SmallData), grant number 499552394. The authors also acknowledge support by the state of Baden - Württemberg through bwHPC and the German Research Foundation (DFG) through grant INST 35/1597-1 FUGG. Noor Awad and André Biedenkapp acknowledge funding from The Carl Zeiss Foundation through the research network "Responsive and Scalable Learning for Robots Assisting Humans" (ReScaLe) of the University of Freiburg.

REFERENCES

[1] Rodrigo de Lazcano, Kallinteris Andreas, Jun Jet Tai, Seungjae Ryan Lee, and Jordan Terry. Gymnasium robotics, 2024. URL http://github.com/Farama-Foundation/Gymnasium-Robotics.

[2] Ben Evans, Abitha Thankaraj, and Lerrel Pinto. Context is everything: Implicit identification for dynamics adaptation. In *2022 International Conference on Robotics and Automation, ICRA 2022, Philadelphia, PA, USA, May 23-27, 2022*, pages 2642–2648. IEEE, 2022. doi: 10.1109/ICRA46639.2022.9812119. URL https://doi.org/10.1109/ICRA46639.2022.9812119.

[3] Dylan J. Foster, Akshay Krishnamurthy, David Simchi-Levi, and Yunzong Xu. Offline reinforcement learning: Fundamental barriers for value function approximation. In Po-Ling Loh and Maxim Raginsky, editors, *Conference on Learning Theory, 2-5 July 2022, London, UK*, volume 178 of *Proceedings of Machine Learning Research*, page 3489. PMLR, 2022. URL https://proceedings.mlr.press/v178/foster22a.html.

[4] Scott Fujimoto, David Meger, and Doina Precup. Off-policy deep reinforcement learning without exploration. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pages 2052–2062. PMLR, 2019. URL http://proceedings.mlr.press/v97/fujimoto19a.html.

[5] Aviral Kumar, Justin Fu, Matthew Soh, George Tucker, and Sergey Levine. Stabilizing off-policy q-learning via bootstrapping error reduction. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 11761–11771, 2019. URL https://proceedings.neurips.cc/paper/2019/hash/c2073ffa77b5357a498057413bb09d3a-Abstract.html.

[6] Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *CoRR*, abs/2005.01643, 2020. URL https://arxiv.org/abs/2005.01643.

[7] Tidiane Camaret Ndir, André Biedenkapp, and Noor H. Awad. Inferring behavior-specific context improves zero-shot generalization in reinforcement learning. *CoRR*, abs/2404.09521, 2024. doi: 10.48550/ARXIV.2404.09521. URL https://doi.org/10.48550/arXiv.2404.09521.

[8] Jack Parker-Holder, Raghu Rajan, Xingyou Song, André Biedenkapp, Yingjie Miao, Theresa Eimer, Baohe Zhang, Vu Nguyen, Roberto Calandra, Aleksandra Faust, Frank Hutter, and Marius Lindauer. Automated reinforcement learning (autorl): A survey and open problems. *J. Artif. Intell. Res.*, 74:517–568, 2022. doi: 10.1613/JAIR.1.13596. URL https://doi.org/10.1613/jair.1.13596.

[9] Antonin Raffin, Ashley Hill, Adam Gleave, Anssi Kanervisto, Maximilian Ernestus, and Noah Dormann. Stable-baselines3: Reliable reinforcement learning implementations. *J. Mach. Learn. Res.*, 22:268:1–268:8, 2021. URL https://jmlr.org/papers/v22/20-1364.html.

[10] Kajetan Schweighofer, Marius-Constantin Dinu, Andreas Radler, Markus Hofmarcher, Vihang Prakash Patil, Angela Bitto-Nemling, Hamid Eghbal-zadeh, and Sepp Hochreiter. A dataset perspective on offline reinforcement learning. In Sarath Chandar, Razvan Pascanu, and Doina Precup, editors, *Conference on Lifelong Learning Agents, CoLLAs 2022, 22-24 August 2022, McGill University, Montréal, Québec, Canada*, volume 199 of *Proceedings of Machine Learning Research*, pages 470–517. PMLR, 2022. URL https://proceedings.mlr.press/v199/schweighofer22a.html.

[11] D. Sculley. Web-scale k-means clustering. In Michael Rappa, Paul Jones, Juliana Freire, and Soumen Chakrabarti, editors, *Proceedings of the 19th International Conference on World Wide Web, WWW 2010, Raleigh, North Carolina, USA, April 26-30, 2010*, pages 1177–1178. ACM, 2010. doi: 10.1145/1772690.1772862. URL https://doi.org/10.1145/1772690.1772862.

[12] Takuma Seno and Michita Imai. d3rlpy: An offline deep reinforcement learning library. *J. Mach. Learn. Res.*, 23:315:1–315:20, 2022. URL https://jmlr.org/papers/v23/22-0017.html.

[13] Kihyuk Sohn, Honglak Lee, and Xinchen Yan. Learning structured output representation using deep conditional generative models. In Corinna Cortes, Neil D. Lawrence, Daniel D. Lee, Masashi Sugiyama, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pages 3483–3491, 2015. URL https://proceedings.neurips.cc/paper/2015/hash/8d55a249e6baa5c06772297520da2051-Abstract.html.

[14] Ruosong Wang, Yifan Wu, Ruslan Salakhutdinov, and Sham M. Kakade. Instabilities of offline RL with pre-trained neural representation. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pages 10948–10960. PMLR, 2021. URL http://proceedings.mlr.press/v139/wang21z.html.

[15] Wenxuan Zhou, Lerrel Pinto, and Abhinav Gupta. Environment probing interaction policies. *CoRR*, abs/1907.11740, 2019. URL http://arxiv.org/abs/1907.11740.

[16] Wenxuan Zhou, Sujay Bajracharya, and David Held. PLAS: latent action space for offline reinforcement learning. In Jens Kober, Fabio Ramos, and Claire J. Tomlin, editors, *4th Conference on Robot Learning, CoRL 2020, 16-18 November 2020, Virtual Event / Cambridge, MA, USA*, volume 155 of *Proceedings of Machine Learning Research*, pages 1719–1735. PMLR, 2020. URL https://proceedings.mlr.press/v155/zhou21b.html.

A APPENDIX

A.1 Fetch Environments

This section contains the details of the Fetch robotic environment suite [1] for four robotic manipulation tasks, on which we evaluated our method. These environments simulate a 7-DoF Fetch robotic arm with a two-finger parallel gripper operating in a tabletop setting.

FetchReachDense-v2: The objective is to position the end-effector at randomly generated target locations in 3D space without object interaction or obstacles. It features a 10-dimensional observation space comprising end-effector position/velocity, gripper state/velocity, and goal information, with gripper control having no functional effect.

FetchPushDense-v2 and FetchPickAndPlaceDense-v2: Both utilize a 25-dimensional observation space that includes end-effector position/velocity, object (block) position/velocity/orientation, relative positions/velocities, and gripper state, along with 3D goal representations. In the Push task, the robot must push a block to target positions on the table surface (fixed height, $z = 0.42m$) with the gripper remaining closed. FetchPickAndPlace extends this to full pick-and-place operations, enabling active gripper control to grasp and place the block at targets either on the table or in mid-air.

FetchSlideDense-v2: This maintains the same 25-dimensional observation structure but replaces the block with a puck, which must be hit across a low-friction tabletop to reach targets outside the robot's workspace. This environment is the most challenging due to: (1) targets positioned beyond reachable workspace requiring indirect manipulation through sliding dynamics, (2) high sensitivity of puck trajectory to hitting force and angle on the slippery surface, and (3) the need to master complex contact dynamics and momentum transfer for accurate long-distance puck propulsion. The gripper remains closed throughout this task.

A.2 Dataset Statistics

Table 1 summarizes the dataset sizes.

Environment	Episodes	Total Transitions
FetchReach	100 000	1 332 724
FetchPush	300 000	5 919 751
FetchPickAndPlace	300 000	6 211 966
FetchSlide	500 000	16 082 758

Table 1: Episodes and transitions of each environment

A.3 Implementation Details

Our implementation builds on d3rlpy [12]. All hyperparameters use d3rlpy default values. The key non-default parameters are:

- CVAE warmup steps (Baseline): 50% of the total steps.
- CVAE warmup steps (Fixed): 50% distilled dataset steps.
- Adaptive convergence threshold δ: {0.03, 0.06, 0.10}
- Adaptive window size W: {5,000, 10,000, 15,000}
- Adaptive patience P: 2
- Dataset distillation ratio α: 0.5

Each reported configuration corresponds to a single trained policy checkpoint. For evaluation, each checkpoint was tested over 10 environment seeds, with 10 episodes per seed, and we report mean ± standard deviation across the 10 seed-wise average episode rewards. These statistics therefore reflect evaluation variability for a fixed trained policy rather than variation across independent training seeds. We did not perform a formal statistical significance test.

B ADAPTIVE BUFFER STEP SIZE EVOLUTION

This appendix provides detailed analysis of the adaptive buffer mechanism across different hyperparameter configurations. Figure 4 shows how the CVAE step size evolves during training for each adaptive configuration across all four manipulation tasks.

Key observations from the step size analysis:

- **Reach**: Rapid step size increase due to fast initial learning
- **Push**: Gradual scaling reflecting steady improvement
- **PickAndPlace**: Delayed scaling due to task complexity
- **Slide**: Conservative scaling matching challenging dynamics

The more aggressive configurations (6% and 10% thresholds) achieve significantly faster convergence, with speedups ranging from 2.5× to 8.9× compared to the conservative default configuration.

C SENSITIVITY ANALYSIS FOR HYPERPARAMETER K

To justify the selection of the number of clusters $K = 2000$, we performed a comprehensive sensitivity analysis across four robotic manipulation environments datasets. This analysis evaluates the trade-off between the granularity of the state-action space discretization and the statistical robustness of the resulting clusters.

Methodology

Data Preparation. We generated a total of 20 distinct datasets for this analysis, corresponding to the combination of four environments (Reach, Push, PickAndPlace, Slide) with five clustering granularities ($K \in \{500, 1000, 2000, 4000, 5000\}$). For each configuration, we collected the top 50% of episodes as selected by our proposed method. This ensures that we are evaluating the specific influence of K on the quality of the data subset retrieved by our algorithm.

Evaluation Metrics. We utilized two complementary metrics to assess the quality of these selected subsets. Let $\mathcal{X} = \{(s_i, a_i)\}_{i=1}^{N}$ be the set of state-action pairs in the selected subset, normalized to zero mean and unit variance.

(1) **k-NN Spread (Coverage Metric):** This measures the diversity and coverage of the selected transitions. For each point $x_i \in \mathcal{X}$, we compute the Euclidean distance $d_k(x_i)$ to its k_{nn}-th nearest neighbor (where $k_{nn} = 50$). The metric is defined as the mean of these distances:

$$\text{Spread} = \frac{1}{N} \sum_{i=1}^{N} d_{k_{nn}}(x_i) \tag{10}$$

A higher spread indicates that the selected clusters cover a broader range of distinct behaviors in the state-action space.

(2) **Average Bin Density (Robustness Metric):** This measures the statistical support of the occupied regions. We project the high-dimensional set $\mathcal{X}$ into a 2D manifold using Principal Component Analysis (PCA) to obtain projected points $\mathcal{P}$. We then discretize this 2D space into a grid of bins $\mathcal{B}$. Let $\mathcal{B}_{active}$ be the set of bins containing at least one sample, and $c(b)$ be the count of samples in bin b. The metric is defined as:

$$\text{Density} = \frac{1}{|\mathcal{B}_{active}|} \sum_{b \in \mathcal{B}_{active}} c(b) \tag{11}$$

Higher density implies that the clusters are statistically robust and not merely artifacts of noise or outliers.

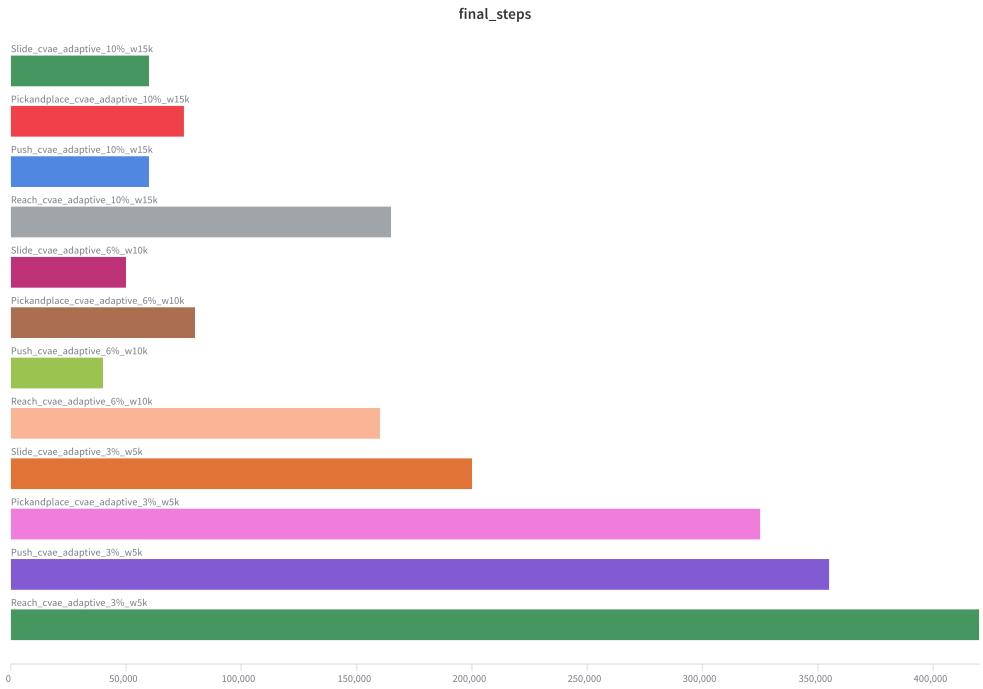


Figure 4: Step size evolution during training for three adaptive buffer configurations across all environments.

Results and Selection

Figure 5 illustrates the relationship between Average Bin Density and k-NN Spread across all environments.

The analysis reveals a consistent trade-off:

- **Low K ($500 - 1000$):** While these settings yield high Average Bin Density, they suffer from lower k-NN Spread. This indicates *aliasing*, where distinct, useful behaviors are merged into single clusters, reducing the resolution of the SACo metric.
- **High K ($4000 - 5000$):** These settings maximize coverage but result in significantly lower bin density. This indicates *over-fragmentation*, where the state-action space is split so finely that clusters become sparse, reducing the statistical reliability of the selection signal.

Conclusion: The value $K = 2000$ consistently occupies the Pareto frontier across the datasets. It maintains a high discriminative resolution (comparable to higher K values) while preserving sufficient sample density to ensure robust selection.

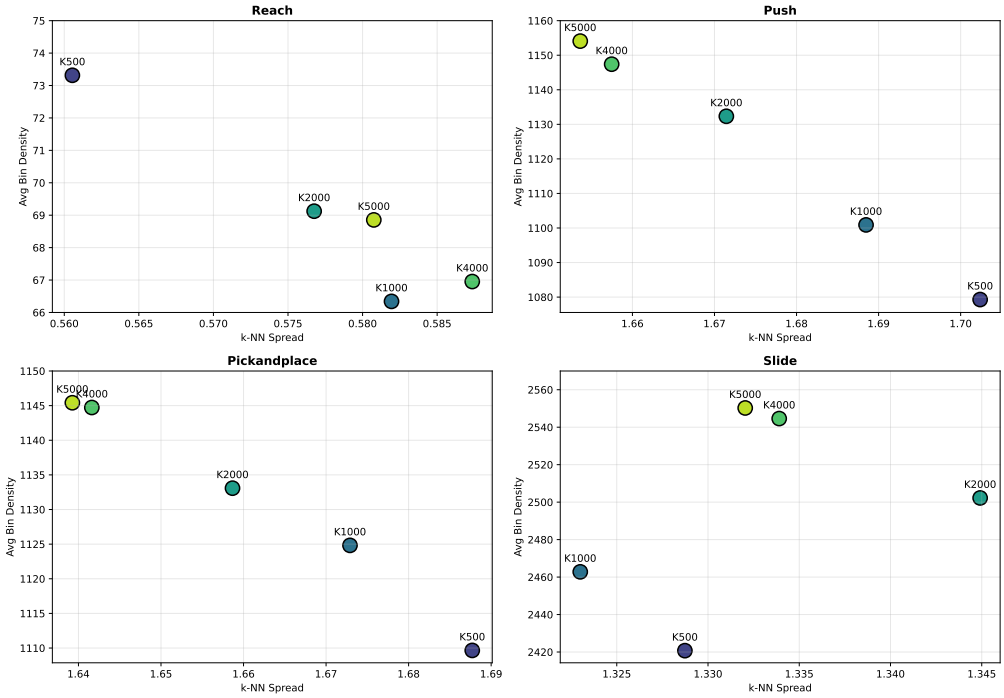


Figure 5: Sensitivity analysis of parameter K.