Policy-Based Trajectory Clustering in Offline Reinforcement Learning

Hao Hu[1] **Xinqi Wang**[2] **Simon S. Du**[3]

[1]Institute for Interdisciplinary Information Sciences, Tsinghua University
[2]Paul G. Allen School of Computer Science & Engineering, University of Washington, Seattle, WA 98195
[3]Paul G. Allen School of Computer Science & Engineering, University of Washington, Seattle, WA 98195

Abstract

We introduce the task of clustering trajectories in offline reinforcement learning (RL) datasets to address the multi-modal nature of offline data. Such datasets often contain trajectories from diverse policies, and treating them as a single distribution can obscure structure and increase distributional shift. We formalize trajectory clustering by linking the KL-divergence of offline trajectory distributions with mixtures of policy-induced distributions. To solve this, we propose Policy-Guided K-means (PG-Kmeans) and Centroid-Attracted Autoencoder (CAAE). PG-Kmeans iteratively trains behavior cloning policies and assigns trajectories based on generation probabilities, while CAAE learns continuous latent representations regularized by a learnable codebook to achieve end-to-end clustering. We prove finite-step convergence of PG-Kmeans and analyze the ambiguity of optimal solutions caused by policy-induced conflicts. Experiments on D4RL and Gridworld show that PG-Kmeans and CAAE partition trajectories into coherent clusters and offer a framework for structuring offline data, with applications in data selection, curriculum learning, and policy transfer.

1 INTRODUCTION

Reinforcement learning (RL) has achieved remarkable progress in diverse domains such as robotic control [Tang et al., 2024], autonomous driving [Kiran et al., 2021], and recommendation systems [Lin et al., 2024]. However, conventional online RL methods require continuous interaction with the environment, which is often impractical due to high costs and safety concerns, especially in sensitive areas such as healthcare and autonomous driving [Dulac-Arnold et al., 2019]. To mitigate these challenges, *offline RL* has emerged as a promising alternative, aiming to learn effective policies from pre-collected datasets without further environment interaction [Levine et al., 2020]. The success of offline RL critically depends on how well the available data are organized and utilized.

Offline RL approaches are commonly divided into two categories. (1) *Value-based methods* constrain policies to remain close to the behavior distribution or adopt conservative value estimation to address distributional shift [Fujimoto et al., 2019, Kumar et al., 2019, Yu et al., 2021, Kumar et al., 2020, Kidambi et al., 2021]. (2) *Behavior cloning methods* optimize policies via supervised learning on offline trajectories or enforce conservative updates to avoid distributional errors [Chen et al., 2020, Brandfonbrener et al., 2021, Kostrikov et al., 2022]. While effective in many cases, both paradigms largely ignore the *structural heterogeneity* of offline datasets—an issue that has increasingly been observed in practice.

Offline datasets are rarely generated by a single policy; instead, they often arise from mixtures of distinct policies with diverse styles. Training on such heterogeneous data can cause mode averaging, interference, and degraded stability. Recent studies provide growing evidence that clustering trajectories is a natural solution, offering several key benefits:

1. **Improved stability and targeted policy learning.** Partitioning trajectories into coherent clusters or modeling them as mixtures improves learning robustness and reduces policy interference [Wang et al., 2024, Osa et al., 2023, Mao et al., 2024]. Beyond simply stabilizing training, clustering also enables *cluster-conditioned policy learning*, where agents selectively exploit high-quality clusters or specialize sub-policies to particular behavioral modes, offering finer control over heterogeneous datasets.
2. **Interpretability and behavioral analysis.** Clustering reveals the latent structure of behavior patterns, helping to diagnose suboptimal strategies and summarize com-

Proceedings of the 42nd Conference on Uncertainty in Artificial Intelligence (UAI 2026), PMLR 337:2196–2222.

plex policies. Prior studies demonstrate that clustering trajectory embeddings [Remman and Lekkas, 2024], applying attention-based abstractions [Bekkemoen and Langseth, 2024], or linking discrete clusters with action attribution [Rishav et al., 2025] can expose distinct behavioral modes. Such analysis not only improves interpretability but also provides a tool for systematically studying how trained agents behave across different contexts.

3. **Data efficiency under limited supervision.** Clustering also supports semi-supervised and sparse-reward scenarios by organizing unlabeled trajectories into meaningful structures. This enables effective learning with limited labels: strong performance can be achieved with as little as 10% reward-labeled data [Zheng et al., 2023], unsupervised skills can be fine-tuned with minimal input [Chebotar et al., 2021], and small sets of preference labels can successfully guide policy optimization when combined with unlabeled data [Kang et al., 2023].

Beyond these points, clustering connects to broader paradigms such as *policy-conditioned agents*, where decision-making is explicitly conditioned on cluster identities, and to modular or hierarchical policy design, which builds on a substantial literature on options and skill discovery [Konidaris and Barto, 2009, Campos et al., 2020, Celik et al., 2024]. Our focus is complementary to this line of work: rather than pre-defining temporal abstractions, we provide an explicit clustering formulation over trajectories.

Despite these promising signs, most prior work clusters or decomposes behavior only implicitly, as a component of a specific algorithm [Mao et al., 2024, Park et al., 2024, Wang et al., 2024]. A formal problem definition, an analysis of its properties (e.g., uniqueness of solutions and conflicts between policies), and dedicated algorithms designed for this setting remain largely missing.

This paper establishes policy-based trajectory clustering as a principled research direction for offline RL. Our main contributions are:

1. We formally define the problem of policy-based trajectory clustering, laying the foundation for systematic study (Section 3).
2. We compare policy-based trajectory clustering with traditional clustering methods, and reveal a fundamental connection to the *K-coloring problem*, highlighting unique challenges absent in classical clustering (Section 4).
3. We propose two new algorithms—Policy-Guided K-means (PG-Kmeans) and Centroid-Attracted Autoencoder (CAAE)—that adopt distinct philosophies: the former explicitly maintains central policies per cluster, while the latter uses an encoder–decoder framework for dataset-level clustering (Section 5).
4. We conduct extensive experiments on the D4RL benchmark and custom environments, showing that our methods achieve superior clustering quality under the Normalized Mutual Information (NMI) metric compared to baseline approaches (Section 6).

2 RELATED WORK

Offline Reinforcement Learning. A central challenge in offline reinforcement learning (RL) is mitigating the distributional shift between the learned policy and the behavior policy from which the dataset was collected. Consistent with the taxonomy in Section 1, existing methods broadly fall into two families. *Value-based methods* address distributional shift by keeping the learned policy close to the behavior policy—explicitly through regularization [Fujimoto et al., 2019, Kumar et al., 2019, Wu et al., 2019] or implicitly via conservative value estimation to prevent reward overestimation [Kumar et al., 2020, Yu et al., 2021]—and often add uncertainty estimation, e.g. ensembles that penalize unreliable actions in out-of-distribution regions [Janner et al., 2019, Kidambi et al., 2021]. *Behavior-cloning methods* either treat behavior cloning as a surrogate for policy learning [Chen et al., 2020] or constrain policy updates to one-step improvements to reduce error accumulation in off-policy evaluation [Brandfonbrener et al., 2021, Kostrikov et al., 2022]. In parallel, alternative strategies such as importance sampling and trajectory reweighting directly optimize over trajectory distributions without explicit distributional correction [Zhang et al., 2020, Nachum et al., 2019, Janner et al., 2021]. Despite their methodological differences, all of these approaches critically rely on the coverage, quality, and structure of the offline dataset.

Policy-Based Clustering. Policy-based clustering has recently emerged as a promising direction for decomposing complex, heterogeneous offline RL datasets into more interpretable and manageable behavior modes. SORL [Mao et al., 2024] and EMPO [Park et al., 2024] adopt expectation-maximization frameworks that alternate between trajectory clustering and policy optimization, enabling the discovery of diverse and high-quality behaviors. Similarly, Wang et al. [2024] propose behavior-aware deep clustering to isolate uni-modal behavioral subsets from multi-behavioral datasets, thereby improving policy stability and performance. Other approaches use probabilistic models for implicit clustering: for instance, Li et al. [2023] employ Gaussian Mixture Models (GMMs) to represent latent behavior policies and derive closed-form policy improvement operators. Diffusion-QL [Wang et al., 2023] leverages expressive diffusion models to capture the multimodal distribution of behavior policies. Collectively, these works highlight the value of policy-level clustering in improving both generalization and robustness in offline RL.

Sequence Models and Skill Discovery. A complementary line models trajectories with sequence models—Decision Transformer [Chen et al., 2021] and Diffuser [Janner et al., 2022]—which capture the multi-modal behavior in offline data implicitly, without producing an explicit dataset partition. Expressive-policy methods such as Diffusion-QL [Wang et al., 2023] similarly represent multi-modal behavior through a flexible generative policy rather than by clustering. Skill- and option-discovery methods instead decompose behavior into reusable temporal abstractions [Konidaris and Barto, 2009, Campos et al., 2020, Celik et al., 2024]. Our setting is distinct in aim: we seek an explicit, policy-level clustering of whole trajectories under a formal objective, rather than an implicit latent code, a generative policy, or a temporal decomposition.

VAEs in Reinforcement Learning. Variational Autoencoders (VAEs) [Kingma and Welling, 2013] are widely used in reinforcement learning for unsupervised representation learning, particularly in high-dimensional or visual domains. Applications include model-based RL [Ha and Schmidhuber, 2018], hierarchical skill discovery [Pertsch et al., 2020], and curiosity-driven exploration [Mohamed and Rezende, 2015, Klissarov et al., 2019]. A common baseline for trajectory modeling is to embed trajectories into latent spaces using VAEs, followed by clustering with standard algorithms such as K-means. VQ-VAE [van den Oord et al., 2018] extends this paradigm by discretizing the latent space via a learned codebook, often improving representation quality and enabling discrete behavior abstraction. However, both continuous VAEs and VQ-VAEs face key limitations: the learned embeddings may not preserve policy-level semantics, and distance-based clustering methods like K-means are poorly aligned with the temporally structured nature of decision-making data (see Table 1). These limitations motivate the design of our CAAE model, which explicitly targets policy-consistent representation learning with clustering objectives in mind.

Deep Clustering. Deep clustering methods aim to jointly learn representations and perform clustering in a unified, often end-to-end, framework. A typical approach involves first learning low-dimensional embeddings with neural networks, then applying clustering algorithms such as K-means or GMMs. DEC [Xie et al., 2016] improves cluster assignments via iterative refinement in an autoencoder architecture, while DEPICT [Dizaji et al., 2017] introduces convolutional backbones for image clustering. DAC [Chang et al., 2017] enforces pairwise similarity constraints to enhance representation learning. More recent advancements explore contrastive learning [Li et al., 2020], graph-based methods that exploit structural relationships [Bo et al., 2020], and fully end-to-end clustering models [Ji et al., 2019]. These advances provide powerful tools for unsupervised data structuring, many of which can be adapted or extended to sequential decision-making data in RL.

3 PROBLEM SETUP: POLICY-BASED TRAJECTORY CLUSTERING

We consider a dataset clustering problem in finite-horizon Markov Decision Process (MDP) offline reinforcement learning (Offline RL), where the objective is to cluster given trajectories based on the policy that generated them.

An MDP is defined as $(\mathcal{S}, \mathcal{A}, P, r, H)$, where $\mathcal{S}$ and $\mathcal{A}$ denote the state and action spaces, respectively. The transition dynamics are given by $P : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$, the reward function is $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$, and H represents the horizon. A trajectory τ is defined as $\{s_t, a_t, r_t\}_{t=1}^{H}$ in the full-reward setting or $\{s_t, a_t\}_{t=1}^{H}$ in the reward-free setting, where $s_t \in \mathcal{S}$ and $a_t \in \mathcal{A}$. A deterministic policy $\pi : \mathcal{S} \times [H] \to \mathcal{A}$ maps each state-timestep pair to an action.

We assume the dataset consists of a mixture of k sub-datasets, each collected under a different deterministic behavior policy $\{\pi_i\}_{i=1}^{k}$. Formally,

$$\mathcal{D} = \bigcup_{i=1}^{k} \mathcal{D}_i,$$

where each $\mathcal{D}_i$ is collected by repeatedly playing π_i. The objective is to cluster the trajectories according to their source. An alternative objective, which casts the task as a K-coloring problem, is provided in Appendix A.2.

For evaluation, we denote ground-truth labels and predicted labels by L and C, respectively, and use Normalized Mutual Information (NMI) [Strehl and Ghosh, 2002] as the performance metric:

$$\text{NMI}(C, L) = \frac{2 \cdot I(C, L)}{H(C) + H(L)},$$

where $I(C, L)$ is the mutual information between C and L, and $H(\cdot)$ denotes Shannon entropy.

4 POLICY-BASED TRAJECTORY CLUSTERING VS. TRADITIONAL CLUSTERING

Nature of Data. Traditional deep clustering techniques are typically designed for static data domains such as images, text, or audio. In these settings, individual samples can be embedded into a high-dimensional feature space, and clustering is performed based on similarity in that space. In contrast, offline reinforcement learning datasets comprise sequential state-action pairs generated by latent policies. Here, similarity is not merely a function of geometric proximity in feature space, but also of the underlying generative process, i.e., the decision-making policy.

Due to the stochasticity and high-dimensionality of RL environments, two trajectories that appear similar in state space may in fact arise from fundamentally different policies. As a result, directly applying conventional clustering methods (e.g., K-means or GMM) to raw or embedded state-action sequences often leads to clusters that reflect environmental or task-level structure rather than true policy-level consistency. This misalignment compromises both interpretability and the utility of the clusters for downstream policy learning.

Geometric vs. Policy-Level Similarity. One might instead cluster trajectories with an off-the-shelf geometric distance on the observation sequence, such as dynamic time warping (DTW) or path-signature features. Such distances quantify how close two trajectories are *as curves in observation space*, which is not the same as whether they were generated by the same policy: two rollouts of a single stochastic policy can be far apart geometrically, whereas two distinct policies can trace nearly identical paths and diverge only at a few decision points. Geometric clustering therefore tends to recover task- or environment-level structure rather than policy identity. Moreover, the natural policy-compatibility "distance" is not even a metric—it violates the triangle inequality (Appendix A.2)—so distance-based clustering lacks its usual guarantees here. We include DTW- and path-signature-based baselines in our experiments (Appendix C.2.3); they succeed only when geometric and policy separability happen to coincide, and fail otherwise.

Ambiguity in Policy-Based Trajectory Clustering: A K–Coloring Perspective Unlike classical clustering tasks, policy-based trajectory clustering may not admit a unique solution. To illustrate this, we draw a reduction to the well-known *K-coloring problem*. Specifically, we construct a graph where each node represents a trajectory, and an edge connects two nodes if the corresponding trajectories exhibit conflicting decision behavior—i.e., they cannot have been generated by the same stationary policy. In this formulation, a valid K-coloring corresponds to a feasible partition of trajectories into policy-consistent clusters.

We formalize this reduction (see Appendix A.2) and show that the policy-based trajectory clustering problem is NP-complete. This theoretical result carries two practical implications. First, unlike standard K-means where the optimal assignment is uniquely determined by minimizing a well-defined distance objective, policy-based clustering may admit multiple equally valid solutions—different partitions can each correspond to a legitimate set of generating policies. Second, there is no efficient algorithm that can guarantee finding the optimal clustering in the worst case, which motivates the use of heuristic approaches such as PG-Kmeans and CAAE. Despite this inherent ambiguity, empirical evidence suggests that stable and semantically meaningful solutions can still be obtained under realistic assumptions on the data distribution (see Appendix A.3).

5 METHODS

To address the problem of policy-based trajectory clustering, we consider two general approaches.

First, we propose Policy-Guided K-means (PG-Kmeans), which explicitly maintains k distinct clusters along with their corresponding policy centroids for clustering. This method operates in the space of behaviors by aligning trajectories with representative policies.

Alternatively, in the direction of representation learning, we introduce the Centroid-Attracted Autoencoder (CAAE), which trains an encoder to map trajectories into a low-dimensional latent space. Clustering is then performed in this latent space based on the distance between the embedded representation and a set of learnable centroids.

5.1 POLICY-GUIDED K-MEANS

The algorithmic foundation of PG-Kmeans is similar to that of the standard K-means clustering algorithm. In K-means-like algorithms, the clustering objective is typically defined as the sum of distances between data points and their respective cluster centers, which is optimized using the Expectation-Maximization (EM) method.

We derive our clustering objective from an information-theoretic perspective, minimizing the *Kullback-Leibler (KL) divergence* between the empirical data distribution $P(\tau)$ and the mixture of policy-induced distributions $\hat{P}(\tau)$, where

$$\hat{P}(\tau) = \sum_{j=1}^{k} w_{i,j} \mathbb{P}(\tau|\theta_j). \tag{1}$$

Here, $\mathbf{W} = \{w_{i,j}\}_{(i,j)=(1,1)}^{(N,k)}$ represents cluster assignments, N is the number of data points, and k is the number of clusters. $w_{i,j}$ is the indicator of trajectory τ_i being assigned to the cluster j. The KL divergence quantifies the discrepancy between these distributions:

$$D_{\mathrm{KL}}(P(\tau)\|\hat{P}(\tau)) = \mathbb{E}_{\tau\sim P}\left[\log \frac{P(\tau)}{\hat{P}(\tau)}\right]. \tag{2}$$

Minimizing this divergence ensures that the learned policies $\{\pi_j\}_{j=1}^{k}$ effectively approximate the empirical data distribution. Since $P(\tau)$ is independent of the optimization variables θ_j and $w_{i,j}$, minimizing KL divergence is equivalent to maximizing:

$$\mathbb{E}_{\tau\sim P}\left[\log \hat{P}(\tau)\right] \approx \sum_{i=1}^{N}\sum_{j=1}^{k} w_{i,j} \log \mathbb{P}(\tau_i|\theta_j). \tag{3}$$

Here we denote τ_i as the sequence $\{(s_{i,h}, a_{i,h})\}_{h=1}^{H}$. Noting that the assignment weights are binary (0 or 1) and that the

environment dynamic is independent of θ_j and $w_{i,j}$, we arrive at the final objective function:

$$\begin{aligned}
&\underset{\theta,\mathbf{W}}{\text{maximize}}\, J(W,\theta) \\
&\quad = \sum_{i=1}^{N}\sum_{j=1}^{k} w_{i,j} \sum_{h=1}^{H} \log \mathbb{P}(a_{i,h}|\theta_j, s_{i,h}), \qquad (4) \\
&\text{s.t.} \quad w_{i,j} \in \{0,1\},\ \forall (i,j) \in [N]\times[k], \\
&\quad \sum_{j=1}^{k} w_{i,j} = 1,\ \forall i \in [N].
\end{aligned}$$

This objective function formulates an optimal distribution matching problem, where we seek to approximate the empirical data distribution using a mixture of policy-induced distributions. The cluster assignments $w_{i,j}$ ensure that each data point is assigned to the most suitable policy, while the policy parameters θ_j are optimized to maximize the likelihood of the assigned trajectories.

5.1.1 Main Algorithm

To solve the clustering objective, we derive an iterative algorithm based on the Expectation-Maximization (EM) framework. See Algorithm 1 for the pseudocode. Similar to K-means, the algorithm alternates between an E-step, where data points are assigned to the most probable generating cluster, and an M-step, where each cluster center is updated to maximize the likelihood of generating its assigned trajectories. This process iterates until convergence, after which cluster centers may be merged to further optimize the objective function $J(W,\theta)$.

The idea of explicitly maintaining k central policies and clustering trajectories based on the likelihood that each central policy generates a target trajectory has also been explored in SORL Mao et al. [2024]. However, it is important to note that SORL employs a continuous assignment matrix $\mathbf{W}$, making it more akin to an EM-style method than to K-means. As a result, SORL is more prone to subpopulation homogenization, a phenomenon also observed in our experiments (see Table 1).

Moreover, due to the use of soft assignments via $\mathbf{W}$, SORL requires training each policy network over the entire dataset. In contrast, PG-Kmeans leverages the classification results from the previous round as a prior approximation, and assigns each data point to a separate network, thereby significantly reducing the overall training cost.

5.1.2 Theoretical Analysis

PG-Kmeans is an instance of *Classification EM* (hard-assignment EM) [Celeux and Govaert, 1992, Bottou and Bengio, 1995]: the E-step assigns each trajectory to the cluster of highest likelihood, $j = \arg\max_{j'} \mathbb{P}(\tau_i \mid \theta_{j'})$, and the M-step re-fits each cluster policy by behavior cloning. Provided the M-step attains at least the previous per-cluster likelihood on its assigned data—which holds under sufficient training or warm-started early stopping—neither step decreases the objective $J(W,\theta)$ of equation 4. Since J is therefore monotone non-decreasing along the iterations and the assignment W takes values in a finite set of partitions, no partition can recur before a fixed point is reached, so the algorithm terminates after finitely many steps.

Algorithm 1 Policy-Guided K-means (PG-Kmeans)

1: **Input:** Dataset $\mathcal{D} = \{\tau_1, \ldots, \tau_N\}$, number of clusters k, ground truth k^* (optional), maximum iterations T.
2: **Initialize** cluster assignments W randomly.
3: **Initialize** k policies $\{\pi_1, \pi_2, \ldots, \pi_k\}$ using behavior cloning (BC) trained on the respective clusters.
4: **for** $t = 1$ to T **do**
5: **M-step:** Update each policy π_j by training a behavior cloning model on the trajectories assigned to cluster j.
6: **E-step:** Assign each trajectory $\tau_i \in \mathcal{D}$ to the cluster $j = \arg\max_{j'} \mathbb{P}(\tau_i|\theta_{j'})$, i.e., $w_i = e_j$, where e_j is a one-hot vector.
7: **Check Convergence:** If cluster assignments W remain unchanged, terminate.
8: **end for**
9: If k^* is given, run Algorithm 4 to merge clusters.
10: **Output:** Final cluster assignments W and policies $\{\pi_1, \pi_2, \ldots, \pi_k\}$.

Theorem 5.1 (Finite-step termination of PG-Kmeans)**.** *Under the Classification-EM updates above, PG-Kmeans reaches a fixed point of the assignment W in at most k^N iterations, at which point $J(W,\theta)$ cannot be improved by any single-trajectory reassignment.*

We emphasize that this is a *termination* guarantee, not a claim of global optimality: exactly as for K-means, the returned partition is only locally optimal with respect to reassignments. The underlying problem is in fact computationally hard—in Appendix A.1 we show that policy-based trajectory clustering is NP-complete via a reduction from K-coloring, which we regard as the distinctive theoretical contribution of this work. The proof of Theorem 5.1 is deferred to the same appendix; although the worst-case bound is $O\left(k^N\right)$, in all our experiments the algorithm converged within 20 iterations.

5.1.3 Implementation details of PG-Kmeans

In practice, the vanilla version of PG-Kmeans exhibited considerable instability during training. To address this, we

adopted a best-of-N selection strategy based on training loss to improve robustness.

In addition, we employed an overparameterization-and-merging approach to handle scenarios where the true number of clusters k is unknown. Empirically, we found that initializing PG-Kmeans with a slightly overestimated number of hypothetical cluster centers leads to substantial performance improvements (see Section 6.2). For further details, please refer to Appendix B.4.

5.2 CENTROID-ATTRACTED AUTOENCODER (CAAE)

In addition to the explicit trajectory clustering approach employed by PG-Kmeans, we introduce an alternative representation learning-based method, the Centroid-Attracted Autoencoder (CAAE). In CAAE, each input trajectory τ is first encoded into a latent representation $z = \text{encoder}(\tau)$, which is then combined with the observation sequence and passed through a decoder to reconstruct the trajectory $\tilde{\tau} \sim \text{decoder}(z, o_{1:H})$.

To regularize the latent space, we impose a constraint motivated by the assumption that each latent variable follows a Gaussian distribution, i.e., $z_i \sim \mathcal{N}(\mu_i, I)$, where μ_i is selected from a learnable codebook $\{\mu_j\}_{j=1}^{k}$. This assumption leads to a penalty term that encourages each z_i to be close to one of the centroids in the codebook, thus promoting structured and interpretable embeddings.

Formally, the CAAE objective is defined as:

$$L(\phi, \theta, \mu) = \sum_{i=1}^{N} \Big(- \sum_{h=1}^{H} \log \mathbb{P}(a_{i,h} \mid \theta, z_i, s_{i,h}) + \min_j \|\mu_j - z_i\|^2 \Big) - \frac{1}{k^2} \sum_{i,j} \min\{1, \|\mu_i - \mu_j\|_2^2\}, \quad (5)$$

where $z_i = \text{encoder}(\tau_i, \phi)$, θ and ϕ represents the parameters of encoder and decoder respectively, and the last term is a regularization term. Implementation details can be found in Appendix B.5 and detailed discussion about the difference between CAAE and VQ-VAE[van den Oord et al., 2018] can be found in Appendix E.

5.3 COMPARISON BETWEEN PG-KMEANS AND CAAE

PG-Kmeans and CAAE represent two complementary design philosophies for policy-based trajectory clustering. We highlight their key differences below.

Architecture and scalability. PG-Kmeans maintains a separate policy network for each cluster, producing sharp cluster boundaries and enabling direct per-cluster policy extraction. However, its computational cost scales linearly with the number of clusters, and each EM iteration requires retraining all cluster policies from scratch on their respective subsets. CAAE uses a single shared encoder–decoder, where the encoder maps an entire trajectory to a latent vector and the decoder reconstructs actions conditioned on both the latent code and individual observations. This parameter sharing makes CAAE more computationally efficient and empirically more stable during optimization.

Granularity of clustering signal. PG-Kmeans assigns trajectories by comparing per-step action likelihoods under each cluster policy, making it sensitive to fine-grained, step-level behavioral differences. This is advantageous when policies diverge at specific decision points. CAAE, by contrast, encodes the trajectory holistically before clustering in latent space, which better captures global behavioral patterns but may be less responsive to localized mode switches within a single trajectory.

Complementarity. These trade-offs suggest that PG-Kmeans is better suited to settings where policies are sharply distinguishable at the action level and the number of clusters is small, while CAAE is preferable when scalability and training stability are priorities, or when behavioral differences manifest at the trajectory level rather than individual steps. We empirically validate these observations in Section 6.2.

6 EXPERIMENTS

We evaluate PG-Kmeans and CAAE in both continuous and discrete action spaces and compare them against several traditional clustering methods.

6.1 ENVIRONMENTS AND BASELINES

Gridworld. We designed three discrete environments and a continuous environment with corresponding policies: Takeball, Diagonal and Extra as discrete environments and PathFollowing as a continuous environment (Figure 2). In the Takeball environment, the agent selects one of four different balls on the map before navigating to the bottom-right corner as the destination. In the Diagonal environment, the agent is randomly initialized in the top-left corner and must reach the bottom-right corner. In the Extra environment, the agent needs to move from a start grid to a terminal grid in a randomly generated map, with two specially marked grids. In PathFollowing, the agent needs to control a ball to be close to a specific point. For each environment, we designed 3-5 different strategies and collected a balanced dataset with them.

To introduce stochasticity, we applied a 0.3 probability of

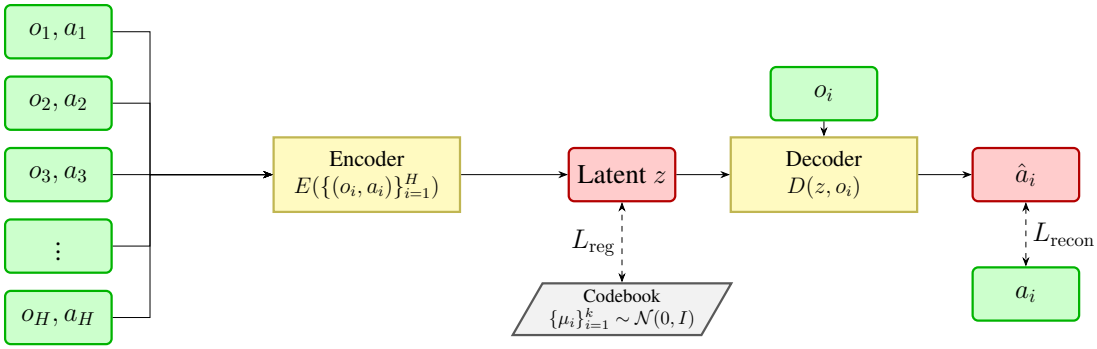


Figure 1: A schematic of our CAAE. The full trajectory $\{(o_i, a_i)\}_{i=1}^H$ is first encoded into a compact latent vector z. A learnable Gaussian codebook $\{\mu_i\}$ imposes a regulation loss to keep z near the prior, while the decoder conditions on both z and a single observation o_i to reconstruct the action $\hat{a}_i$.

random dynamics at each step in discrete environments, and added Gaussian noise to each move in PathFollowing. The initial position was sampled uniformly from $[-1.5, -0.5]^2$ in PathFollowing. We stress that **our clustering methods are reward-free**: PG-Kmeans and CAAE never access any reward signal, so the reward seen by the clustering stage is a constant 0. The shaped rewards described in Appendix B.1 are used *only* to train the PPO experts that generate each dataset; they are never exposed to the clustering algorithms. See Appendix B.1 for details.

Gym Environments. For continuous tasks, we also used the widely adopted D4RL dataset [Fu et al., 2020]. Specifically, we selected the medium-expert datasets from four Gym environments, as they align well with the definition of datasets composed of multiple deterministic policies.

Baseline Methods. To the best of our knowledge, no algorithm has been developed specifically for policy-based clustering prior to this work. Therefore, in our experiments we compare against a selection of deep clustering methods and functionally analogous approaches. Specifically, we use Deep Embedded Clustering[Xie et al., 2016], SORL[Mao et al., 2024], Return + Kmeans, VQ-VAE[van den Oord et al., 2018] and VAE + Kmeans as our baselines.

Evaluation protocol. For PG-Kmeans, we report the best result out of 5 independent runs (selected by training loss) to account for sensitivity to initialization, and sweep the initial cluster count over $\{k^*, k^*+1, k^*+2, k^*+4\}$, where k^* is the ground-truth number of policies. For CAAE and all autoencoder-based baselines, we train for 200 epochs with early stopping based on validation loss. The best-of-N selection uses only the internal objective $J(W, \theta)$ and no ground-truth labels, so it is deployable in practice; when k is unknown we over-parameterize and merge, defaulting to $k = 2\lceil \hat{k} \rceil$ with N=5 restarts. Seed counts vary by method and are reported per cell in Appendix C.2.2; we additionally report the Adjusted Rand Index (Appendix C.2.1), five-seed reruns (Appendix C.2.6), wall-clock cost (Table 6), geometric-distance baselines (Appendix C.2.3), and the sensitivity of our D4RL labels to the split threshold (Appendix C.2.4). Further hyperparameter details are provided in Appendix B.5 and B.4.

6.2 RESULTS AND DISCUSSION

As shown in Table 1, both PG-Kmeans and CAAE achieve consistently high NMI scores across most datasets. PG-Kmeans outperforms SORL in nearly all configurations, highlighting the advantage of explicitly maintaining central policies. Conventional representation learning-based approaches, such as DEC or VAE with K-means, generally fail to provide meaningful clustering.

A noteworthy baseline is VQ-VAE, which already surpasses DEC and in many environments even outperforms SORL by leveraging discrete latent codes. Nevertheless, CAAE achieves a clear margin over VQ-VAE. By learning continuous latent representations rather than relying on codebook projection, CAAE avoids quantization errors and early-stage instability caused by the Straight-Through Estimator, leading to faster convergence and more stable clusters. As illustrated in Figure 7, a t-SNE visualization of the TakeBall environment shows that CAAE produces compact clusters aligned with policy semantics, while VQ-VAE clusters remain fragmented.

We also note that in certain environments, even the best algorithms fail to achieve an NMI close to 1.0. This gap is partly due to optimization but also reflects inherent indistinguishability in the data. For example, in the Diagonal environment, Expert 3 and Expert 4 exhibit nearly identical behavior near the diagonal, making separation difficult. Similarly, in Extra, Experts 2 and 3 sometimes follow almost identical paths on specific maps.

Another important challenge is overfitting (Figure 3). When the conflict rate of (s, a) pairs between two policies is low,

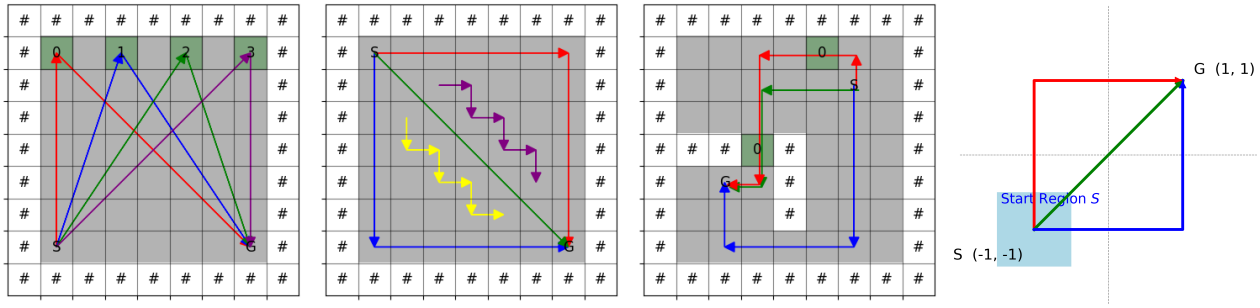


Figure 2: **Takeball (left 1), Diagonal (left 2), Extra (left 3), PathFollowing (left 4).** In Takeball, the four ground-truth policies correspond to tendencies to collect each of the four balls, respectively. In Diagonal, Expert 1 (red) prefers moving right first; Expert 2 (blue) prefers moving down first; Expert 3 (green) follows the diagonal; Experts 4 (purple) and 5 (yellow) follow zigzag paths. In Extra, Expert 1 (red) always visits the two special grids; Expert 2 (blue) never visits them; Expert 3 (green) ignores the special grids. In PathFollowing, each of the three experts first heads toward its own designated waypoint and then follows a shared route to the terminal point.

Table 1: Normalized Mutual Information (NMI) scores for all methods on D4RL and Gridworld environments. The values in the table are presented as mean ± std. VAE and Return represent the algorithms VAE+Kmeans and Return+Kmeans, respectively. Per-(method, task) seed counts are reported in Appendix C.2.2, and the Adjusted Rand Index for these cells is given in Appendix C.2.1. Our algorithms, PG-Kmeans and CAAE, demonstrate consistently high clustering accuracy across the majority of datasets. In contrast, baseline methods such as DEC, SORL, and Return+Kmeans perform well only on a limited subset of datasets.

Task	PG-Kmeans	CAAE	VAE	VQ-VAE	Return	DEC	SORL
HalfCheetah	**0.99** ± 0.00	**0.99** ± 0.00	0.00 ± 0.00	0.96 ± 0.01	0.97 ± 0.00	0.95 ± 0.01	0.12 ± 0.33
Ant	0.92 ± 0.00	**0.96** ± 0.01	0.01 ± 0.01	0.68 ± 0.11	0.05 ± 0.00	0.39 ± 0.17	0.00 ± 0.00
Walker2d	**0.94** ± 0.01	0.88 ± 0.10	0.00 ± 0.00	0.59 ± 0.18	0.23 ± 0.00	0.77 ± 0.12	0.08 ± 0.24
Hopper	**0.99** ± 0.00	**0.99** ± 0.01	0.00 ± 0.00	0.80 ± 0.20	0.86 ± 0.00	0.00 ± 0.00	0.84 ± 0.26
Diagonal	**0.92** ± 0.00	0.87 ± 0.05	0.10 ± 0.02	0.16 ± 0.03	N/A	0.28 ± 0.02	0.18 ± 0.05
Takeball	**1.00** ± 0.00	**1.00** ± 0.00	0.00 ± 0.00	0.57 ± 0.15	N/A	0.05 ± 0.10	0.54 ± 0.13
PathFollowing	0.12 ± 0.11	0.15 ± 0.02	0.03 ± 0.05	0.09 ± 0.04	N/A	**0.26** ± 0.10	0.14 ± 0.05
Extra	0.02 ± 0.01	**0.43** ± 0.22	0.04 ± 0.05	0.15 ± 0.16	N/A	0.28 ± 0.36	0.00 ± 0.00

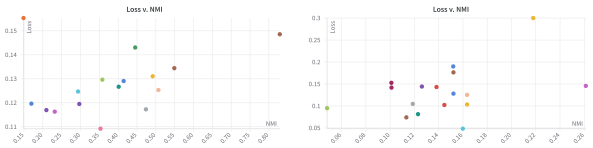


Figure 3: NMI vs. loss in Extra (Left) and PathFollowing (Right) with CAAE. The figure shows no significant correlation between NMI and loss, indicating overfitting.

merging them incurs only minor loss increase. For instance, in PathFollowing, policies conflict mainly in the start region, allowing a single network to approximate all policies with relatively low loss—even if it fails to distinguish them accurately.

PG-Kmeans vs. CAAE. Comparing our two methods reveals complementary strengths consistent with the design analysis in Section 5. PG-Kmeans excels on environments where policies diverge at the per-step action level: it achieves the highest NMI on Diagonal (0.92 vs. 0.87) and Walker2d (0.94 vs. 0.88), where expert behaviors differ in their immediate movement preferences. Conversely, CAAE outperforms PG-Kmeans on tasks that require capturing global trajectory-level structure: on Ant (0.96 vs. 0.92) and Extra (0.43 vs. 0.02), holistic encoding proves more effective than step-wise likelihood comparison. In particular, PG-Kmeans struggles on Extra because the conflicting decisions occur at only a few specific map locations, making per-step likelihood differences negligible across the full trajectory horizon. These results validate the complementarity of the two approaches and suggest that practitioners should choose between them based on whether the target policies are most distinguishable at the step level or the trajectory level.

Downstream utility. Beyond clustering accuracy, the cluster-centroid policies recovered by PG-Kmeans are directly useful downstream. Without any additional RL training, evaluating the non-trivial centroid policies yields normalized D4RL returns of 83/130/104/87

on HalfCheetah/Ant/Walker2d/Hopper—competitive with CQL and AWAC and well above plain behavior cloning (Table 4). Moreover, clustering the dataset before offline RL improves both CQL and IQL on most environments (Table 16); this benefit is *algorithm-dependent*, as advantage-weighted and BC-style methods gain consistently while heavily conservative methods can over-suppress on the smaller per-cluster subsets. Since return-based expert selection is itself fragile under reward noise (Appendix C.2.5), the label-free clustering route is an attractive alternative.

Finally, ablation studies on the choice of cluster number k and the role of regularization (Appendix C.3) confirm that regularization is crucial, and mild overparameterization can further improve performance.

7 CONCLUSION, LIMITATIONS, AND FUTURE WORK

In this paper, we formalized the problem of policy-based trajectory clustering in offline reinforcement learning, established its NP-completeness via a reduction to the K-coloring problem, and analyzed how it fundamentally differs from conventional clustering tasks. To address this challenge, we introduced two complementary methods—PG-Kmeans and CAAE—and evaluated them on D4RL benchmarks and custom Gridworld environments. Experimental results demonstrate their clear and consistent advantages over existing baselines across both continuous and discrete action spaces.

Limitations. Several limitations of this work should be noted. First, the scale of our experiments remains relatively modest; the proposed methods have not yet been tested on large-scale datasets or complex, high-dimensional environments such as vision-based control. Second, our problem formulation assumes deterministic behavior policies and a known (or mildly overestimated) number of clusters k, which may not always hold in practice. Third, although PG-Kmeans enjoys finite-step convergence, CAAE currently lacks formal convergence guarantees, and neither method provides bounds on the quality of the obtained clustering relative to the global optimum.

Future work. Several directions are worth exploring. First, the most immediate application is to leverage the clustering output for downstream offline RL: training separate policies on each cluster and selecting the best one with minimal reward information could provide a practical alternative to filtering-based approaches. Second, extending the framework to handle stochastic or non-stationary behavior policies would broaden its applicability to real-world datasets where policies evolve over time. Third, developing principled methods for automatically determining the number of clusters—for example, via information-theoretic criteria or gap statistics adapted to the policy-clustering objective—would reduce the reliance on prior knowledge of k. Finally, integrating the clustering step into end-to-end offline RL pipelines, where cluster assignments and policy optimization are jointly learned, represents a promising direction toward fully exploiting the structure of heterogeneous offline data.

Author Contributions

H. Hu and X. Wang contributed equally to this work. H. Hu conceived the idea and conducted the experiments. X. Wang developed the theoretical analysis. S. S. Du supervised the project.

Acknowledgements

SSD acknowledges the support of NSF DMS 2134106, NSF CCF 2212261, NSF IIS 2143493, NSF IIS 2229881, Alfred P. Sloan Research Fellowship, and Schmidt Sciences AI 2050 Fellowship.

References

Yanzhe Bekkemoen and Helge Langseth. ASAP: Attention-based state space abstraction for policy summarization. In Berrin Yanıkoğlu and Wray Buntine, editors, *Proceedings of the 15th Asian Conference on Machine Learning*, volume 222 of *Proceedings of Machine Learning Research*, pages 137–152. PMLR, 11–14 Nov 2024. URL https://proceedings.mlr.press/v222/bekkemoen24a.html.

Deyu Bo, Xiao Wang, Chuan Shi, Meiqi Zhu, Emiao Lu, and Peng Cui. Structural deep clustering network. In *Proceedings of The Web Conference 2020*, WWW '20, page 1400–1410. ACM, April 2020. doi: 10.1145/3366423.3380214. URL http://dx.doi.org/10.1145/3366423.3380214.

Léon Bottou and Yoshua Bengio. Convergence properties of the k-means algorithms. In *Advances in Neural Information Processing Systems*, volume 7, pages 585–592. MIT Press, 1995.

David Brandfonbrener, William F. Whitney, Rajesh Ranganath, and Joan Bruna. Offline rl without off-policy evaluation, 2021. URL https://arxiv.org/abs/2106.08909.

Víctor Campos, Alexander Trott, Caiming Xiong, Richard Socher, Xavier Giró-i Nieto, and Jordi Torres. Explore, discover and learn: Unsupervised discovery of state-covering skills. In *International Conference on Machine Learning (ICML)*, 2020.

Gilles Celeux and Gérard Govaert. A classification EM algorithm for clustering and two stochastic versions. *Computational Statistics & Data Analysis*, 14(3):315–332, 1992.

Onur Celik, Aleksandar Taranovic, and Gerhard Neumann. Acquiring diverse skills using curriculum reinforcement learning with mixture of experts. In *International Conference on Machine Learning (ICML)*, 2024.

Jianlong Chang, Lingfeng Wang, Gaofeng Meng, Shiming Xiang, and Chunhong Pan. Deep Adaptive Image Clustering . In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 5880–5888, Los Alamitos, CA, USA, October 2017. IEEE Computer Society. doi: 10.1109/ICCV.2017.626. URL https://doi.ieeecomputersociety.org/10.1109/ICCV.2017.626.

Yevgen Chebotar, Karol Hausman, Yao Lu, Ted Xiao, Dmitry Kalashnikov, Jake Varley, Alex Irpan, Benjamin Eysenbach, Ryan Julian, Chelsea Finn, and Sergey Levine. Actionable models: Unsupervised offline reinforcement learning of robotic skills, 2021. URL https://arxiv.org/abs/2104.07749.

Lili Chen, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Michael Laskin, Pieter Abbeel, Aravind Srinivas, and Igor Mordatch. Decision transformer: Reinforcement learning via sequence modeling, 2021. URL https://arxiv.org/abs/2106.01345.

Xinyue Chen, Zijian Zhou, Zheng Wang, Che Wang, Yanqiu Wu, and Keith Ross. Bail: Best-action imitation learning for batch deep reinforcement learning, 2020. URL https://arxiv.org/abs/1910.12179.

Kamran Ghasedi Dizaji, Amirhossein Herandi, Cheng Deng, Weidong Cai, and Heng Huang. Deep clustering via joint convolutional autoencoder embedding and relative entropy minimization, 2017. URL https://arxiv.org/abs/1704.06327.

Gabriel Dulac-Arnold, Daniel Mankowitz, and Todd Hester. Challenges of real-world reinforcement learning, 2019. URL https://arxiv.org/abs/1904.12901.

Justin Fu, Aviral Kumar, Ofir Nachum, George Tucker, and Sergey Levine. D4RL: datasets for deep data-driven reinforcement learning. *CoRR*, abs/2004.07219, 2020. URL https://arxiv.org/abs/2004.07219.

Scott Fujimoto, David Meger, and Doina Precup. Off-policy deep reinforcement learning without exploration, 2019. URL https://arxiv.org/abs/1812.02900.

David Ha and Jürgen Schmidhuber. World models. 2018. doi: 10.5281/ZENODO.1207631. URL https://zenodo.org/record/1207631.

Michael Janner, Justin Fu, Marvin Zhang, and Sergey Levine. When to trust your model: Model-based policy optimization. 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 12498–12509, 2019. URL https://proceedings.neurips.cc/paper/2019/hash/5faf461eff3099671ad63c6f3f094f7f-Abstract.html.

Michael Janner, Qiyang Li, and Sergey Levine. Offline reinforcement learning as one big sequence modeling problem, 2021. URL https://arxiv.org/abs/2106.02039.

Michael Janner, Yilun Du, Joshua B. Tenenbaum, and Sergey Levine. Planning with diffusion for flexible behavior synthesis. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvári, Gang Niu, and Sivan Sabato, editors, *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*, volume 162 of *Proceedings of Machine Learning Research*, pages 9902–9915. PMLR, 2022. URL https://proceedings.mlr.press/v162/janner22a.html.

Xu Ji, João F. Henriques, and Andrea Vedaldi. Invariant information clustering for unsupervised image classification and segmentation, 2019. URL https://arxiv.org/abs/1807.06653.

Yachen Kang, Diyuan Shi, Jinxin Liu, Li He, and Donglin Wang. Beyond reward: Offline preference-guided policy optimization. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, editors, *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 15753–15768. PMLR, 23–29 Jul 2023. URL https://proceedings.mlr.press/v202/kang23b.html.

Rahul Kidambi, Aravind Rajeswaran, Praneeth Netrapalli, and Thorsten Joachims. Morel : Model-based offline reinforcement learning, 2021. URL https://arxiv.org/abs/2005.05951.

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

B Ravi Kiran, Ibrahim Sobh, Victor Talpaert, Patrick Mannion, Ahmad A. Al Sallab, Senthil Yogamani, and Patrick Pérez. Deep reinforcement learning for autonomous driving: A survey, 2021. URL https://arxiv.org/abs/2002.00444.

Martin Klissarov, Pierre-Luc Bacon, Joelle Pineau, and Doina Precup. Variational state encoding as intrinsic motivation in reinforcement learning. In *Task-Agnostic Reinforcement Learning Workshop at Proceedings of the International Conference on Learning Representations (ICLR)*, volume 15, 2019.

George Konidaris and Andrew Barto. Skill discovery in continuous reinforcement learning domains using skill chaining. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 22, 2009.

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

Aviral Kumar, Justin Fu, George Tucker, and Sergey Levine. Stabilizing off-policy q-learning via bootstrapping error reduction, 2019. URL https://arxiv.org/abs/1906.00949.

Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative q-learning for offline reinforcement learning. *CoRR*, abs/2006.04779, 2020. URL https://arxiv.org/abs/2006.04779.

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.

Jiachen Li, Edwin Zhang, Ming Yin, Qinxun Bai, Yu-Xiang Wang, and William Yang Wang. Offline reinforcement learning with closed-form policy improvement operators, 2023. URL https://arxiv.org/abs/2211.15956.

Yunfan Li, Peng Hu, Zitao Liu, Dezhong Peng, Joey Tianyi Zhou, and Xi Peng. Contrastive clustering, 2020. URL https://arxiv.org/abs/2009.09687.

Yuanguo Lin, Yong Liu, Fan Lin, Lixin Zou, Pengcheng Wu, Wenhua Zeng, Huanhuan Chen, and Chunyan Miao. A survey on reinforcement learning for recommender systems. *IEEE Transactions on Neural Networks and Learning Systems*, 35(10):13164–13184, October 2024. ISSN 2162-2388. doi: 10.1109/tnnls.2023.3280161. URL http://dx.doi.org/10.1109/TNNLS.2023.3280161.

Yihuan Mao, Chengjie Wu, Xi Chen, Hao Hu, Ji Jiang, Tianze Zhou, Tangjie Lv, Changjie Fan, Zhipeng Hu, Yi Wu, Yujing Hu, and Chongjie Zhang. Stylized offline reinforcement learning: Extracting diverse high-quality behaviors from heterogeneous datasets. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=rnHNDihrIT.

Shakir Mohamed and Danilo Jimenez Rezende. Variational information maximisation for intrinsically motivated reinforcement learning. In *Advances in neural information processing systems*, pages 2125–2133, 2015.

Ofir Nachum, Bo Dai, Ilya Kostrikov, Yinlam Chow, Lihong Li, and Dale Schuurmans. Algaedice: Policy gradient from arbitrary experience, 2019. URL https://arxiv.org/abs/1912.02074.

Takayuki Osa, Akinobu Hayashi, Pranav Deo, Naoki Morihira, and Takahide Yoshiike. Offline reinforcement learning with mixture of deterministic policies. *Transactions on Machine Learning Research*, 2023. ISSN 2835-8856. URL https://openreview.net/forum?id=zkRCp4RmAF.

Jongeui Park, Myungsik Cho, and Youngchul Sung. EMPO: A clustering-based on-policy algorithm for offline reinforcement learing. In *ICML 2024 Workshop: Aligning Reinforcement Learning Experimentalists and Theorists*, 2024. URL https://openreview.net/forum?id=Vga4Kz3rEj.

Karl Pertsch, Youngwoon Lee, and Joseph J. Lim. Accelerating reinforcement learning with learned skill priors, 2020. URL https://arxiv.org/abs/2010.11944.

Sindre Benjamin Remman and Anastasios M. Lekkas. Discovering behavioral modes in deep reinforcement learning policies using trajectory clustering in latent space, 2024. URL https://arxiv.org/abs/2402.12939.

Rishav Rishav, Somjit Nath, Vincent Michalski, and Samira Ebrahimi Kahou. Behaviour discovery and attribution for explainable reinforcement learning, 2025. URL https://arxiv.org/abs/2503.14973.

Alexander Strehl and Joydeep Ghosh. Cluster ensembles—a knowledge reuse framework for combining multiple partitions. *Journal of Machine Learning Research*, 3(Dec): 583–617, 2002.

Chen Tang, Ben Abbatematteo, Jiaheng Hu, Rohan Chandra, Roberto Martín-Martín, and Peter Stone. Deep reinforcement learning for robotics: A survey of real-world successes, 2024. URL https://arxiv.org/abs/2408.03539.

Denis Tarasov, Alexander Nikulin, Dmitry Akimov, Vladislav Kurenkov, and Sergey Kolesnikov. CORL: Research-oriented deep offline reinforcement learning library. In *3rd Offline RL Workshop: Offline RL as a "Launchpad"*, 2022. URL https://openreview.net/forum?id=SyAS49bBcv.

Aaron van den Oord, Oriol Vinyals, and Koray Kavukcuoglu. Neural discrete representation learning, 2018. URL https://arxiv.org/abs/1711.00937.

Qiang Wang, Yixin Deng, Francisco Roldan Sanchez, Keru Wang, Kevin McGuinness, Noel O'Connor, and Stephen J. Redmond. Dataset clustering for improved offline policy learning, 2024. URL https://arxiv.org/abs/2402.09550.

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

Yifan Wu, George Tucker, and Ofir Nachum. Behavior regularized offline reinforcement learning, 2019. URL https://arxiv.org/abs/1911.11361.

Junyuan Xie, Ross Girshick, and Ali Farhadi. Unsupervised deep embedding for clustering analysis, 2016. URL https://arxiv.org/abs/1511.06335.

Tianhe Yu, Aviral Kumar, Rafael Rafailov, Aravind Rajeswaran, Sergey Levine, and Chelsea Finn. Combo: Conservative offline model-based policy optimization. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 28954–28967. Curran Associates, Inc., 2021.

Ruiyi Zhang, Bo Dai, Lihong Li, and Dale Schuurmans. Gendice: Generalized offline estimation of stationary values. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. URL https://openreview.net/forum?id=HkxlcnVFwB.

Qinqing Zheng, Mikael Henaff, Brandon Amos, and Aditya Grover. Semi-supervised offline reinforcement learning with action-free trajectories, 2023. URL https://arxiv.org/abs/2210.06518.

Policy-Based Trajectory Clustering in Offline Reinforcement Learning (Supplementary Material)

Hao Hu[1] **Xinqi Wang**[2] **Simon S. Du**[3]

[1]Institute for Interdisciplinary Information Sciences, Tsinghua University
[2]Paul G. Allen School of Computer Science & Engineering, University of Washington, Seattle, WA 98195
[3]Paul G. Allen School of Computer Science & Engineering, University of Washington, Seattle, WA 98195

CONTENTS

A APPENDIX: MATHEMATICAL DISCUSSION

A.1 PROOF OF THEOREM 5.1

Theorem A.1 (Finite convergence of Algorithm 1)**.** *Algorithm 1 is guaranteed to converge in $O(k^N)$ iterations.*

Proof. We denote the original assignment matrix W as W^0, policy parameters as θ^0. After iteration t, the assignment matrix and parameters become W^t and θ^t. Assume that the policy converges at after $T \in \mathbb{Z}^+ \cup \{\infty\}$ iterations. From the definition we see that $\forall t = 1, 2, \cdots T$,

$$\max_{\theta} J(W^{t-1}, \theta) = J(W^{t-1}, \theta^t) \tag{6}$$

$$< \max_{W} J(W, \theta^t) \tag{7}$$

$$= J(W^t, \theta^t) \tag{8}$$

$$\leq \max_{\theta} J(W^t, \theta). \tag{9}$$

Therefore $\{\max_\theta J(W^{t-1}, \theta)\}_{t=1}^{T}$ is a strictly increasing sequence. Because there are at most k^N different valid values for W, the sequence length T is no larger than k^N, which concludes the proof. □

A.2 POLICY-BASED TRAJECTORY CLUSTER AND K-COLORING

One fundamental reason why K-means clustering cannot be directly applied to policy-based trajectory clustering is the absence of a well-defined distance metric in trajectory space. A natural idea is to define the "distance" between two trajectories based on their compatibility, i.e., whether they could have been generated by the same policy:

$$d(\tau_1, \tau_2) = \begin{cases} 1, & \exists (s, a) \in \tau_1, (s, a') \in \tau_2, a \neq a' \\ 0, & \text{otherwise} \end{cases} \tag{10}$$

However, this definition does not satisfy the triangle inequality, i.e.,

$$|d(x, y)| \leq |d(x, z)| + |d(z, y)| \tag{11}$$

which means it is not a proper metric and provides limited mathematical utility.

For example, consider the trajectories $x = [(s_1, a_1)]$, $y = [(s_1, a_2)]$, and $z = [(s_2, a_1)]$. In this case, we observe that:

$$|d(x, y)| = 1 \not\leq |d(x, z)| + |d(z, y)| = 0. \tag{12}$$

We also note that, in the absence of additional assumptions, Equation (10) encapsulates all the reliable information available in the dataset. That is, any clustering scheme W that satisfies the condition that the total intra-cluster distance is zero provides a valid solution:

$$D(W) = \sum_{k=1}^{K} \sum_{i=1}^{N} \sum_{j=1}^{N} w_{i,k} w_{j,k} d(\tau_i, \tau_j) = 0. \tag{13}$$

This formulation precisely aligns with the definition of the K-coloring problem.

Furthermore, we can construct a simple proof for the following theorem:

Theorem A.2 (Reduction from K-coloring to Policy Clustering)**.** *For any K-coloring problem with N nodes and a maximum degree of d, there exists a Markov Decision Process (MDP) with $|\mathcal{S}| \geq 2d$ and $|H| \geq d$, along with a corresponding dataset, such that the dataset can be generated by K distinct policies, and its valid clustering corresponds to a solution of the original K-coloring problem.*

Proof. We can construct a dataset by following steps:

Algorithm 2 Reduction from K-coloring to policy clustering

Input: a graph $G = (V, E), s.t. \max(\text{degree}(v)) = d$
Initialize Initialize a list t_i for each node v_i.
for $i = 1$ to $|V|$ **do**
  for $j = i + 1$ to $|V|$ **do**
    If $(v_i, v_j) \in E$, find the smallest integer l such that $s_l \notin t_i \cup t_j$
    Append t_i with (s_l, a_1), t_j with (s_l, a_2)
  end for
end for
Pad $\{t_i\}_i$ to length H and concatenate them to get trajectories $\{\tau_i\}_i$.
Output: Dataset for clustering $\{\tau_i\}_{i=1}^{|V|}$.

Because $|t_i \cup t_j| \leq \text{degree}(v_i) + \text{degree}(v_j) \leq 2d$, l would never take value over $2d$. And $\max_i(|t_i|) \leq \max(\text{degree}(v)) = d$, so horizon $H > d$ is enough. □

By construction, trajectories τ_i and τ_j conflict (have distance 1) if and only if $(v_i, v_j) \in E$. Hence a partition of $\{\tau_i\}$ into K clusters with zero total intra-cluster distance exists *if and only if* G admits a proper K-coloring, so the reduction preserves solutions in both directions. Since the decision problem is also in NP—given a candidate assignment W, the total intra-cluster distance $D(W)$ built from the pairwise distance equation 10 can be verified in polynomial time—and 3-coloring (already NP-complete for maximum degree $d \geq 2$) reduces to it, the general policy-guided clustering problem is NP-complete.

A.3 POLICY AMBIGUITY

The above reduction from k-coloring to policy-based clustering demonstrates that policy-based clustering may have multiple solutions. However, this is not necessarily unacceptable. As long as the center policies of the clusters remain consistent, different clustering solutions merely arise due to the fact that these policies exhibit identical expressions in certain instances. The more critical issue is that not only can the trajectory clustering solutions be non-unique, but the center policies themselves may also constitute entirely different constructure. Consider the following example:

Let us examine a contextual bandit setting with only two states and two actions. This environment allows for exactly four distinct deterministic policies and four possible trajectories:

- **Policy 1:** $(s_1, a_1), (s_2, a_1)$
- **Policy 2:** $(s_1, a_1), (s_2, a_2)$
- **Policy 3:** $(s_1, a_2), (s_2, a_1)$
- **Policy 4:** $(s_1, a_2), (s_2, a_2)$

Using either Policy 1 and Policy 4, or Policy 2 and Policy 3, we can generate all four possible trajectories. Thus, for any dataset collected from this contextual bandit, there will always exist at least two valid clustering solutions. Furthermore, when the occurrence probabilities of s_1 and s_2 are equal, these two clustering solutions are completely irrelevant with each other.

The experimental results demonstrate that the non-uniqueness of policy combinations is not merely a theoretical possibility but frequently manifests in practical datasets. In the deprecated environment analogous to Takeball, we observe a modified scenario where four balls are randomly permuted across four fixed positions, while the expert policy π_i still maintains the strategy of collecting the i-th numbered ball and returning.

Notably, the neural network rapidly learns a distinct policy categorization approach: instead of differentiating policies by ball numbering, it develops four position-specific policies that collect balls based on their spatial locations. Although both policy combinations generate identical trajectory distributions within the environment, the position-based strategy exhibits significantly simpler implementation requirements. Each agent in this paradigm only needs to learn navigation to a fixed location, avoiding the more complex task of simultaneously identifying specific ball numbers and selecting among four potential destinations.

Through empirical analysis, we find that random initialization almost invariably leads the network to discover the position-based discrimination method. In contrast, achieving number-based trajectory differentiation requires initializations remarkably close to the target clustering configuration (specifically, Normalized Mutual Index (NMI) > 0.6 in our experiments).

This fundamental non-uniqueness in policy combinations constitutes a critical challenge that directly impacts problem solvability, as it introduces substantial ambiguity in policy identification and may lead to suboptimal solutions that fail to capture the intended behavioral semantics.

B APPENDIX: IMPLEMENTATION DETAILS

B.1 DATASET SETTINGS

D4RL We directly use the medium-expert datasets provided by D4RL for four Gym environments. Each dataset consists of 1,000,000 timesteps. The original datasets are not segmented into episodes nor labeled with their generating policies. Therefore, we divide the dataset into episodes based on the ‘Done‘ signal and assign labels accordingly.

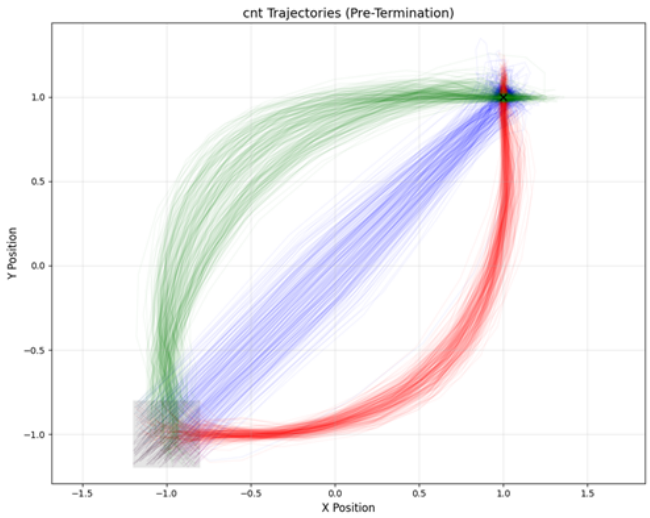


Figure 4: Each color represents one expert policy in PathFollowing.

Since the first half of the dataset corresponds to medium-level data and the latter half to expert-level data, we partition the dataset by maximizing the average return of the first and second halves. However, as it is possible that some of the initial expert episodes achieve relatively low returns, this method may introduce an error of up to five episodes (approximately 5,000 timesteps). Nonetheless, this minor misclassification has a negligible impact on the final NMI and does not affect the quantitative conclusions presented earlier.

Gridworld The Diagonal, Takeball and Extra environments are implemented in JAX. To align with the MDP framework, we use a fixed 9×9 grid map. The action space consists of five discrete actions: movement in four directions and a standby action. The maximum episode length is set to 40 timesteps, and an episode terminates immediately when the agent reaches the goal position. Additionally, to simulate stochastic dynamics, each step has a 0.3 probability of ignoring the action input and taking a random action.

The observation space contains the full state information. In Diagonal, the observation is represented as a 9×9×3 matrix obtained by stacking the wall map, agent map, and goal map. In Takeball, the observation further includes four additional one-hot matrices to encode the positions of four balls, resulting in a 9×9×4 representation. In Extra, the observation further includes two additional one-hot matrices to encode the positions of two special grid, resulting in a 9×9×2 representation.

For Diagonal and Takeball environments, all nine expert policies are rule-based, relying solely on the current state without considering historical information. They are implemented in JAX. For each expert, we collect 20,000 trajectories for evaluation. As a result, the Diagonal dataset consists of a total of 100,000 trajectories, while the Takeball dataset contains 80,000 trajectories. And for Extra environment. We train three policies using PPO algorithm by setting positive/zero/negative rewards for the agent to reach the special grid. Other details are the same as Diagonal and Takeball. The Extra dataset consists of a total of 60,000 trajectories.

B.2 DATASET COLLECTION METHODS IN GRIDWORLD

Diagonal There are five different rule-based policies:

1. Always move right until the wall is reached.
2. Always move down until the wall is reached.
3. Move to the right iff at the left-down half of the gridworld.
4. Move to the right iff at the black grid, if we seem the gridworld as a chess board.
5. Move to the right iff at the white grid.

Takeball There are four different rule-based policies, i-th policy will pick the i-th ball first.

Extra We use PPO algorithm to train three policies by giving positive/zero/negative rewards for the agent to reach the special grid. The rewards are set to +10, 0 or -10, respectively. When the agent reach the goal, the reward is set to +10. The agent will get a penalty of -0.3 for each step and each unit of distance to the goal.

PathFollowing We use PPO algorithm to train three policies by giving extra penalty for the agent when it off path. For each step, the reward is negative distance to the goal add the following penalty for each policy:

1. No extra penalty.
2. 5 times of distance to the polyline $(-1,-1),(-1,1),(1,1)$.
3. 5 times of distance to the polyline $(-1,-1),(1,-1),(1,1)$.

These policies correspond to: no preference, prefer to go up first, and prefer to go right first, respectively. Figure 4 illustrates the trajectories associated with different datasets.

B.3 BASELINE METHODS

Deep Embedded Clustering (DEC). Deep Embedded Clustering (DEC) Xie et al. [2016] is a widely used deep clustering method that integrates representation learning with clustering optimization. It first pretrains an autoencoder to map high-dimensional data into a low-dimensional latent space. Clustering is then performed in this learned space by iteratively refining cluster assignments using a self-training objective. Specifically, DEC employs a Student's t-distribution to measure the similarity between data points and cluster centers and minimizes a Kullback-Leibler (KL) divergence loss to refine embeddings for more compact and well-separated clusters.

In this work, we adapt DEC for policy clustering by modifying its encoding process to focus on capturing policy-related information rather than trajectory-level dynamics. The details are attached in Appendix B.5.

SORL Stylized Offline Reinforcement Learning(SORL) Mao et al. [2024] is a offline reinforcement learning algorithm that designed to learn diverse policies with varied styles. The first step of the algorithm is soft clustering, that is assigning weight to each trajectories on clusters. In this paper, we only adopt the clustering component of SORL, without performing the subsequent reinforcement learning stage. Specifically, we execute lines 1–11 of Algorithm 1 from Mao et al. [2024] and then cluster trajectories according to the posterior distribution $\hat{p}(z \mid \tau)$ as defined in the original paper.

VAE, VQ-VAE VAE learns a continuous latent representation by mapping inputs to a Gaussian distribution and reconstructing them through a probabilistic decoder. This structure encourages smooth latent interpolation and provides a flexible foundation for unsupervised representation learning. Our VAE baseline is constructed by first training a standard VAE model, then applying the K-means algorithm directly to the latent variables obtained from encoding each trajectory.

VQ-VAE incorporates a discrete codebook to represent latent variables, replacing the continuous latent distribution used in CAAE. This quantization mechanism constrains representations to a finite set of embeddings, helping stabilize training and promoting structured latent reuse. Beyond this latent design, the architecture and training configuration remain fully aligned with CAAE for a fair comparison.

To ensure a fair comparison, our VQ-VAE and VAE implementation adopts the same encoder, decoder, reconstruction loss, and all hyperparameters, if applicable, as CAAE. The primary difference between VQ-VAE and CAAE lies in the design and usage of the latent variables. A more detailed analysis of their distinctions can be found in Appendix E.

Returns + Kmeans The *returns+K-means* baseline performs clustering solely based on the return of each trajectory using the K-means algorithm. Consequently, it cannot be applied in scenarios where return information is unavailable. As shown in our results, this method performs poorly in environments with high stochasticity, such as *Ant* and *Walker2d*.

B.4 DETAILS OF PG-KMEANS IMPLEMENTATION

B.4.1 Best-of-N PG-Kmeans

Similar to K-means, PG-Kmeans is highly sensitive to initialization. Moreover, since PG-Kmeans optimizes a neural network for each cluster, small clusters are particularly prone to severe overfitting and non-convex optimization issues, which can

ultimately lead to their disappearance.

This instability makes PG-Kmeans less robust during training. To mitigate this issue, we propose the Best-of-N technique. As shown in Algorithm 3, we use the final objective function $J(W, \theta)$ as an internal metric to evaluate the quality of different runs in the absence of ground truth. The best clustering result is then selected for output.

Algorithm 3 Best-of-N PG-Kmeans

1: **Input:** Input for PG-Kmeans and number of runs N.
2: **for** $t = 1$ to N **do**
3: Run PG-Kmeans (Algorithm 1) and obtain the corresponding output and $J(W, \theta)$.
4: **end for**
5: **Output:** Result from the run with the highest $J(W, \theta)$.

B.4.2 Over-parameterization and Merging

To further improve optimization performance, we introduce the over-parameterization and merging technique. The algorithm faces two primary challenges: (1) center policies often overfit to short and low-density trajectories, causing them to become trapped in incorrect clusters, and (2) with suboptimal initialization, different clusters may be mistakenly merged during clustering. Over-parameterization mitigates these issues by initializing the cluster count k larger than the true number of clusters k^*, enhancing clustering robustness. However, this approach introduces a new challenge: data points from the same cluster may become dispersed, reducing clustering quality. The merging step addresses this issue by consolidating similar clusters, counteracting the adverse effects of a large k while preserving clustering coherence.

For detailed experimental results, see Section 6.2.

Algorithm 4 Merge Clusters

1: **Input:** Datasets and center policies $\{(\mathcal{D}_i, \pi_i)\}_{i=1}^{k}$, target number of clusters k^*.
2: **for** $i = 1$ to $k - k^*$ **do**
3: Find indices $i, j = \arg\min_{i \neq j} \sum_{\tau \in \mathcal{D}_j} \log \mathbb{P}(\tau \mid \theta_i)$.
4: Merge dataset j into dataset i, discard policy π_j. Renumber datasets and policies from 1 to $k - i$.
5: **end for**
6: **Output:** Datasets and center policies $\{(\mathcal{D}_i, \pi_i)\}_{i=1}^{k^*}$.

B.5 DETAILS OF REPRESENTATION LEARNING-BASED ALGORITHM IMPLEMENTATION

In practice, we found that the most important information to distinguish different strategies for a trajectory are given by a contiunous sub-trajectory. Inspired by this observation, we design the encoder model as follows: use a GRU network to encode every prefix and use attention mechanism to get the weighted sum of the outputs of GRU. That is, if the output of GRU is $Y = (y_1, \ldots, y_H)$, we will train three matrix Q, K, V, then the output of attention is:

$$a = \text{softmax}(Q(KY)^\top)(VY)$$

During decoding, we condition on the observation at each timestep along with the embedding to generate the action distribution, using the negative log-likelihood of the true action under this distribution as the reconstruction loss. Under this formulation, the encoder is explicitly designed to capture only policy information, i.e., the action generation mechanism.

To highlight the advantages of CAAE over other representation learning-based algorithms such as DEC and VAE, we employed the same Encoder and Decoder structures used in CAAE for DEC and VAE.

B.6 NETWORK ARCHITECTURES AND TRAINING DETAILS

All networks are trained using the Adam optimizer with a learning rate of 1×10^{-3}. The parameter α of CAAE is set to 1. For the D4RL environments, observations are pre-normalized using statistics from the training set, while no normalization is applied to Gridworld inputs.

Policy Networks. For continuous environments, policies use a `MultivariateNormalDiag` distribution, where two fully connected (FC) layers process extracted features to produce `action_logits` and `action_std`. For discrete environments, policies use a `Categorical` distribution, with a single FC layer mapping extracted features to `action_logits`.

Feature Extractors. The feature extractors for different models are summarized in Table 2. All networks use the ReLU activation function.

Table 2: Network architectures for different models.

Model	Fully Connected Layers	GRU Hidden Size	GRU Output Size
PG-Kmeans	(128, 128)	64	128
VAE/DEC/CAAE-Encoder	(128, 128)	64	128
VAE/DEC/CAAE-Decoder	(128, 32, 32)	N/A	N/A
VAE/DEC/CAAE-Encoder-Attention-Heads		2	
VAE/DEC/CAAE-Encoder-Attention-Feature-Size		2×8	

C APPENDIX: FULL RESULTS AND ABLATION STUDIES

C.1 FULL RESULTS

Here, we provide experimental results for single-run PG-Kmeans and Best-of-5 DEC for comparison with PG-Kmeans. Notably, even when DEC is allowed to enhance its performance by repeatedly running and selecting the best result, it still fails to achieve satisfactory classification in certain environments. This limitation arises because DEC lacks an explicit policy clustering representation, meaning that its embedding-based clustering approach does not guarantee successful categorization in complex environments. Also, as we discussed in Section 6.2, there is no significant negative correlation between loss and NMI for CAAE algorithm, so we didn't do Best-of-5 experiment for CAAE algorithm.

We also evaluated the performance of PG-Kmeans' centroid strategy, the experimental results are presented in Table 15, with classification histograms shown in Figure 6. Across all four environments, PG-Kmeans successfully classified trajectories generated by different policies. When $k > 5$, the accuracy of classification exceeded 50% in all cases. In most cases, the best output policy reaches the performance of agents trained with 10% BC[Tarasov et al., 2022].

C.2 ADDITIONAL METRICS, BASELINES, AND SENSITIVITY ANALYSES

We report several additional analyses requested during review: the Adjusted Rand Index (ARI) alongside NMI, an explicit accounting of per-(method, task) seed counts, geometric-distance baselines (dynamic time warping and path signatures), the sensitivity of our D4RL labels to the medium/expert split threshold, the robustness of return-based expert selection to reward noise, and multi-seed reruns.

C.2.1 Adjusted Rand Index

Table 7 reports ARI for the cells corresponding to the NMI results of Table 1. PG-Kmeans values are the best of five seeds (0–4) selected by the deployable internal objective J (Section 5.1); "n/a" marks baseline–environment combinations without a ground-truth return signal. ARI corroborates the NMI ranking: PG-Kmeans and CAAE attain near-perfect agreement on HalfCheetah, Ant, and Takeball, while the geometric/return baselines collapse on the harder environments.

C.2.2 Per-(method, task) seed counts

Table 8 makes explicit how many seeds underlie each cell of Table 1. PG-Kmeans uses five seeds with best-of-N selection by the internal objective J; CAAE is averaged over 20 seeds and the remaining baselines over 10, with per-seed values for the completed reruns reported in Appendix C.2.6.

Task	Single-run PG-Kmeans	PG-Kmeans	DEC	DEC (best of 5)*
halfcheetah	0.495 ($50\% \in [0.99, 1.00]$)	**0.989** $\pm$ 0.000	0.945 $\pm$ 0.031	**0.989** $\pm$ 0.000
ant	0.745 ($85\% \in [0.83, 0.94]$)	**0.924** $\pm$ 0.003	0.390 $\pm$ 0.512	0.756 $\pm$ 0.003
walker2d	0.557 ($50\% \in [0.70, 0.99]$)	**0.942** $\pm$ 0.005	0.767 ($70\% \in [0.99, 1.00]$)	**0.990** $\pm$ 0.000
hopper	0.258 ($25\% \in [0.99, 1.00]$)	**0.994** ($80\% \in [0.99, 1.00]$)	0.000 $\pm$ 0.000	0.000 $\pm$ 0.000
Diagonal	0.892 $\pm$ 0.032	**0.920** $\pm$ 0.001	0.276 $\pm$ 0.017	0.287 $\pm$ 0.001
Takeball	**0.996** $\pm$ 0.002	**0.997** $\pm$ 0.000	0.054 $\pm$ 0.097	0.175 $\pm$ 0.027

Table 3: Normalized Mutual Information (NMI) scores for all methods on D4RL and Gridworld environments, averaged over at least 10 random seeds. For experiments with clearly distinguishable success or failure outcomes, we report the probability of success along with the NMI range conditioned on successful trials. For all other cases, we assume Gaussian noise and report the 95% confidence interval. *Note: DEC (best of 5) is not a practically feasible algorithm, as ground truth labels are unavailable in real-world clustering scenarios. Without access to true labels, selecting the best result is infeasible. In contrast, PG-Kmeans (best of N) remains a valid approach, as it relies on the final objective function $J(W, \theta)$ as an internal metric to determine the best clustering result for output.

Task	Single-run PG-Kmeans	PG-Kmeans	BC	10%-BC	AWAC	CQL
halfcheetah	56.5	83.4	55.9	90.1	93.6	95.6
ant	76.3	130.4	/	/	/	/
walker2d	44.3	104.5	99.0	108.7	49.4	109.6
hopper	42.48	87.2	52.3	111.2	52.7	99.3

Table 4: Normalized returns of PG-Kmeans and baseline offline RL algorithms on D4RL datasets. Results for other methods are taken from the CORL benchmark Tarasov et al. [2022]. PG-Kmeans is initialized with four cluster centers, and the reported returns are obtained by evaluating the center policies of the two non-trivial (non-zero) clusters in each environment. The results indicate that PG-Kmeans significantly outperforms Behavior Cloning. However, for efficiency reasons, center policies are not fully trained to convergence during optimization. As a result, even with nearly perfect clustering, the learned center policies may not always serve as optimal action generators. Notably, PG-Kmeans operates as a semi-supervised learning method, requiring only a minimal amount of return signals for evaluation, yet achieving performance comparable to fully supervised RL algorithms.

Table 5: NMI of CAAE with different cluster counts k in different environments. The results are averaged over 10 random seeds.

Task	k=4	k=5	k=6	k=7	k=8
Ant	**0.88**	0.87	0.83	0.81	0.79
HalfCheetah	**0.90**	0.85	0.81	0.80	0.75
Hopper	**0.66**	0.55	0.50	0.47	0.46
Walker2d	**0.79**	0.74	0.71	0.69	0.64
Diagonal	N/A	0.82	**0.87**	0.80	0.83
Takeball	**1.00**	0.95	0.90	0.88	0.85
PathFollowing	0.18	0.22	0.27	0.34	**0.35**
Extra	0.38	0.34	**0.43**	0.36	0.37

Table 6: Running time of PG-Kmeans and CAAE. The unit of time is minute in this table. The GPU used is NVIDIA RTX A6000, 48GB.

Algorithm	HalfCheetah	Ant	Walker2d	Hopper	Diagonal	Takeball	PathFollowing	Extra
PG-Kmean	17	24	32	70	22	5.7	24	9.7
CAAE	6.6	10	9.7	15	2.4	1.9	38	2.3

C.2.3 Geometric-distance baselines: DTW and path signatures

As discussed in Section 4, geometric trajectory distances measure proximity of observation curves rather than policy identity. We evaluate two such baselines on D4RL medium-expert: (i) **DTW**: multivariate dynamic time warping (`dtaidistance`) on 200 sub-sampled trajectories per environment, converted to an RBF affinity and clustered with

Table 7: Adjusted Rand Index (ARI) for the cells of Table 1. "n/a" = combination without a ground-truth return signal. PG-Kmeans values are the best of five seeds selected by the internal objective J.

Task	PG-Kmeans	CAAE	VQ-VAE	Return+Km	SORL
HalfCheetah	1.00	1.00	1.00	0.99	0.00
Ant	0.99	0.97	0.61	0.01	0.00
Walker2d	0.94	0.92	0.95	0.28	0.00
Hopper	1.00	1.00	0.00	0.93	0.58
Diagonal	0.76	0.74	0.29	n/a	0.10
Takeball	1.00	1.00	0.88	n/a	0.54
Extra	0.00	0.46	0.49	n/a	0.01

Table 8: Per-(method, task) seed counts underlying Table 1. "B5" = best-of-5 selection by the internal objective J; "n/a" = not applicable.

Method / Task	HC	Ant	Walker	Hopper	Diag	Take	PathF	Extra
PG-Kmeans (seeds $\times$ Best-of-N)	$5\times$B5	$5\times$B5	$5\times$B5	$5\times$B5	$5\times$B5	$5\times$B5	$5\times$B5	$5\times$B5
CAAE (seeds)	20	20	20	20	20	20	20	20
VAE+Km, VQ-VAE, DEC, SORL (seeds)	10	10	10	10	10	10	10	10
Return+Km (`random_state` $\times 10$)	10	10	10	10	n/a	n/a	n/a	n/a

Table 9: Trajectory-geometric baselines on D4RL medium-expert (NMI / ARI).

Task	DTW+Km (NMI / ARI)	PathSig+Km, lvl 3, PCA-8 (NMI / ARI)
HalfCheetah	0.96 / 0.98	0.99 / 1.00
Ant	0.06 / 0.00	0.83 / 0.88
Walker2d	0.96 / 0.98	0.99 / 1.00
Hopper	0.82 / 0.88	0.85 / 0.91

Table 10: PG-Kmeans / CAAE NMI under a D4RL split-threshold shift (uniform-flip upper bound, mean over 20 trials). Each cell is PG-Kmeans / CAAE.

Threshold shift	HalfCheetah	Ant	Hopper
-10%	0.62 / 0.61	0.61 / 0.59	0.61 / 0.61
-5%	0.76 / 0.76	0.75 / 0.73	0.75 / 0.75
0 (paper)	0.99 / 0.99	0.98 / 0.94	1.00 / 1.00
$+5\%$	0.76 / 0.75	0.74 / 0.72	0.73 / 0.73
$+10\%$	0.61 / 0.61	0.60 / 0.58	0.57 / 0.57

`SpectralClustering`(k=2); and (ii) **PathSig**: level-3 path signatures (`iisignature`) over an 8-dimensional PCA of observations, clustered with k-means (best of 10 `random_state`). Table 9 shows they are strong exactly when geometric and policy separability coincide (HalfCheetah, Walker2d) but degrade sharply otherwise (DTW on Ant), confirming that geometric separability is neither necessary nor sufficient for policy separability.

C.2.4 Sensitivity to the D4RL split threshold

Our D4RL labels are constructed by splitting each dataset at the medium/expert boundary (Appendix B.1). To quantify the effect of this heuristic, Table 10 perturbs the split threshold by $\pm 5\%$ and $\pm 10\%$ (a uniform-flip upper bound averaged over 20 trials); predictions are held fixed and only the ground-truth labels are shifted. NMI degrades gracefully and symmetrically, indicating the reported scores are not an artifact of a single threshold choice.

Table 11: Per-trajectory return-noise selector ablation ($\sigma \in \{1, 2, 4\} \times \sigma_{\text{return}}$; means over 20 trials). Top-10% overlap = Jaccard between noisy and true top-10%; expert precision = fraction of noisy top-10% that is truly expert; PG-Km cluster-rank error = $\mathbb{P}$(noisy expert-cluster mean $\leq$ noisy medium-cluster mean).

Env	10%-BC top-10 overlap	10%-BC expert precision	PG-Km cluster-rank error
HalfCheetah	0.22 / 0.18 / 0.15	0.99 / 0.86 / 0.72	0.00 / 0.00 / 0.00
Ant	0.24 / 0.19 / 0.15	0.83 / 0.69 / 0.58	0.00 / 0.00 / 0.00
Walker2d	0.20 / 0.17 / 0.14	0.90 / 0.74 / 0.62	0.00 / 0.00 / 0.00
Hopper	0.33 / 0.24 / 0.17	0.95 / 0.72 / 0.52	0.00 / 0.00 / 0.00

Table 12: Per-seed NMI for the five-seed rerun on D4RL medium-expert. PG-Km rows report best-of-J; other methods report five-seed mean $\pm$ std. "Table 1" = the corresponding value in Table 1.

Method	Env	s_0	s_1	s_2	s_3	s_4	summary	Table 1
PG-Km	HC	0.000	0.994	0.002	0.000	0.012	best-of-J = 0.994	0.99
PG-Km	Ant	0.963	0.978	0.962	0.966	0.964	best-of-J = 0.978	0.92
PG-Km	Walker	0.943	0.912	0.003	0.050	0.892	best-of-J = 0.943	0.94
PG-Km	Hopper	0.980	0.102	0.001	0.932	0.989	best-of-J = 0.989	0.99
CAAE	HC	0.989	0.989	0.989	0.989	0.989	0.989 ± 0.000	0.99
CAAE	Ant	0.942	0.939	0.955	0.947	0.990	0.955 ± 0.021	0.96
CAAE	Walker	0.054	0.990	0.990	0.985	0.995	0.803 ± 0.419	0.88
CAAE	Hopper	0.996	0.992	0.917	0.928	0.958	0.958 ± 0.036	0.99
VQ-VAE	HC	0.989	0.972	0.605	0.657	0.964	0.837 ± 0.189	0.96
VQ-VAE	Ant	0.601	0.525	0.466	0.436	0.479	0.501 ± 0.064	0.68
VQ-VAE	Walker	0.906	0.008	0.368	0.438	0.617	0.467 ± 0.330	0.59
VQ-VAE	Hopper	0.000	0.799	0.803	0.930	0.834	0.673 ± 0.380	0.80
DEC	HC	0.989	0.989	0.994	0.989	0.994	0.991 ± 0.003	0.95
DEC	Ant	0.115	0.012	0.872	0.007	0.124	0.226 ± 0.365	0.39
DEC	Walker	0.742	0.743	0.990	0.990	0.990	0.891 ± 0.136	0.77
DEC	Hopper	0.003	0.000	0.000	0.000	0.000	0.001 ± 0.001	0.00
SORL	HC	0.005	0.088	0.112	0.009	0.074	0.058 ± 0.048	0.12
SORL	Ant	0.003	0.000	0.000	0.000	0.000	0.001 ± 0.001	0.00
SORL	Walker	0.000	0.071	0.000	0.075	0.079	0.045 ± 0.041	0.08
SORL	Hopper	0.001	0.000	0.953	0.951	0.826	0.546 ± 0.501	0.84

C.2.5 Robustness of return-based selection to reward noise

A natural alternative to clustering is to select experts directly by return (e.g., top-10% behavior cloning). Table 11 injects Gaussian noise with $\sigma \in \{1, 2, 4\} \times \sigma_{\text{return}}$ into each trajectory's return (means over 20 trials). As noise grows, the top-10% overlap and expert precision of return-based selection degrade, whereas the PG-Kmeans cluster-mean ranking of expert vs. medium clusters remains error-free, since it relies on policy structure rather than noisy scalar returns.

C.2.6 Multi-seed reruns

We report multi-seed reruns on both D4RL medium-expert and the hardest Gridworld task. Table 12 gives per-seed NMI on D4RL (s_0 = the original single-seed run; s_1–s_4 new): PG-Kmeans rows report best-of-J (the deployable Section 5.1 protocol), while the other methods report five-seed mean $\pm$ std. The high CAAE Walker2d std reflects a single mode-collapsed seed (s_0=0.054; s_1–s_4 are all $\geq$ 0.985), so its median (0.990) is a more robust per-environment summary. PG-Kmeans is likewise bimodal on Walker2d and Hopper—individual seeds land either near 0.9 or near 0—which is precisely why we report the label-free best-of-J selection for PG-Kmeans rather than a raw seed mean. Table 13 additionally gives the full five-seed statistics on the Extra environment; the qualitative ranking again matches Table 1, with CAAE clearly ahead and the step-level and shallow baselines near chance.

Table 13: Extra environment: five-seed NMI and ARI (mean $\pm$ std, no best-of-N selection). "Table 1" = the corresponding NMI in Table 1.

Method	NMI (mean $\pm$ std)	ARI (mean $\pm$ std)	Table 1 (NMI)
PG-Kmeans	0.005 ± 0.004	-0.004 ± 0.006	0.02
CAAE	0.323 ± 0.189	0.229 ± 0.150	0.43
VAE+Km	0.037 ± 0.030	0.031 ± 0.022	0.04
VQ-VAE	0.166 ± 0.254	0.103 ± 0.192	0.15
DEC	0.012 ± 0.006	0.006 ± 0.005	0.28
SORL	0.004 ± 0.002	0.002 ± 0.003	0.00

Table 14: NMI of CAAE with or with out codebook regularization.

Regularization	HalfCheetah	Ant	Walker2d	Hopper	Diagonal	Takeball	PathFollowing	Extra
W/o	0.99	0.96	0.65	0.99	0.84	1.00	0.14	0.39
W/	0.99	0.96	0.88	0.99	0.86	1.00	0.15	0.43

C.2.7 PathFollowing and the role of expert convergence

The PathFollowing results deserve a closer look, because they turn out to be sensitive to how the behavior policies are trained. In the main text (Table 1), the three PathFollowing experts are PPO policies whose early-stage behavior in the shared start region largely overlaps; the induced (s, a) distributions are therefore hard to separate, and both PG-Kmeans and CAAE score low (NMI 0.12 and 0.15, respectively). During the rebuttal period we re-collected the PathFollowing dataset using more thoroughly converged PPO experts, whose distinct route preferences are already expressed from the first few steps. With these better-separated experts, both methods recover the ground-truth clustering almost perfectly (NMI ≈ 1.00 for PG-Kmeans and CAAE).

We deliberately report both outcomes rather than replacing the original. The contrast makes clear that the difficulty on PathFollowing is a property of the specific data-generating policies—how early their behaviors diverge—rather than a fundamental limitation of policy-based clustering. It also highlights a subtlety for benchmark design in this setting: the empirical difficulty of a policy-clustering task depends on the degree of convergence of the behavior policies used to construct the dataset, so under-converged experts can make an otherwise separable task appear unsolvable.

C.3 ABLATION STUDIES

C.3.1 Regularization in CAAE

To mitigate the overfitting issue, we introduced a regularization term to the codebook in CAAE. Without this regularization, for the codebook μ, the last layer of the encoder C and the first layer of the decoder D, and for a real number $0 < \lambda < 1$. We can find the recon loss will not change when $C' = \lambda C, D' = \lambda D$, but the codebook loss will being smaller if $\mu' = \lambda\mu$. This phenomenon will lead the codebook to collapse to a single point, which is not desirable. To prevent this, we add a regularization term to the codebook loss.

We can see from Table 14, regularized CAAE is more robust and achieves better performance in some hard environments. And regularization will not hurt the performance in almost all the environments.

C.3.2 Impact of Initial Cluster Count k.

We further examined the impact of the initial cluster count k on the clustering performance to assess the algorithm's robustness when the true number of categories k^* is unknown or inaccurately estimated. Due to poor performance in the Gridworld environment, this analysis focuses solely on the Gym dataset.

As shown in Figure 5, DEC is highly sensitive to k. On the Gym dataset, increasing k to 10 results in a significant performance drop ($0.8 \rightarrow 0.5$). This degradation primarily stems from the emergence of multiple active cluster centers, which fragment a single true class—an inherent limitation of K-means-based methods. PG-Kmeans exhibits a similar issue but to a lesser extent. On the Gym dataset, it does not naturally disperse, and in the more challenging Gridworld dataset, its

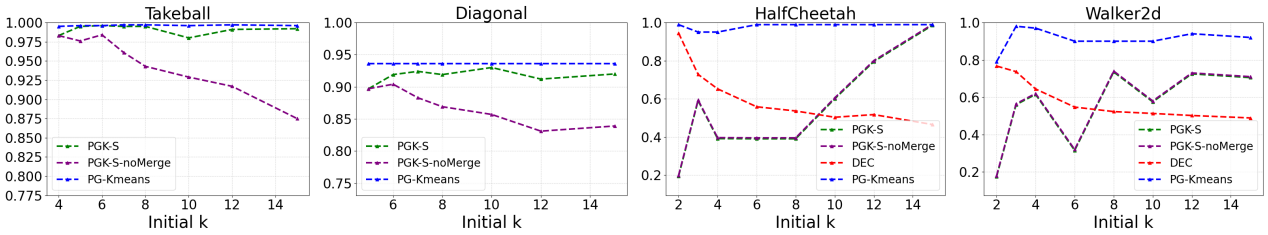


Figure 5: The impact of different initial cluster counts k on the final NMI. All reported values are averaged over 10 random seeds. PGK-S represents the results of single-run PG-Kmeans, while PGK-S-noMerge represents single-run PG-Kmeans without merging. As shown in the figure, increasing the initial cluster count k leads to a noticeable decline in performance metrics for both PG-Kmeans and DEC, with the effect being more pronounced in DEC. This is because a single class is more likely to be split into multiple subclasses that cannot be automatically merged. However, after introducing the merging process, this issue no longer significantly affects PG-Kmeans' performance. Additionally, a higher initial k reduces the probability of cluster fusion, thereby stabilizing PG-Kmeans training. Consequently, in the HalfCheetah and Walker2d environments, increasing the initial k actually improves PG-Kmeans' clustering performance.

NMI decreases by less than 0.1 even when k is increased to three times the ground-truth value. Furthermore, after applying the merging process, PG-Kmeans experiences almost no performance degradation. In fact, when k is slightly larger than k^*, it benefits from reduced overfitting, leading to improved performance.

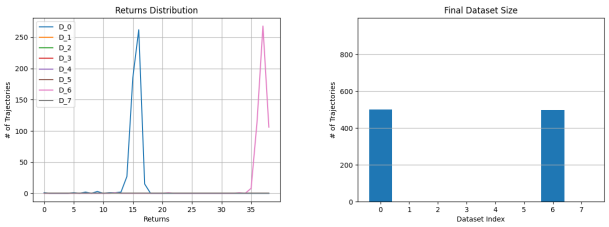
As illustrated in Table 5, CAAE's clustering performance does not exhibit consistent patterns like PG-Kmeans or DEC, but rather demonstrates distinct and divergent trends across different datasets – showing a clear positive correlation in PathFollowing environments, a negative correlation in Walker2d, and only minor fluctuations likely caused by variance in Extra. Generally, more challenging environments benefit more from increasing the k-value, while simpler ones show the opposite trend. We believe that this phenomenon may stem from the aforementioned limitations of the K-means method, which not only compromises model performance but also mitigates the overfitting issue discussed earlier.

Another advantage of increasing k is the enhanced stability of PG-Kmeans, as it reduces the likelihood of mode collapse. Across all tested Gym environments, we observe that the probability of correctly identifying the two policies increases as k grows.

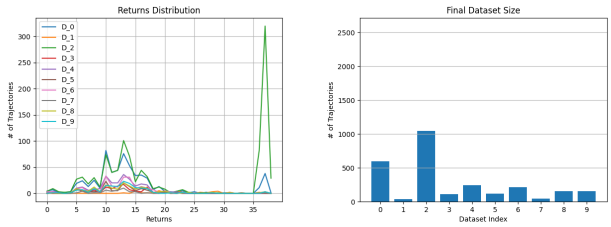
Empirically, for PG-Kmeans algorithm, the optimal choice of k should be slightly larger than k^*, as this strikes a balance between improving stability, reducing overfitting, and minimizing classification accuracy loss.

Table 15: Highest normalized evaluation returns of k output policies, where the expert performance is scaled to 100.0. Every policy is evaluated on 10 different random seeds. Because we stopped the training once all the trajectories have got clear preference over some certain policy, so some networks were not fully trained for evaluation and those results are marked with *. We also highlighted all the results that clearly suffered from mode collapse in red. In these experiments, almost all trajectories were assigned to the same cluster.

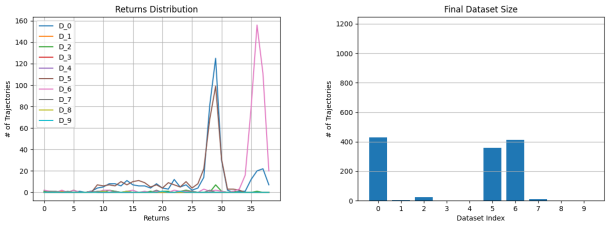
Env \ k	3	4	5	6	7	8	10	12	15
HalfCheetah	40.47	42.13	45.75*	32.58*	69.79	68.47	28.76	88.95	49.59*
Hopper	44.42	45.56	46.47	111.24	80.84*	69.85*	93.44	24.07*	101.33
Ant	22.27	116.91	111.94	61.97	103.52	130.80	47.78*	67.38	67.80
Walker2d	107.44	8.69*	109.84	62.74	103.32	1.29*	98.26	6.33*	-0.09*



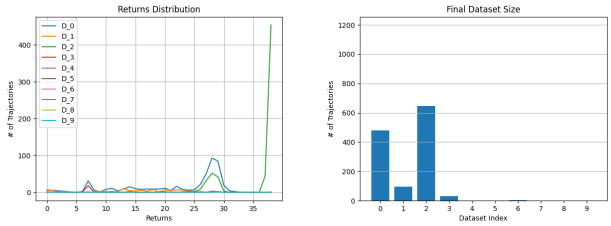
(a) HalfCheetah with $k = 8$



(b) Hopper with $k = 10$



(c) Ant with $k = 10$



(d) Walker2d with $k = 10$

Figure 6: Successful classification samples for different environments and k values. From the figures, it is evident that the dataset is divided into 2–3 primary components. Since the dataset does not come with inherent classification labels, it is not possible to directly calculate classification accuracy. However, we can infer the differences between clusters indirectly by analyzing the return distribution of each cluster.

D ADDITIONAL DISCUSSION: CLUSTERING FOR DOWNSTREAM RL TRAINING

To further investigate the utility of trajectory clustering for downstream policy learning, we conducted experiments on the four D4RL medium-expert datasets (Ant, HalfCheetah, Hopper, and Walker2d). Since the Gridworld dataset does not provide reward signals, it was excluded from this study.

We applied CAAE to cluster trajectories, then sampled 10 trajectories from each cluster to estimate expected returns. CQL and IQL were subsequently trained only on the cluster with the highest average return, denoted as "–clustered." Each result is averaged over 9 random seeds. For PG-Kmeans ($k = 5$), we sampled 10 trajectories per cluster to estimate cluster returns and directly evaluated the centroid policy of the highest-return cluster without additional RL training. Implementations of CQL and IQL followed the JAX-CORL framework.[1]

Table 16: Performance of CQL, IQL, and PG-Kmeans with and without dataset clustering on D4RL medium-expert tasks.

Method + Dataset	CQL-All	CQL-Clustered	IQL-All	IQL-Clustered	PG-Kmeans
Ant	127.11	93.72	122.55	123.14	121.94
HalfCheetah	38.00	48.23	91.83	93.67	69.79
Hopper	1.64	76.98	35.40	105.51	97.33
Walker2d	106.46	79.98	108.37	108.71	107.44

The results indicate that clustering does not universally outperform training on the full dataset, but it can yield substantial improvements in specific problem–algorithm combinations, particularly in Hopper and HalfCheetah with IQL. Two additional observations are worth noting:

- **Minimal reliance on reward signals.** PG-Kmeans achieves competitive performance without any additional RL training, relying only on return samples for cluster selection and centroid evaluation.
- **Clustering as a practical tool.** Although clustering cannot guarantee improvements in every downstream setting, it provides a flexible mechanism to reorganize heterogeneous datasets and selectively exploit high-quality behaviors, thereby enhancing the practical applicability of offline RL.

[1] https://github.com/nissymori/JAX-CORL

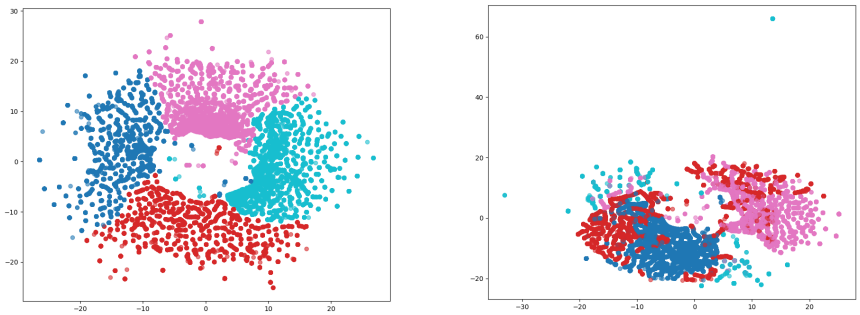


Figure 7: t-SNE visualization of the latent spaces on the TakeBall environment. CAAE (left) produces compact and semantically coherent clusters, while VQ-VAE (right) exhibits fragmented and less stable clusters.

E ADDITIONAL DISCUSSION: CAAE VS. VQ-VAE

In the main text, we reported that VQ-VAE constitutes a strong baseline, already outperforming DEC and in many cases surpassing SORL. However, CAAE consistently achieves superior clustering stability and accuracy. Here we provide a more detailed comparison.

The key differences lie in the latent representation design:

- **Continuous vs. discrete representation.** CAAE directly learns continuous latent vectors, avoiding the discrete codebook projection used in VQ-VAE. This eliminates quantization errors during training.
- **Stability during early training.** VQ-VAE relies on the Straight-Through Estimator, which is known to cause instability in the early stages due to gradient mismatch. CAAE circumvents this issue by optimizing in continuous space.
- **Codebook competition.** In VQ-VAE, multiple codebook vectors may compete for assignment, leading to unstable optimization. CAAE avoids this bottleneck and achieves faster encoder–decoder convergence.
- **Balanced gradient flow.** VQ-VAE sometimes suffers from gradient monopolization by a small subset of codebook entries, which biases representation learning. CAAE distributes gradients more evenly across latent dimensions, leading to more balanced policy-semantic learning.

We further illustrate this comparison in Figure 7, which visualizes clustering on the TakeBall environment via t-SNE. The VQ-VAE representation exhibits noticeable fragmentation, whereas CAAE produces more compact and semantically coherent clusters.

These results confirm that while VQ-VAE is a competitive baseline, the design of CAAE offers distinct advantages for stable and interpretable trajectory clustering.