VALUE-GUIDED IMPLICIT MAXIMUM LIKELIHOOD ESTIMATION FOR OFFLINE REINFORCEMENT LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Offline reinforcement learning aims to learn effective policies from pre-collected datasets without further environment interaction. However, offline datasets often exhibit multimodal behavior distributions, making accurate behavior modeling difficult and increasing the risk of selecting out-of-distribution actions during policy improvement. Recent diffusion- and flow-based policies improve multimodal modeling but often rely on iterative generation, distillation, or auxiliary optimization, leading to substantial computational overhead. To address these issues, we propose Value-guided Implicit Maximum Likelihood Estimation (VIMLE), an efficient offline RL framework that enables multimodal behavior modeling, behavioral consistency filtering, and value-guided action selection. Specifically, VIMLE uses multiple conditional IMLE generators for one-step parallel action generation, filters unreliable candidates through ensemble agreement, and selects high-value actions using an independently learned value function. This decoupled design enables policy improvement over behavior-consistent candidates while avoiding multi-step generation and auxiliary policy training. Extensive evaluations on D4RL demonstrate competitive performance across diverse tasks, while substantially reducing training time and inference latency compared with diffusion-based approaches. Code is provided in the supplementary material.

1 INTRODUCTION

Reinforcement learning (RL) (Sutton, 1988) has achieved remarkable advancements in domains such as robotic control, game playing, and autonomous driving (Kiran et al., 2020). However, conventional online RL requires extensive interactions with the environment for policy optimization, which may incur substantial costs in many practical applications (Levine et al., 2020).

To address this limitation, offline RL learns policies exclusively from pre-collected static datasets without further environment interaction (Levine et al., 2020). However, policy improvement may produce actions rarely observed or absent in the dataset, leading to overestimated OOD actions (Fujimoto et al., 2019). To address the OOD problem, existing methods mainly mitigate this issue through conservative value learning or behavior regularization (Kumar et al., 2020; Kostrikov et al., 2021; Wu et al., 2019). However, most early offline RL methods rely on Gaussian policies, which are often inadequate for modeling the multimodal behavior distributions induced by datasets collected from multiple behavior policies. As shown in Figure 1(b), this limited expressiveness can place probability mass in low-density regions between behavior modes, thereby weakening the effectiveness of behavior regularization and conservative value learning (Wang et al., 2022).

Accordingly, recent offline RL methods have increasingly adopted expressive generative policies, including diffusion and flow-based models (Ho et al., 2020; Lipman et al., 2022), to capture complex multimodal action distributions. Representative methods include diffusion-based approaches such as Diffuser (Janner et al., 2022) and IDQL (Hansen-Estruch et al., 2023), and flow-based approaches such as Flow Q-Learning (Park et al., 2025). As illustrated in Figure 1(c), their greater expressiveness enables better modeling of multimodal behavior data. However, iterative generation introduces considerable computational overhead during training and inference, limiting broader applicability. Recent studies have explored policy extraction and one-step distillation to reduce this

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

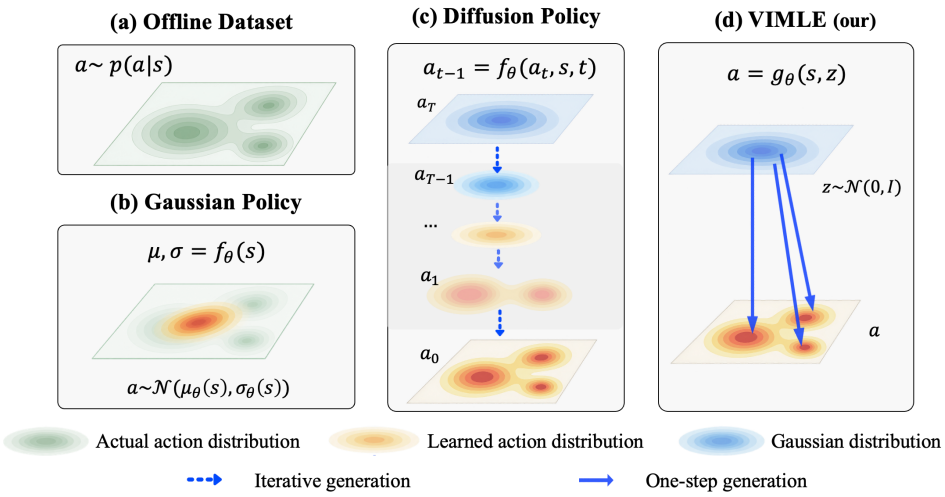


Figure 1: **Multimodal Action Modeling in Offline RL** (a) **Multimodal action distributions** in offline datasets. (b) **Gaussian policies** use unimodal distributions and may average modes. (c) **Diffusion policies** model multimodal actions via iterative denoising, incurring substantial inference latency. (d) **VIMLE** samples latents z in parallel to generate diverse actions in one step.

cost (Park et al., 2025), but such solutions often require additional training stages or model components, increasing overall complexity.

To balance expressive multimodal modeling with computational efficiency, Implicit Maximum Likelihood Estimation (IMLE) (Li & Malik, 2018) provides an alternative. IMLE trains implicit generative models by matching each observed sample to its nearest candidate among generated samples and updating the generator accordingly, encouraging every training sample to be covered by at least one generated output. This sample-wise matching helps alleviate local mode dropping while preserving one-step generation at inference time (Li & Malik, 2018). Compared with generative policies relying on iterative sampling, this formulation can substantially reduce inference latency in closed-loop control. Recent methods, including IMLE Policy (Rana et al., 2025), PRISM (Bhaskar et al., 2026b), and MIRROR (Bhaskar et al., 2026a), have introduced IMLE into behavior cloning and demonstrated multimodal action modeling from limited demonstrations with real-time one-step execution. Nevertheless, these methods primarily focus on reproducing demonstrated behaviors, while integrating IMLE-based behavior generation with offline value learning for value-guided policy improvement over multimodal action candidates remains unexplored.

Our contribution. Building on the above analysis, we propose Value-guided IMLE for Offline Reinforcement Learning (VIMLE), an ensemble-based offline RL framework integrating IMLE-based behavior modeling with value-guided policy improvement. VIMLE enables behavior generation, behavioral consistency filtering, and value learning. Specifically, multiple independent IMLE behavior generators are trained on the offline dataset to model the conditional mapping $a = g_\theta(s, z)$, producing diverse action candidates in parallel through one-step generation, as illustrated in Figure 1(d), while the value function is learned independently from offline data. During inference, VIMLE aggregates candidates from different models, uses ensemble-averaged local agreement to identify behavior-consistent candidates, and performs policy improvement by selecting the highest-value action within the retained set. This design preserves IMLE's multimodal modeling and one-step generation efficiency while constraining value-based selection to behavior-consistent candidates through ensemble agreement, without propagating value gradients into the behavior generators. In contrast to generative offline RL methods that rely on iterative sampling or additional policy distillation, VIMLE avoids multi-step generation and auxiliary policy training, resulting in a more efficient and streamlined learning framework. Extensive experiments across 18 tasks in the MuJoCo and AntMaze domains of D4RL (Fu et al., 2020) demonstrate competitive performance across diverse tasks. Moreover, compared with the diffusion-based Diffuser, VIMLE reduces training time and inference latency by approximately 92.5% and 93.6%, while retaining one-step action generation without additional policy distillation.

2 PRELIMINARIES

2.1 OFFLINE REINFORCEMENT LEARNING

Reinforcement learning (RL) models interactions with the environment as a Markov decision process (MDP) (Puterman, 2014), defined by $\mathcal{M} = (\mathcal{S}, \mathcal{A}, P, r, \gamma)$, where $\mathcal{S}$ and $\mathcal{A}$ denote the state and action spaces, $P(s'|s,a)$ denotes the transition probability, $r(s,a,s')$ is the reward, and $\gamma \in [0,1)$ is the discount factor. A policy $\pi(a|s)$ selects actions conditioned on the current state and, together with the environment dynamics, induces a trajectory distribution. The objective of RL is to learn a policy that maximizes the expected discounted return $J(\pi) = \mathbb{E}_{\pi,P}\left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t, s_{t+1})\right]$. Accordingly, for a given policy π, the action-value function is defined as $Q^\pi(s,a) = \mathbb{E}_{\pi,P}\left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t, s_{t+1}) | s_0 = s, a_0 = a\right]$.

In offline RL, the agent is provided only with a static dataset $\mathcal{D} = \{(s_i, a_i, r_i, s_i')\}_{i=1}^N$, pre-collected by a policy π_β, with no further interaction with the environment during training (Fu et al., 2020). To mitigate unreliable value estimation, many offline RL methods impose behavioral constraints during policy optimization, encouraging the learned policy to remain close to the distribution represented by the offline data (Wu et al., 2019). A general formulation can be written as

$$\pi = \arg\min_{\pi} \mathbb{E}_{s\sim\mathcal{D}}\left[-Q_\theta(s, \pi(s)) + D\left(\pi(\cdot|s) | \pi_\beta(\cdot|s)\right)\right],$$

where Q_θ denotes the value function learned from the offline dataset, π_β denotes the behavior policy that generated the offline dataset, and $D(\cdot|\cdot)$ measures the divergence between the learned and behavior policies. Unlike explicit behavior-regularized methods, VIMLE does not impose an additional distributional constraint. Instead, it models the multimodal behavior distribution with multiple IMLE generators and uses ensemble agreement to identify candidates consistent with the learned generators for subsequent value-guided selection.

2.2 IMPLICIT MAXIMUM LIKELIHOOD ESTIMATION

Implicit Maximum Likelihood Estimation (IMLE) (Li & Malik, 2018) learns a latent-variable generator $g_\theta(z)$ that maps noise samples $z \sim \mathcal{N}(0, I)$ to the data space. Given a dataset $\{x_i\}_{i=1}^N$, the IMLE objective is formulated as

$$\theta^\star = \arg\min_{\theta} \mathbb{E}_{\{z_j\}_{j=1}^m} \sum_{i=1}^{N} \min_{1\le j\le m} \left\| g_\theta(z_j) - x_i \right\|^2.$$

In practice, the optimization is carried out through two alternating steps. For each x_i in a mini-batch $\mathcal{B}$, m candidate latent variables are first sampled, and a nearest-neighbor assignment is performed without gradient propagation:

$$z_i^\star = \arg\min_{1\le j\le m} \left\| g_\theta(z_j) - x_i \right\|^2, \qquad z_j \sim \mathcal{N}(0, I).$$

With $z_i^\star$ fixed, the generator is optimized over the matched pairs by minimizing the empirical loss

$$\mathcal{L}_{\mathrm{IMLE}}(\theta; \mathcal{B}) = \frac{1}{|\mathcal{B}|} \sum_{i\in\mathcal{B}} \left\| g_\theta(z_i^\star) - x_i \right\|^2.$$

Optimizing the IMLE objective is equivalent to maximum likelihood estimation (MLE) under suitable conditions (Li & Malik, 2018), while encouraging mode coverage by ensuring that each data sample is matched by at least one generated sample. This property makes IMLE particularly effective for modeling complex multi-modal distributions.

3 METHOD

In this section, we introduce VIMLE, a value-guided IMLE ensemble framework for offline reinforcement learning. VIMLE consists of three components: conditional IMLE for multimodal behavior modeling, ensemble-based behavioral consistency filtering for removing unreliable action candidates, and an independently learned IQL value function for selecting high-value actions from the retained candidate set. This design enables value-guided policy improvement while reducing the risk of selecting OOD actions. Figure 2 provides an overview of VIMLE.

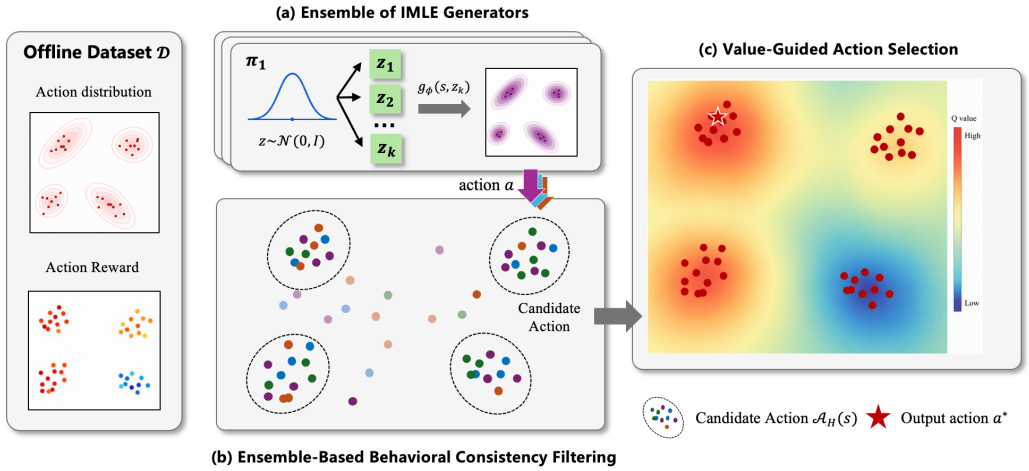


Figure 2: **Overview of VIMLE.** (a) Multiple conditional IMLE generators independently learn the multimodal behavior distribution from the offline dataset and generate diverse action candidates in parallel; (b) ensemble agreement is used to identify behavior-consistent candidates and filter unreliable ones; (c) the retained candidates are evaluated by the value function, and the action with the highest estimated Q-value is selected.

3.1 CONDITIONAL IMLE POLICY MODELING

Offline datasets are often collected from multiple behavior policies or different stages of policy, resulting in multimodal action distributions conditioned on the same state. To model such distributions, we employ conditional IMLE to parameterize the behavior policy. We represent it with a conditional generator g_ϕ:

$$a = g_\phi(s, z), \qquad z \sim \mathcal{N}(0, I) \tag{1}$$

where the latent variable z enables the generator to capture diverse action modes conditioned on the same state. Given a state-action pair (s_i, a_i) from the offline dataset, we first sample K latent variables and obtain a set of candidate actions:

$$a_{i,k} = g_\phi(s_i, z_{i,k}), \qquad z_{i,k} \sim \mathcal{N}(0, I), \quad k = 1, \ldots, K. \tag{2}$$

For each observed action a_i, the generated candidate with the minimum reconstruction distance is then selected as its matched sample:

$$k_i^\star = \arg\min_{1 \le k \le K} \|g_\phi(s_i, z_{i,k}) - a_i\|_2^2 . \tag{3}$$

Given the resulting assignments, the generator is optimized by minimizing the matching error over a mini-batch $\mathcal{B}$:

$$\mathcal{L}_{\text{IMLE}}(\phi) = \frac{1}{|\mathcal{B}|} \sum_{i \in \mathcal{B}} \left\| g_\phi(s_i, z_{i,k_i^\star}) - a_i \right\|_2^2 . \tag{4}$$

Equivalently, the above optimization can be expressed as the following conditional IMLE objective:

$$\mathcal{L}_{\text{IMLE}}(\phi) = \mathbb{E}_{(s,a)\sim\mathcal{D}, z\sim\mathcal{N}(0,I)} \left[\min_{1 \le k \le K} \|g_\phi(s, z_k) - a\|_2^2 \right] .$$

Since the matching operation is performed independently for each observed action, different latent samples can capture distinct behavior modes rather than collapsing into an averaged action, making conditional IMLE suited to offline datasets with multimodal behavior distributions. Moreover, the generator can produce multiple action candidates in parallel through one-step generation, providing an efficient basis for subsequent ensemble-based behavioral consistency filtering.

3.2 ENSEMBLE-BASED BEHAVIORAL CONSISTENCY FILTERING

Although conditional IMLE effectively captures multimodal behavior distributions, a single generator may still produce actions outside the support of the offline data. If selected directly by value

estimates, such candidates may be favored due to estimation errors, increasing OOD risk. To mitigate this issue, we exploit the one-step generation and parallel sampling properties of IMLE to construct an ensemble of M independently trained conditional generators at low computational cost

$$\mathcal{G} = \{g_{\phi_m}\}_{m=1}^{M}.$$

All generators are trained independently on the same offline dataset. During inference, given a state s, we sample candidate actions $\{a_i\}$ from the ensemble in parallel, while each generator additionally produces R reference actions $\{\tilde{a}_{m,r}\}_{r=1}^{R}$. For a candidate action a_i, its consistency with the m-th generator is measured by the nearest-neighbor distance to that generator's reference set, and the distances from all generators are further aggregated into an ensemble consistency score $C(s, a_i)$:

$$d_m(s, a_i) = \min_{1 \le r \le R} \|a_i - \tilde{a}_{m,r}\|_2^2, \qquad C(s, a_i) = \frac{1}{M} \sum_{m=1}^{M} d_m(s, a_i). \tag{5}$$

A candidate with small $d_m(s, a_i)$ across multiple generators yields a smaller $C(s, a_i)$ and is more consistent with the outputs of the independently learned models. We rank the candidate actions according to $C(s, a_i)$ and retain the H candidates with the smallest consistency scores to form the behavior-consistent action set $\mathcal{A}_H(s)$. We provide further theoretical analysis in Appendix D.

3.3 VALUE-GUIDED ACTION SELECTION

The IMLE generators are trained solely to model behavioral actions and therefore provide no direct estimate of long-term return. We thus learn the value function independently from real offline transitions using Implicit Q-Learning (IQL), which does not require interaction with the IMLE policy and naturally decouples behavior modeling from value estimation.

Let $V_\psi(s)$ denote a single state-value function, and let $Q_{\theta_j}(s, a)$, $j \in \{1, 2\}$, denote two Q-functions with separately parameterized networks. We maintain corresponding target critics $Q_{\bar{\theta}_j}(s, a)$ and update the shared state-value function using their conservative estimate:

$$\mathcal{L}_V(\psi) = \mathbb{E}_{(s,a)\sim\mathcal{D}} \left[L_2^{\tau_{\exp}} \left(\min_{j\in\{1,2\}} Q_{\bar{\theta}_j}(s, a) - V_\psi(s) \right) \right], \tag{6}$$

and update each online Q-function using the same state-value function:

$$\mathcal{L}_Q(\theta_j) = \mathbb{E}_{(s,a,r,s',d)\sim\mathcal{D}} \left[\left(r + \gamma(1-d) V_\psi(s') - Q_{\theta_j}(s, a) \right)^2 \right], \tag{7}$$

where $j \in \{1, 2\}$, $d \in \{0, 1\}$ indicates a terminal transition, and $L_2^{\tau_{\exp}}(u) = |\tau_{\exp} - \mathbb{I}(u < 0)|u^2$ denotes the expectile regression loss. The target critics are softly updated after each critic update.

For action selection, we use the conservative estimate and select the highest-value action from the behavior-consistent candidate set:

$$a^* = \arg \max_{a \in \mathcal{A}_H(s)} \min\{Q_{\theta_1}(s, a), Q_{\theta_2}(s, a)\}. \tag{8}$$

The action selection proceeds in two stages: ensemble agreement filters candidates inconsistent across the learned behavior generators, and the IQL critic selects the highest-value action from the remaining set. Since value gradients are not backpropagated through the generators, behavior modeling and value learning remain decoupled and can be trained independently in parallel, accelerating training. Appendix B provides the VIMLE training and inference procedures.

3.4 ANALYSIS OF CONSISTENCY FILTERING AND VALUE GUIDANCE

Existing offline RL methods often incorporate value information into policy optimization through weighted updates or value-based regularization (Kumar et al., 2020; Kostrikov et al., 2021). In contrast, VIMLE decouples behavior modeling from value optimization and adopts ensemble-based behavioral consistency filtering followed by value-based selection. To motivate this design, we construct a toy example comparing different forms of value guidance for IMLE.

The toy example considers a continuous action space $a = (a_1, a_2) \in [-1, 1]^2$ and an offline dataset $\mathcal{D} = \{a_j\}_{j=1}^{10000}$ sampled from an equally weighted mixture of four two-dimensional Gaussian distributions, resulting in four distinct behavior modes. To simulate different returns, we assign high rewards to two of the four behavior modes and low rewards to the other two.

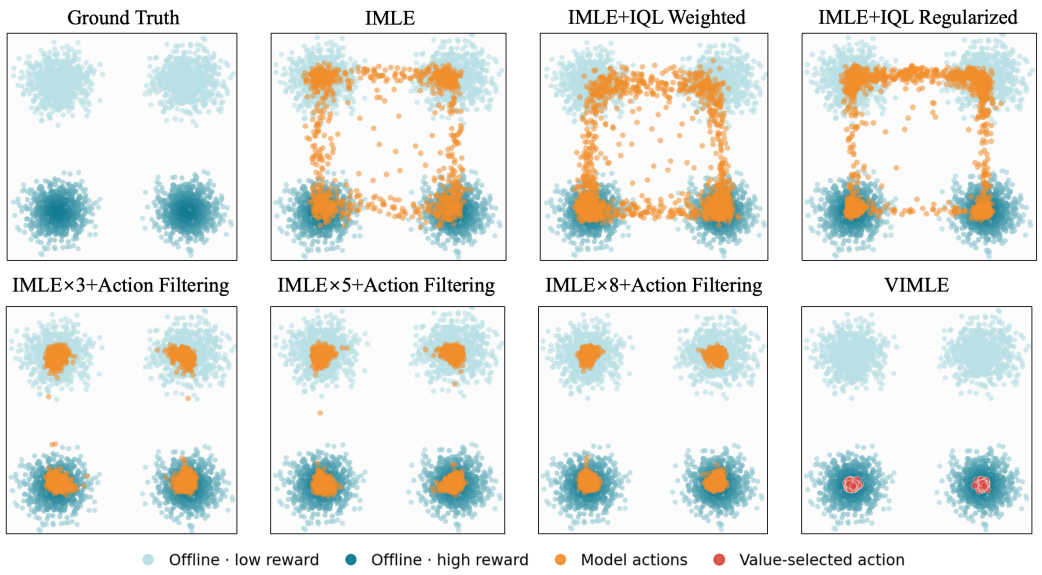


Figure 3: **Toy example for ensemble-based behavioral consistency filtering and value-guided action selection.** The figure compares a single IMLE generator, two forms of direct value guidance, ensemble-based behavioral consistency filtering with different ensemble sizes, and the complete VIMLE framework. Light-blue and dark-blue points denote low- and high-value offline actions.

As shown in Figure 3, the single IMLE generator recovers all four behavior modes, illustrating that conditional IMLE can represent the multimodal behavior distribution. Its samples nevertheless also occupy low-density regions between modes, so multimodal coverage alone does not make every generated candidate reliable for value-based selection. Directly incorporating value information does not fully resolve this issue: IMLE+IQL Weighted and IMLE+IQL Regularized shift generated actions toward higher-value regions, yet candidates remain between the separated behavior modes.

VIMLE instead filters candidates before applying value guidance. As the ensemble size increases from 3 to 5 and 8, the retained candidates increasingly concentrate on the four observed behavior modes, while inter-modal low-density candidates are suppressed. The value function is then applied only to this more reliable candidate set and selects actions from the two high-value modes. Together, these results show that IMLE preserves multimodal coverage without collapsing to a single averaged mode, ensemble agreement reduces the risk of value-estimation errors induced by OOD candidates, and the value function selects high-value actions to improve policy performance. Further details on value-guided IMLE variants and additional toy experiments are provided in Appendix E.

4 EXPERIMENTS

4.1 SETTING

Datasets. We evaluate VIMLE on the MuJoCo and AntMaze domains of D4RL benchmark (Fu et al., 2020). The MuJoCo domain consists of continuous-control tasks with dense rewards, providing a standard benchmark for evaluating policy learning under varying levels of dataset quality and behavior-policy competence. The AntMaze domain consists of sparse-reward navigation tasks with varying maze scales and behavior distributions, emphasizing long-horizon planning and trajectory stitching. Together, these domains evaluate VIMLE under complementary offline RL settings.

Baselines. We compare VIMLE against representative baselines from five categories: (1) conventional behavior modeling, including BC (Pomerleau, 1991), TD3+BC (Fujimoto & Gu, 2021), and IQL (Kostrikov et al., 2021); (2) model-based methods, including MOPO (Yu et al., 2020) and SUMO (Qiao et al., 2025); (3) diffusion-based policy methods, including Diffuser (Janner et al., 2022), IDQL (Hansen-Estruch et al., 2023), and SRPO (Chen et al., 2024); (4) sequence modeling

methods, including Decision Mamba (DM) (Lv et al., 2024) and Reinformer (Zhuang et al., 2024); (5) flow-based policy methods, including FQL (Park et al., 2025).

Setup. We implement VIMLE based on the implementations of IMLE (Li & Malik, 2018) and IQL (Kostrikov et al., 2021). The value network and policy generators are parameterized by three-layer MLPs (Rumelhart et al., 1986) with 256 hidden units per layer and ReLU activations (Nair & Hinton, 2010). We use the Adam optimizer (Kingma & Ba, 2014) throughout training, with a learning rate of 3×10^{-4} for both the value network and the policy generator. All experiments are conducted on a single NVIDIA RTX 3090 GPU.

Details of the D4RL datasets, baseline implementations, and remaining hyperparameter settings are provided in Appendix A and Appendix H.

4.2 PERFORMANCE ON MUJOCO DOMAIN

Table 1 summarizes the performance of all methods on the D4RL MuJoCo benchmark. Results are reported as normalized scores, where 100 corresponds to expert-level performance (Fu et al., 2020).

Overall, VIMLE achieves an average score of 94.18 across the 12 MuJoCo tasks and 90.32 over the nine non-expert tasks. Its gains are particularly pronounced on the medium-replay and medium-expert datasets, demonstrating robustness to variations in behavior quality and data distribution. Although not uniformly superior, VIMLE remains competitive across environments while avoiding iterative action generation. Appendix C further compares training and inference efficiency. Compared with Diffuser, VIMLE reduces training time and inference latency by 92.5% and 93.6%, respectively, demonstrating a favorable balance between policy performance and computational cost.

Table 1: **Normalized scores on the D4RL MuJoCo tasks.** For VIMLE, results are reported as the mean and standard deviation over 10 evaluations across 5 random seeds. Dataset abbreviations are as follows: "m" for 'medium', "m-r" for 'medium-replay', "m-e" for 'medium-expert', and "e" for 'expert'. DM and Rf denote Decision Mamba and Reinformer, respectively. The best and second-best results are highlighted in **bold** and underlined, respectively.

Dataset	Conventional			Model-based		Diffusion			Sequence		Flow	VIMLE
	BC	TD3+BC	IQL	MOPO	SUMO	Diffuser	IDQL	SRPO	DM	Rf	FQL	
hopper-m	52.9	59.3	66.2	26.5	94.3	89.5	70.1	<u>95.5</u>	**98.5**	81.6	74.5	90.8 ± 0.9
hopper-m-r	18.1	60.9	94.7	92.5	88.1	91.8	99.4	<u>101.2</u>	89.1	83.3	85.4	$\mathbf{102.9 \pm 14.5}$
hopper-m-e	52.5	98.0	91.5	51.7	100.7	107.2	110.1	100.1	<u>111.9</u>	107.8	86.2	$\mathbf{112.2 \pm 0.3}$
hopper-e	–	109.6	108.8	16.2	106.5	–	–	–	**112.5**	–	–	<u>112.4 ± 0.2</u>
halfcheetah-m	42.6	48.3	47.4	40.2	**84.2**	43.8	51.5	<u>60.4</u>	43.8	42.9	60.1	60.2 ± 0.2
halfcheetah-m-r	36.6	44.6	44.2	54.0	**76.7**	36.0	46.5	51.4	40.8	39.0	51.1	<u>57.6 ± 0.4</u>
halfcheetah-m-e	55.2	90.7	86.7	57.9	99.6	90.3	91.3	92.2	93.5	92.0	<u>99.8</u>	$\mathbf{100.4 \pm 1.9}$
halfcheetah-e	–	93.4	<u>95.0</u>	1.4	**110.2**	–	–	–	93.5	–	–	94.8 ± 0.1
walker2d-m	75.3	83.7	78.3	14.0	66.3	79.4	<u>88.1</u>	84.4	80.3	80.5	72.7	$\mathbf{90.5 \pm 0.5}$
walker2d-m-r	26.0	81.8	73.8	42.7	67.8	58.3	**89.1**	84.6	79.3	72.9	82.1	<u>88.6 ± 2.4</u>
walker2d-m-e	107.5	110.1	109.6	55.0	<u>113.4</u>	107.4	110.6	**114.0**	111.6	109.4	100.5	109.8 ± 0.4
walker2d-e	–	<u>110.0</u>	109.4	0.1	105.9	–	–	–	108.3	–	–	$\mathbf{110.1 \pm 0.3}$
Average (All)	–	82.5	83.8	37.7	<u>92.8</u>	–	–	–	88.6	–	–	**94.18**
Average (w/o e)	51.9	75.3	76.9	48.3	<u>87.9</u>	78.2	84.1	87.1	83.2	78.8	79.2	**90.32**

4.3 PERFORMANCE ON ANTMAZE DOMAIN

To further evaluate VIMLE in sparse-reward, long-horizon settings, we conduct experiments on the D4RL AntMaze benchmark, which requires stitching suboptimal trajectory segments. We consider six tasks—umaze, umaze-diverse, medium-play, medium-diverse, large-play, and large-diverse—covering different maze scales and behavior distributions.

Table 2 reports the normalized scores on AntMaze. VIMLE achieves an average score of 75.5 across the 6 tasks. As the maze scale increases, BC, DT, and several behavior-constrained methods suffer substantial performance degradation, with some methods failing entirely on the medium and large mazes, whereas VIMLE maintains consistently non-zero and comparatively strong performance.

Table 2: **Normalized scores on the D4RL AntMaze tasks.** For VIMLE, results are reported as the mean and standard deviation over 10 evaluations across 5 random seeds. The best and second-best results are highlighted in **bold** and underlined, respectively.

Dataset	BC	TD3+BC	IQL	CQL	PLAS	DT	IFQL	EDP	SRPO	VIMLE
antmaze-umaze	54.6	78.6	87.5	74.0	70.7	59.2	84.3	<u>94.2</u>	**97.1**	**97.1 ± 1.5**
antmaze-umaze-diverse	45.6	71.4	62.2	**84.0**	45.3	53.0	74.5	79.0	82.1	<u>83.9 ± 1.4</u>
antmaze-medium-play	0.0	10.6	71.2	61.2	16.0	0.0	59.9	<u>81.8</u>	80.7	**83.3 ± 2.6**
antmaze-medium-diverse	0.0	3.0	70.0	53.7	0.7	0.0	72.3	<u>82.3</u>	75.0	**84.4 ± 4.4**
antmaze-large-play	0.0	0.2	39.6	15.8	0.7	0.0	49.9	42.3	**53.6**	<u>50.7 ± 5.2</u>
antmaze-large-diverse	0.0	0.0	47.5	14.9	0.3	0.0	<u>55.5</u>	**60.6**	53.6	53.3 ± 4.8
Average	16.7	27.3	63.0	50.6	22.3	18.7	66.1	73.4	<u>73.7</u>	**75.5**

4.4 ABLATION OF ENSEMBLE-BASED BEHAVIORAL CONSISTENCY FILTERING

To assess the role of ensemble-based behavioral consistency filtering, we replace consistency-based retention with random selection. The variant randomly retains H actions from IMLE ensemble candidate set, and the IQL critic selects the highest-value retained action. The generators, candidate count, critic, and value-based selection remain unchanged. Learning curves are shown in Figure 4.

As shown in Figure 4, VIMLE exhibits more stable learning dynamics and higher final performance on these tasks than the random-retention variant. These results suggest that ensemble agreement effectively filters unreliable candidates and mitigates the selection of OOD actions with overestimated values, thereby improving the robustness of value-guided policy extraction.

Other ablation studies on ensemble size and reference sample count are provided in Appendix G.

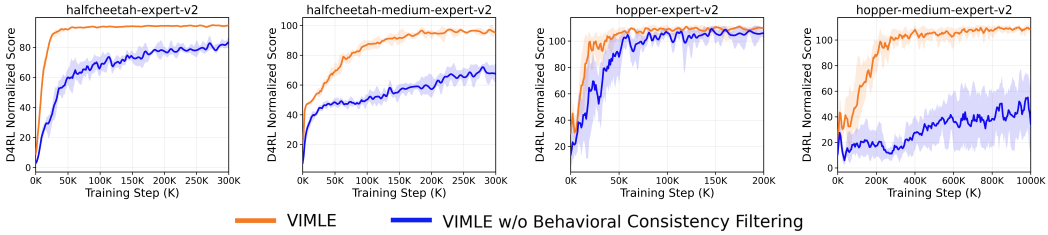


Figure 4: **Ablation of behavioral consistency filtering.** Learning curves of VIMLE with and without behavioral consistency filtering. Solid lines show the mean, with min–max ranges over 5 seeds.

4.5 COMPARISON OF IMLE-BASED POLICY LEARNING STRATEGIES

Existing offline reinforcement learning methods typically incorporate value information into policy learning through value weighting or explicit regularization. In Section 3.4, we compare four IMLE-based strategies under multimodal behavior distributions: IMLE, which only models dataset actions without action filtering or value-based selection; value-weighted IMLE (IMLE+IQL Weighted); explicitly regularized IMLE (IMLE+IQL Regularized); and VIMLE. We further evaluate these strategies on D4RL MuJoCo, with results reported in Figure 5. IMLE achieves an average normalized score of 65.24 across the 12 tasks, while IMLE+IQL Weighted and IMLE+IQL Regularized obtain 68.31 and 40.03, respectively. In comparison, VIMLE achieves 94.18, substantially outperforming all three variants. These results suggest that behavior modeling alone or direct value incorporation is insufficient for reliable policy improvement, whereas behavioral consistency filtering followed by value-guided selection better exploits multimodal action candidates.

5 RELATED WORK

Multimodal Distribution Modeling in Offline Reinforcement Learning Traditional offline RL methods, such as IQL (Kostrikov et al., 2021), and CQL (Kumar et al., 2020), often rely on simple

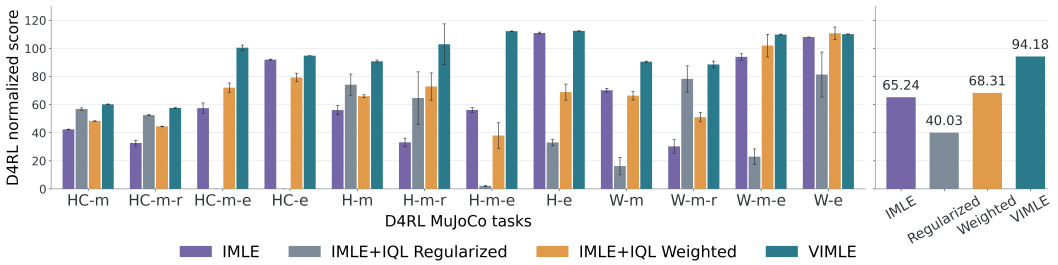


Figure 5: Comparison of IMLE-based policy learning strategies on the D4RL MuJoCo tasks. Task labels use HC, H, and W for HalfCheetah, Hopper, and Walker2d, respectively.

unimodal Gaussian policies, whose limited expressiveness makes it difficult to capture multimodal behavior distributions and may consequently produce actions poorly supported by the offline data.

This limitation has motivated the use of more expressive generative policies. Diffuser (Janner et al., 2022) formulates decision making as diffusion-based trajectory generation, producing action sequences through iterative trajectory denoising. IDQL (Hansen-Estruch et al., 2023) further applies diffusion at the action level, modeling the behavior policy with a diffusion model and extracting high-value actions through critic-based reweighting. EDP (Kang et al., 2023) subsequently improves the training efficiency of diffusion policies by avoiding the full sampling chain during policy optimization. Despite their improved multimodal expressiveness, diffusion policies rely on iterative denoising for action generation, introducing considerable computational latency. Flow-based generative models were later introduced into offline RL to partially alleviate this computational burden. For example, Flow Q-Learning (Park et al., 2025) introduces flow matching into offline reinforcement learning and distills the iterative flow policy into a one-step policy for efficient execution. Nevertheless, flow-based approaches generally still require either iterative generation or an additional policy distillation stage, which introduces extra computational overhead and increases the complexity of policy training and deployment.

IMLE for Multimodal Generation and Policy Learning IMLE (Li & Malik, 2018) is an implicit generative modeling approach that captures multimodal distributions while avoiding the iterative generation required by diffusion- and flow-based models, enabling efficient one-step sampling. These properties make IMLE suitable for tasks requiring both multimodal modeling and efficient generation, leading to its adoption across diverse domains, including conditional IMLE for diverse image generation from semantic layouts (Li et al., 2019), Shape IMLE for improving diversity in text-guided 3D shape generation (Liu et al., 2022), and WIMLE for learning stochastic multimodal world models (Aghabozorgi et al., 2026). More recently, IMLE has been introduced into policy generation. IMLE Policy (Rana et al., 2025), PRISM (Bhaskar et al., 2026b), and MIRROR (Bhaskar et al., 2026a) employ IMLE-based generation for multimodal behavior cloning and robotic policy learning. However, these methods primarily focus on imitation learning and behavior reproduction, without explicitly using value information to guide policy improvement.

6 CONCLUSION

We propose VIMLE, a value-guided IMLE ensemble framework for offline reinforcement learning that combines multimodal behavior modeling, behavioral consistency filtering, and value-guided action selection. By decoupling IMLE behavior modeling from value learning, VIMLE allows both components to be trained independently while retaining efficient one-step action generation. Ensemble agreement restricts value-based selection to behavior-consistent candidates, reducing the risk of exploiting unreliable OOD actions. Experiments on D4RL demonstrate competitive performance across diverse tasks. VIMLE also improves computational efficiency over iterative generative policies, with lower training cost and inference latency. Nevertheless, VIMLE may still underfit highly complex non-Gaussian behavior distributions with a finite set of IMLE samples. This limitation may reduce the fidelity of behavior modeling in settings with highly irregular action distributions.

AI USE STATEMENT

In this work, generative AI tools were used to assist with language editing and formatting checks. All AI-assisted content was reviewed and verified by the authors, including the mathematical formulations, experimental descriptions and reported results. The authors take responsibility for the final content of this work, including all text, claims, and artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

An anonymized implementation of VIMLE is provided in the supplementary material. The source code will be released upon publication.

REFERENCES

Mehran Aghabozorgi, Alireza Moazeni, Yanshu Zhang, and Ke Li. Wimle: Uncertainty-aware world models with imle for sample-efficient continuous control. In C. Vondrick, B. Hariharan, C. Raffel, L. Pinto, D. Yang, and A. Faust (eds.), *International Conference on Learning Representations*, volume 2026, pp. 59660–59685, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/6179c61e70d961ddfd441ac329278c9d-Paper-Conference.pdf.

Amisha Bhaskar, Pratap Tokekar, Stefano Di Cairano, and Alexander Schperberg. Mirror: Multi-sensory implicit rejection-sampled robotic policy. In *ICML 2026 Workshop on Structured Probabilistic Inference Generative Modeling*, 2026a.

Amisha Bhaskar, Pratap Tokekar, Stefano Di Cairano, and Alexander Schperberg. Prism: Performer rs-imle for single-pass multisensory imitation learning. *arXiv preprint arXiv:2602.02396*, 2026b.

Huayu Chen, Cheng Lu, Zhengyi Wang, Hang Su, and Jun Zhu. Score regularized policy optimization through diffusion behavior, 2024. URL https://arxiv.org/abs/2310.07297.

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

JaeHyeok Doo, Byeongguk Jeon, Seonghyeon Ye, Kimin Lee, and Minjoon Seo. Q-flow: Stable and expressive reinforcement learning with flow-based policy, 2026. URL https://arxiv.org/abs/2605.13435.

Justin Fu, Aviral Kumar, Ofir Nachum, George Tucker, and Sergey Levine. D4rl: Datasets for deep data-driven reinforcement learning. *arXiv preprint arXiv:2004.07219*, 2020.

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

Scott Fujimoto, David Meger, and Doina Precup. Off-policy deep reinforcement learning without exploration. In *International conference on machine learning*, pp. 2052–2062. PMLR, 2019.

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

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

Michael Janner, Yilun Du, Joshua B Tenenbaum, and Sergey Levine. Planning with diffusion for flexible behavior synthesis. *arXiv preprint arXiv:2205.09991*, 2022.

Xingyu Jiang, Ning Gao, Xiuhui Zhang, Hongkun Dou, and Yue Deng. Value-aligned behavior cloning for offline reinforcement learning via bi-level optimization. In *International Conference on Learning Representations*, volume 2025, pp. 8228–8252, 2025.

Bingyi Kang, Xiao Ma, Chao Du, Tianyu Pang, and Shuicheng Yan. Efficient diffusion policies for offline reinforcement learning. *Advances in Neural Information Processing Systems*, 36:67195–67212, 2023.

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

Bangalore Ravi Kiran, Ibrahim Sobh, Victor Talpaert, Patrick Mannion, Ahmad A. Al Sallab, Senthil Kumar Yogamani, and Patrick Pérez. Deep reinforcement learning for autonomous driving: A survey. *CoRR*, abs/2002.00444, 2020. URL https://arxiv.org/abs/2002.00444.

Ilya Kostrikov, Ashvin Nair, and Sergey Levine. Offline reinforcement learning with implicit q-learning. *arXiv preprint arXiv:2110.06169*, 2021.

Aviral Kumar, Justin Fu, Matthew Soh, George Tucker, and Sergey Levine. Stabilizing off-policy q-learning via bootstrapping error reduction. *Advances in neural information processing systems*, 32, 2019.

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

Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv preprint arXiv:2005.01643*, 2020.

Ke Li and Jitendra Malik. Implicit maximum likelihood estimation. *arXiv preprint arXiv:1809.09087*, 2018.

Ke Li, Tianhao Zhang, and Jitendra Malik. Diverse image synthesis from semantic layouts via conditional imle. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019.

Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. *arXiv preprint arXiv:2210.02747*, 2022.

Zhengzhe Liu, Yi Wang, Xiaojuan Qi, and Chi-Wing Fu. Towards implicit text-guided 3d shape generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 17896–17906, June 2022.

Qi Lv, Xiang Deng, Gongwei Chen, Michael Y Wang, and Liqiang Nie. Decision mamba: A multi-grained state space model with self-evolution regularization for offline RL. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=dc4xbVfdzy.

Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pp. 807–814, 2010.

Thanh Nguyen and Chang D. Yoo. One-step flow q-learning: Addressing the diffusion policy bottleneck in offline reinforcement learning, 2026. URL https://arxiv.org/abs/2508.13904.

Alexander Nikulin, Vladislav Kurenkov, Denis Tarasov, and Sergey Kolesnikov. Anti-exploration by random network distillation. In *International conference on machine learning*, pp. 26228–26244. PMLR, 2023.

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

Dean A Pomerleau. Efficient training of artificial neural networks for autonomous navigation. *Neural computation*, 3(1):88–97, 1991.

Martin L Puterman. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

Ziyu Qiao, Jiashuo Lyu, Kaixuan Jiao, Qi Liu, and Xiang Li. Sumo: Search-based uncertainty estimation for model-based offline reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2025. doi: 10.1609/aaai.v39i19.34206. URL https://doi.org/10.1609/aaai.v39i19.34206.

Krishan Rana, Robert Lee, David Pershouse, and Niko Suenderhauf. Imle policy: Fast and sample efficient visuomotor policy learning via implicit maximum likelihood estimation. *arXiv preprint arXiv:2502.12371*, 2025.

David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *nature*, 323(6088):533–536, 1986.

Richard S Sutton. Learning to predict by the methods of temporal differences. *Machine learning*, 3 (1):9–44, 1988.

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

Zhendong Wang, Jonathan J Hunt, and Mingyuan Zhou. Diffusion policies as an expressive policy class for offline reinforcement learning. *arXiv preprint arXiv:2208.06193*, 2022.

Yifan Wu, George Tucker, and Ofir Nachum. Behavior regularized offline reinforcement learning. *arXiv preprint arXiv:1911.11361*, 2019.

Rui Yang, Chenjia Bai, Xiaoteng Ma, Zhaoran Wang, Chongjie Zhang, and Lei Han. Rorl: Robust offline reinforcement learning via conservative smoothing. *Advances in neural information processing systems*, 35:23851–23866, 2022.

Tianhe Yu, Garrett Thomas, Lantao Yu, Stefano Ermon, James Y Zou, Sergey Levine, Chelsea Finn, and Tengyu Ma. Mopo: Model-based offline policy optimization. *Advances in neural information processing systems*, 33:14129–14142, 2020.

Wenxuan Zhou, Sujay Bajracharya, and David Held. Plas: Latent action space for offline reinforcement learning. In *Conference on Robot Learning*, pp. 1719–1735. PMLR, 2021.

Zifeng Zhuang, Dengyun Peng, Jinxin Liu, Ziqi Zhang, and Donglin Wang. Reinformer: Max-return sequence modeling for offline rl, 2024. URL https://arxiv.org/abs/2405.08740.

A Details of Experimental Settings

A.1 Details of D4RL

D4RL is a standard benchmark for offline reinforcement learning, providing fixed datasets collected by prior controllers or demonstrations (Fu et al., 2020). We conducted experiments based on four task types: Gym-MuJoCo, AntMaze, Franka Kitchen, and Adroit, and report performance using the standard D4RL normalized score,

$$Score = 100 \times \frac{R - R_{\text{random}}}{R_{\text{expert}} - R_{\text{random}}},$$

where R is the return of the evaluated policy, and R_{random} and R_{expert} denote the reference returns of random and expert policies, respectively.

A.1.1 Gym-MuJoCo

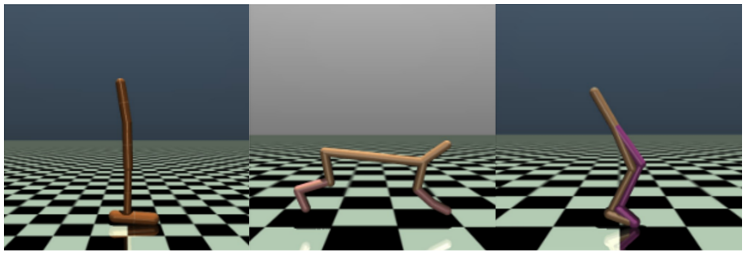


Figure 6: Visualization of the Gym-MuJoCo tasks. From left to right: Hopper, Half-Cheetah, and Walker2d.

Gym-MuJoCo contains three continuous-control tasks: Hopper, Half-Cheetah, and Walker2d. They use dense rewards and test whether a policy can produce stable forward locomotion. Each task is paired with four v2 datasets: medium, medium-replay, medium-expert, and expert. medium is collected from a Soft Actor-Critic policy stopped at approximately one-third of expert performance. medium-replay contains transitions accumulated in the replay buffer while that policy was being trained, and therefore mixes behaviors from different training stages. expert consists of trajectories from a fully trained high-performing policy. medium-expert mixes equal portions of medium and expert data. These datasets preserve the same control problem while changing behavior quality and distributional heterogeneity.

A.1.2 AntMaze

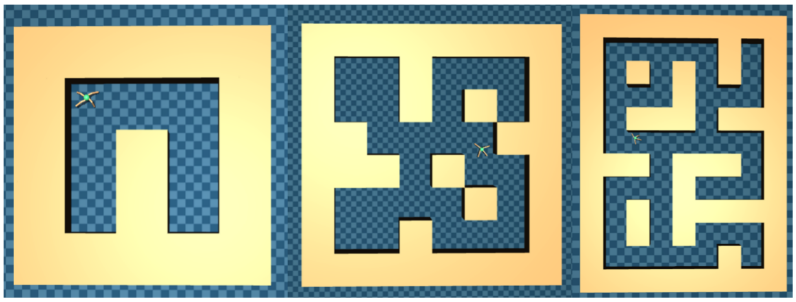


Figure 7: Visualization of the AntMaze tasks. From left to right: umaze, medium, and large.

AntMaze replaces the point-mass agent of a basic maze with an eight-degree-of-freedom quadruped Ant. A goal-reaching controller and a high-level waypoint generator produce trajectories composed of local movements that must later be assembled into a complete route. umaze, medium and large

denote increasing map scales. In umaze, the Ant moves from a fixed start to a specified goal. diverse datasets use random starts and goals, whereas play datasets use hand-selected starts and intermediate locations that are not necessarily the evaluation goal. The domain therefore emphasizes sparse-reward planning and trajectory stitching rather than single-step action imitation.

A.1.3 FRANKA KITCHEN

Franka Kitchen controls a 9-DoF Franka robot interacting with a microwave, kettle, overhead light, cabinets, and oven. Each episode requires a sequence of object manipulations to reach a target configuration. kitchen-complete contains demonstrations that execute the required sequence successfully. In kitchen-partial, only part of the data completes relevant tasks and the remaining trajectories contain unrelated subtasks. kitchen-mixed contains no complete successful sequence, requiring the learner to combine useful sub-trajectories from different episodes. Kitchen consequently evaluates temporal composition and the selective reuse of task-relevant behavior.

A.1.4 ADROIT

Adroit uses a 24-DoF Shadow Hand for four tasks: rotating a pen (pen), opening a door (door), hammering a nail (hammer), and relocating a ball (relocate). Its data are divided into human, cloned, and expert. The human set contains a small number of demonstrations (25 trajectories in the benchmark setting); the expert set is generated by a high-performing RL policy; and the cloned set combines demonstrations with data from an imitation policy trained on them. Sparse rewards, high-dimensional actions, and a narrow expert distribution make Adroit a demanding test of dexterous control under limited and noisy supervision.

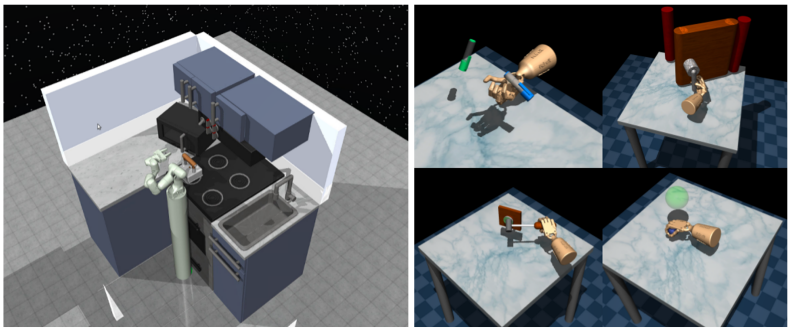


Figure 8: Visualization of the Franka Kitchen environment (left) and Adroit environment (right).

A.2 DETAILS OF BASELINE

We select representative algorithms from offline reinforcement learning as baselines, which are divided into five categories:

- **Conventional behavior modeling:** BC (Pomerleau, 1991), TD3+BC (Fujimoto & Gu, 2021), and IQL (Kostrikov et al., 2021).
- **Model-based methods:** MOPO (Yu et al., 2020) and SUMO (Qiao et al., 2025).
- **Diffusion-based policy methods:** Diffuser (Janner et al., 2022), IDQL (Hansen-Estruch et al., 2023), and SRPO (Chen et al., 2024).
- **Sequence modeling methods:** DM (Lv et al., 2024) and Reinformer (Zhuang et al., 2024).
- **Flow-based policy methods:** FQL (Park et al., 2025).

The environment-specific additional baselines are as follows:

- AntMaze includes CQL (Kumar et al., 2020), DT (Chen et al., 2021), PLAS (Zhou et al., 2021), EDP (Kang et al., 2023), and IFQL (Park et al., 2025; Doo et al., 2026). Specifically, CQL mitigates overestimation of out-of-distribution actions through conservative

value learning, and DT formulates offline policy learning as return-conditioned sequence modeling; PLAS models the behavior policy in a conditional variational-autoencoder action latent space, thereby implicitly constraining policy improvement to actions supported by the offline data. EDP uses a diffusion model to represent complex action distributions, and IFQL is a flow-based baseline.

- Kitchen includes PLAS and BCQ (Fujimoto et al., 2019). BCQ generates candidate actions close to the data distribution and restricts policy perturbations, reducing the risk of evaluating out-of-distribution actions.
- Adroit includes SAC-RND (Nikulin et al., 2023), BEAR (Kumar et al., 2019), and RORL (Yang et al., 2022). SAC-RND introduces random network distillation into SAC and uses feature modulation to promote conservatism; BEAR constrains the deviation between the learned policy and the behavior distribution through maximum mean discrepancy; and RORL suppresses unreliable value estimates through conservative smoothing to improve offline policy robustness.

Except for VIMLE, most baseline scores are collected from normalized results reported in the corresponding original papers or established follow-up studies, which reflect the strong performance of each method and thereby facilitate a fair comparison. All methods are compared using the standard D4RL normalized score, and the sources of the adopted baseline results are detailed below.

- **Gym-MuJoCo:** The SUMO scores are taken from Qiao et al. (2025). The SRPO, DM, and Reinformer scores are taken from Chen et al. (2024); Lv et al. (2024); Zhuang et al. (2024), respectively. The Diffuser, IDQL and FQL scores are taken from Nguyen & Yoo (2026). The BC scores follow Kostrikov et al. (2021), while the TD3+BC, IQL, and MOPO scores are reproduced from Jiang et al. (2025).
- **AntMaze:** The BC, TD3+BC, IQL, and CQL scores follow Kostrikov et al. (2021); the PLAS and EDP scores are taken from Zhou et al. (2021); Kang et al. (2023), respectively; and the DT scores are taken from Wang et al. (2022). The SRPO scores are taken from Chen et al. (2024). The IFQL scores are taken from Doo et al. (2026).
- **Kitchen and Adroit:** All baseline scores on the Kitchen and Adroit domains are reproduced from the benchmark compilation reported by Jiang et al. (2025).

B VIMLE Training and Inference Procedures

During training, VIMLE optimizes its conditional behavior modeling module in parallel, as summarized in Algorithm 1, while the value learning module is trained concurrently, as summarized in Algorithm 2. During inference, VIMLE performs ensemble-based behavioral consistency filtering followed by value-guided action selection, as summarized in Algorithm 3.

Algorithm 1 Conditional IMLE training

Input: Offline state–action dataset $\mathcal{D}_{sa}$; generators $\{g_{\phi_m}\}_{m=1}^{M}$; IMLE sample count K
for $m = 1$ to M **in parallel do**
  while the prescribed number of updates is not reached **do**
    Sample minibatch $\{(s_i, a_i)\}_{i=1}^{B} \sim \mathcal{D}_{sa}$ and obtain $\{a_{i,k}\}_{k=1}^{K}$ according to Eq. 2
    Match each a_i using Eq. 3
    Update ϕ_m by minimizing Eq. 4
  end while
end for
Output: Independently trained generators $\{g_{\phi_m}\}_{m=1}^{M}$

Algorithm 2 Value function training

Input: Offline dataset $\mathcal{D}$; twin critics $Q_{\theta_1}, Q_{\theta_2}$; target critics $Q_{\bar{\theta}_1}, Q_{\bar{\theta}_2}$; value function V_ψ
Initialize $\bar{\theta}_j \leftarrow \theta_j$ for $j \in \{1, 2\}$
while the prescribed number of updates is not reached **do**
    Sample $(s, a, r, s', d) \sim \mathcal{D}$
    Update V_ψ using Eq. 6
    Update the twin critics $Q_{\theta_1}, Q_{\theta_2}$ using Eq. 7.
    Soft-update $\bar{\theta}_j \leftarrow (1 - \tau_{\text{target}})\bar{\theta}_j + \tau_{\text{target}}\theta_j$ for $j \in \{1, 2\}$.
end while
Output:IQL critics for action ranking

Algorithm 3 VIMLE inference

Input: State s; M generators; N candidates; R references; retain H candidates
for $m = 1$ to M **in parallel do**
    Generate the assigned candidate subset $\{a_i\}$ using Eq. 1
    Sample $\{\tilde{z}_{m,r}\}_{r=1}^{R} \sim \mathcal{N}(0, I)$ and generate $\{\tilde{a}_{m,r}\}_{r=1}^{R}$
end for
Compute $d_m(s, a_i)$ and $C(s, a_i)$ using Eq. 5
$\mathcal{A}_H(s) \leftarrow$ the H candidates with the smallest $C(s, a_i)$ values.
Select the action according to Eq. 8
Output: Selected action a^*

Clarification on Action Candidate Allocation. The total number of action candidates N is distributed as evenly as possible across the M generators. Specifically, the first $N \bmod M$ generators each produce $\lceil N/M \rceil$ candidates, and the remaining generators each produce $\lfloor N/M \rfloor$ candidates. Thus, for the default setting $N = 64$ and $M = 5$, four generators each produce 13 candidates and the remaining generator produces 12.

C Analysis of Training and Inference Efficiency

VIMLE decouples IMLE-based multimodal behavior modeling from value learning, allowing the action generators and value function to be trained independently and in parallel, thereby avoiding the serial optimization overhead caused by repeated interaction between policy generation and value learning. Meanwhile, IMLE performs one-step forward generation and naturally supports parallel candidate sampling, offering substantially higher inference efficiency than diffusion-based methods that rely on iterative denoising. We compare the training time and inference latency of different methods on the hopper-medium-expert-v2 dataset under the same training budget of 1M steps, as reported in Table 3.

VIMLE achieves a normalized score of 112.2 with only 2.02 hours of training, reducing training time by approximately 92.5% compared with Diffuser's 26.79 hours. Its per-decision inference latency is only 0.95 ms, making it approximately 15.5$\times$ faster than Diffuser at 14.73 ms. Moreover, increasing the ensemble size from 1 to 5 raises the inference latency only marginally, from 0.88 ms to 0.95 ms, demonstrating that the parallel generation mechanism introduces little additional computational overhead. Overall, VIMLE substantially reduces both training and inference costs in generative offline reinforcement learning while maintaining strong policy performance.

D Theoretical Analysis of Ensemble Agreement for Candidate Filtering

VIMLE ranks candidate actions using an ensemble consistency distance computed from nearest-neighbor distances to reference actions generated by IMLE. We first characterize the relationship between this distance and the local conditional density learned by an individual generator, and then establish the theoretical motivation for aggregating the distances across independently learned generators.

Table 3: Training efficiency and inference latency on the hopper-medium-expert-v2 dataset. Training time is reported in hours and inference latency in milliseconds, with all measurements conducted on a single NVIDIA RTX 3090 GPU. VIMLE-1, VIMLE-3, and VIMLE-5 denote VIMLE with 1, 3, and 5 IMLE generators in the ensemble, respectively.

Method	BC	TD3+BC	IQL	CQL	FQL	DT	Diffuser	VIMLE-1	VIMLE-3	VIMLE-5
Training Time (h)	1.30	3.20	3.40	8.80	3.46	14.90	26.79	1.67	1.87	2.02
Inference Time (ms)	0.17	0.16	0.24	0.60	2.00	1.35	14.73	0.88	0.92	0.95
Normalized Score	52.5	98.0	91.5	96.9	86.2	107.6	107.2	102.2	104.9	**112.2**

D.1 Nearest-Neighbor Distance and Local Conditional Density

For a fixed state s, the nearest-neighbor distance of a candidate action a to the m-th IMLE generator is measured using R reference actions:

$$d_m(s,a) = \min_{1\le r\le R} \|a - \tilde{a}_{m,r}\|_2^2, \qquad \tilde{a}_{m,r}\sim q_m(\cdot \mid s).$$

Assuming that the action space has dimension d_a, $q_m(\cdot \mid s)$ is continuous and positive near a, and R is sufficiently large, we have

$$\Pr\left(d_m(s,a) \le \epsilon\right) \approx 1 - \exp\left(-R q_m(a \mid s) V_{d_a} \epsilon^{d_a/2}\right), \tag{9}$$

where R denotes the number of reference actions generated by each model, d_a is the dimensionality of the action space, and V_{d_a} denotes the volume of the d_a-dimensional unit ball.

Theoretical Derivation. Define the local probability mass of the m-th generator around candidate action a as

$$\mu_m(s,a;\epsilon) = \Pr_{\tilde{a}\sim q_m(\cdot|s)}\left(\|\tilde{a} - a\|_2^2 \le \epsilon\right).$$

Here, $\mu_m(s,a;\epsilon)$ denotes the probability that an action sampled from $q_m(\cdot \mid s)$ falls within a ball centered at a with radius $\sqrt{\epsilon}$.

If $d_m(s,a) > \epsilon$, none of the R reference actions generated by the model falls within this neighborhood. For an individual reference action,

$$\Pr\left(\|\tilde{a}_{m,r} - a\|_2^2 > \epsilon\right) = 1 - \mu_m(s,a;\epsilon).$$

Since the R reference actions are sampled independently,

$$\Pr\left(d_m(s,a) > \epsilon\right) = \left[1 - \mu_m(s,a;\epsilon)\right]^R.$$

Equivalently,

$$\Pr\left(d_m(s,a) \le \epsilon\right) = 1 - \left[1 - \mu_m(s,a;\epsilon)\right]^R. \tag{10}$$

For sufficiently small ϵ, the local probability mass can be written as

$$\mu_m(s,a;\epsilon) = \int_{\|a'-a\|_2^2\le\epsilon} q_m(a' \mid s)\, da'.$$

Within a sufficiently small neighborhood of a, $q_m(a' \mid s) \approx q_m(a \mid s)$. Therefore,

$$\mu_m(s,a;\epsilon) = q_m(a \mid s) V_{d_a} \epsilon^{d_a/2} + o\left(\epsilon^{d_a/2}\right). \tag{11}$$

Substituting Eq. 11 into Eq. 10 gives

$$\Pr\left(d_m(s,a) \le \epsilon\right) \approx 1 - \left[1 - q_m(a \mid s) V_{d_a} \epsilon^{d_a/2}\right]^R.$$

For a sufficiently small local neighborhood, using $(1-x)^R \approx \exp(-Rx)$, we obtain Eq. 9.

Eq. 9 shows that, for fixed R and ϵ, a larger conditional density $q_m(a \mid s)$ implies a smaller $\Pr(d_m(s,a) > \epsilon)$. When a candidate action lies in a high-density region learned by the generator, reference samples are more likely to appear nearby, making $d_m(s,a)$ more likely to be small. Conversely, for actions in low-density regions, nearby reference samples are less likely to occur, resulting in larger nearest-neighbor distances.

Thus, $d_m(s,a)$ is an inverse statistic of the local conditional density represented by the generator: a smaller nearest-neighbor distance indicates higher local density, whereas a larger distance indicates lower local density. It should be noted that, although the analysis assumes a large R, practical training and inference costs limit R in implementation, which can make nearest-neighbor distances less discriminative in high-dimensional action spaces and remains a limitation of the current approach.

D.2 Ensemble Consistency and Cross-Model Agreement

The above analysis shows that the nearest-neighbor distance $d_m(s,a)$ of an individual generator is closely related to the local conditional density learned around candidate action a. VIMLE further aggregates M independently trained IMLE generators and defines the ensemble consistency distance as

$$C(s,a) = \frac{1}{M} \sum_{m=1}^{M} d_m(s,a).$$

Under the same local continuity assumption as in Eq. 9, and further assuming $q_m(a \mid s) > 0$, when the number of reference actions R is sufficiently large such that nearest-neighbor distances concentrate in a local neighborhood of a, we obtain

$$\mathbb{E}[C(s,a)] \approx \frac{\Gamma\left(1 + \frac{2}{d_a}\right)}{M(RV_{d_a})^{2/d_a}} \sum_{m=1}^{M} q_m(a \mid s)^{-2/d_a}, \tag{12}$$

and, if the reference samples from different generators are conditionally independent,

$$\mathrm{Var}[C(s,a) \mid q_1, \ldots, q_M] = \frac{1}{M^2} \sum_{m=1}^{M} \mathrm{Var}[d_m(s,a) \mid q_m] \leq \frac{\sigma^2(s,a)}{M}. \tag{13}$$

Here, $\Gamma(\cdot)$ denotes the Gamma function, M is the number of IMLE generators, and $\sigma^2(s,a)$ upper-bounds the conditional variance of each nearest-neighbor distance.

Eq. 12 shows that the expected ensemble consistency distance is jointly determined by the local conditional densities of all generators. Since $q_m(a \mid s)$ appears with a negative exponent, low density under any generator increases $\mathbb{E}[C(s,a)]$, whereas consistently high local density across generators leads to smaller values. Meanwhile, Eq. 13 shows that ensemble averaging reduces variance induced by finite reference sampling at a rate of $\mathcal{O}(1/M)$. Therefore, $C(s,a)$ ranks candidates according to their agreement with the learned conditional models while providing a statistically more stable score under finite reference sampling.

Theoretical Derivation. From Eq. 9, for the m-th generator,

$$\Pr\left(d_m(s,a) > \epsilon\right) \approx \exp\left(-Rq_m(a \mid s)V_{d_a}\epsilon^{d_a/2}\right).$$

We further assume that R is sufficiently large such that the probability mass of $d_m(s,a)$ is concentrated within a local neighborhood of a. As R increases, the nearest reference sample becomes increasingly likely to lie close to a, causing the distribution of $d_m(s,a)$ to concentrate at small distances. Consequently, in the large-R regime, the local approximation above captures the dominant contribution to $\mathbb{E}[d_m(s,a)]$.

Since $d_m(s,a)$ is a nonnegative random variable, its expectation can be expressed through its survival function as

$$\mathbb{E}[d_m(s,a)] = \int_0^{\epsilon_0} \Pr\left(d_m(s,a) > \epsilon\right) d\epsilon + \int_{\epsilon_0}^{\infty} \Pr\left(d_m(s,a) > \epsilon\right) d\epsilon,$$

where $\epsilon_0 > 0$ denotes a sufficiently small neighborhood scale over which the local approximation is valid.

In the large-R regime, the probability mass of $d_m(s,a)$ is concentrated at small distances, making the contribution of the tail region $\epsilon \geq \epsilon_0$ negligible. Hence,

$$\mathbb{E}[d_m(s,a)] \approx \int_0^{\epsilon_0} \Pr\left(d_m(s,a) > \epsilon\right) d\epsilon.$$

Applying the local asymptotic form over $[0, \epsilon_0]$ yields

$$\mathbb{E}[d_m(s,a)] \approx \int_0^{\epsilon_0} \exp\left(-Rq_m(a \mid s) V_{d_a} \epsilon^{d_a/2}\right) d\epsilon.$$

For compactness, define

$$\lambda_m = Rq_m(a \mid s) V_{d_a}.$$

Then,

$$\mathbb{E}[d_m(s,a)] \approx \int_0^{\epsilon_0} \exp\left(-\lambda_m \epsilon^{d_a/2}\right) d\epsilon.$$

Using the change of variables $u = \lambda_m \epsilon^{d_a/2}$, we have

$$d\epsilon = \frac{2}{d_a} \lambda_m^{-2/d_a} u^{\frac{2}{d_a}-1} du.$$

Therefore,

$$\mathbb{E}[d_m(s,a)] \approx \frac{2}{d_a} \lambda_m^{-2/d_a} \int_0^{\lambda_m \epsilon_0^{d_a/2}} e^{-u} u^{\frac{2}{d_a}-1} du.$$

When R is sufficiently large, $\lambda_m = Rq_m(a \mid s) V_{d_a}$ grows with R, and hence the upper integration limit $\lambda_m \epsilon_0^{d_a/2}$ also increases. Since the integrand $e^{-u} u^{\frac{2}{d_a}-1}$ decays rapidly for large u, extending the upper limit to infinity introduces only a negligible error. Thus,

$$\begin{aligned}\mathbb{E}[d_m(s,a)] &\approx \frac{2}{d_a} \lambda_m^{-2/d_a} \int_0^{\infty} e^{-u} u^{\frac{2}{d_a}-1} du \\ &= \frac{2}{d_a} \lambda_m^{-2/d_a} \Gamma\left(\frac{2}{d_a}\right).\end{aligned}$$

Using the Gamma-function identity $\Gamma(x+1) = x\Gamma(x)$, we obtain

$$\frac{2}{d_a}\Gamma\left(\frac{2}{d_a}\right) = \Gamma\left(1 + \frac{2}{d_a}\right).$$

Therefore,

$$\mathbb{E}[d_m(s,a)] \approx \Gamma\left(1 + \frac{2}{d_a}\right) \left[Rq_m(a \mid s) V_{d_a}\right]^{-2/d_a}. \tag{14}$$

According to the definition of the VIMLE ensemble consistency distance,

$$C(s,a) = \frac{1}{M} \sum_{m=1}^{M} d_m(s,a),$$

its expectation is

$$\mathbb{E}[C(s,a)] = \frac{1}{M} \sum_{m=1}^{M} \mathbb{E}[d_m(s,a)]. \tag{15}$$

Substituting Eq. 14 into Eq. 15, we obtain

$$\begin{aligned}\mathbb{E}[C(s,a)] &\approx \frac{1}{M}\sum_{m=1}^{M}\Gamma\left(1+\frac{2}{d_a}\right)[Rq_m(a\mid s)V_{d_a}]^{-2/d_a}\\ &= \frac{\Gamma\left(1+\frac{2}{d_a}\right)}{M(RV_{d_a})^{2/d_a}}\sum_{m=1}^{M}q_m(a\mid s)^{-2/d_a},\end{aligned}$$

which gives Eq. 12.

When the candidate action lies in low-density regions under some generators, the corresponding nearest-neighbor distances increase, leading to a larger $C(s,a)$. Conversely, consistently high density across generators yields a smaller ensemble consistency distance. Beyond aggregating cross-model agreement information, the ensemble also reduces fluctuations caused by finite reference sampling.

Conditional on the trained generators, if reference actions from different generators are sampled independently, the variance of the ensemble consistency distance is

$$\mathrm{Var}[C(s,a)\mid q_1,\ldots,q_M] = \frac{1}{M^2}\sum_{m=1}^{M}\mathrm{Var}[d_m(s,a)\mid q_m]\,.$$

If the nearest-neighbor distances across generators have bounded variances of comparable order, such that

$$\mathrm{Var}[d_m(s,a)\mid q_m]\le\sigma^2(s,a),$$

then

$$\mathrm{Var}[C(s,a)\mid q_1,\ldots,q_M]\le\frac{\sigma^2(s,a)}{M},$$

which gives Eq. 13.

Therefore, as the number of generators M increases, fluctuations in the ensemble consistency distance induced by random reference sampling decrease. Thus, the ensemble mechanism provides a more stable agreement score under finite reference sampling. However, this conditional result does not remove modeling errors shared across generators, which remains an unresolved limitation of the ensemble mechanism.

E Detailed Analysis of Ensemble-Based Behavioral Consistency Filtering and Value Guidance

In Section 3.4, we compare VIMLE with IMLE+IQL Weighted and IMLE+IQL Regularized through visualization to demonstrate the effectiveness of ensemble-based behavioral consistency filtering and value-guided action selection. In this section, we first provide the detailed formulations of these two value-guided IMLE variants, and then further analyze VIMLE using another toy example with a non-Gaussian multimodal behavior distribution.

E.1 Value-guided IMLE Variants

The conditional IMLE generator follows the same formulation as in VIMLE. Let $g_\phi(s,z)$ denote the conditional IMLE generator, where $z_{i,k}\sim\mathcal{N}(0,I)$ and $k=1,\ldots,K$. For each offline sample $(s_i,a_i)\sim\mathcal{D}$, we generate K candidate actions:

$$\tilde{a}_{i,k}=g_\phi(s_i,z_{i,k}).$$

The candidate action closest to the data action is then selected:

$$k_i^\star=\arg\min_{k\in\{1,\ldots,K\}}\|\tilde{a}_{i,k}-a_i\|_2^2\,,\qquad d_i=\left\|\tilde{a}_{i,k_i^\star}-a_i\right\|_2^2.$$

IMLE+IQL Weighted. The variant uses the IQL advantage of the data action to reweight the matched IMLE residual:

$$\mathcal{L} = \mathbb{E}_{(s_i,a_i)\sim\mathcal{D}}\left[w_i d_i\right],$$

where

$$w_i = \min\left\{\exp\left(\beta\left[Q_{\min}(s_i,a_i) - V(s_i)\right]\right), w_{\max}\right\}.$$

We use $\beta = 3.0$ and $w_{\max} = 100$.

IMLE+IQL Regularized. The variant retains the original unweighted IMLE matching objective while directly encouraging the matched generated action to achieve a higher target Q-value:

$$\mathcal{L} = \mathbb{E}_{(s_i,a_i)\sim\mathcal{D}}\left[d_i - \lambda Q_{\min}\left(s_i, \tilde{a}_{i,k_i^\star}\right)\right].$$

We set $\lambda = 0.1$.

E.2 Visualization on a Non-Gaussian Multimodal Behavior Distribution

As shown in Figure 9, we construct a single-state continuous-action environment whose offline behavior distribution consists of five disconnected modes, forming a clearly multimodal and non-contiguous behavior support. The upper-right, center, and lower-left modes are assigned higher rewards, whereas the upper-left and lower-right modes have lower rewards.

The visualization results are shown in Figure 9. Both direct value-guided variants noticeably bias generated actions toward higher-value regions, but still produce a considerable number of OOD actions outside the behavior support. In contrast, VIMLE substantially suppresses such OOD samples through ensemble-based behavioral consistency filtering, and the final selected actions almost entirely remain within regions supported by the offline data.

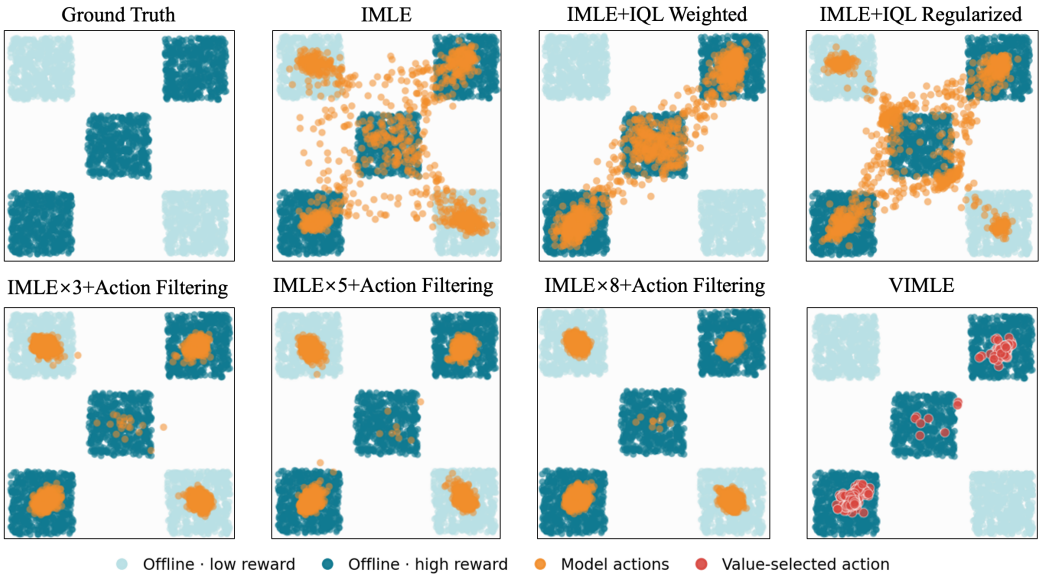


Figure 9: **Toy example on a non-Gaussian multimodal behavior distribution.** The figure compares a single IMLE generator, two forms of direct value guidance, ensemble-based behavioral consistency filtering with different ensemble sizes, and the complete VIMLE framework. Light-blue and dark-blue points denote low- and high-value offline actions.

Nevertheless, the final action distribution of VIMLE is noticeably imbalanced across the three high-value modes. Although VIMLE is able to cover multiple behavior modes, an IMLE generator with a finite candidate set may still underfit non-Gaussian distributions with complex geometric structures. Diffusion- and flow-matching-based generative models are generally more flexible for modeling such distributions, highlighting a potential limitation of the current VIMLE formulation.

F ADDITIONAL EXPERIMENTS ON ADROIT AND KITCHEN DOMAINS

We further evaluate VIMLE on the Adroit and Franka Kitchen domains of the D4RL benchmark (Fu et al., 2020). The Adroit experiments cover the pen, door, hammer, and relocate tasks with human, cloned, and expert datasets, while the Kitchen experiments use the complete, partial, and mixed datasets. The normalized scores are reported in Tables 4 and 5. For VIMLE, each task score is reported as the mean $\pm$ standard deviation; the remaining comparison scores are reproduced from the corresponding published results (Tarasov et al., 2023).

On Adroit, VIMLE achieves an average score of 55.77 across the 12 tasks and obtains the highest reported score on 9 tasks. On Kitchen, it reaches an average score of 62.17 over the three datasets, obtaining the highest reported average among the compared results. Although VIMLE is not uniformly superior on every individual task, it remains competitive overall, indicating that ensemble-based behavioral consistency filtering and value-guided candidate selection can also be effective for dexterous manipulation and multi-subtask composition.

Table 4: Normalized scores on the D4RL Kitchen tasks. For VIMLE, results are reported as the mean and standard deviation over 10 evaluations across 5 random seeds. The best and second-best results in each row are shown in bold and underlined, respectively.

Dataset	BC	IQL	CQL	PLAS	BCQ	VIMLE
Kitchen-complete	<u>65.00</u>	62.50	43.80	38.10	9.10	**66.08 $\pm$ 8.20**
Kitchen-partial	38.00	46.30	<u>49.80</u>	27.00	17.60	**58.07 $\pm$ 0.83**
Kitchen-mixed	<u>51.50</u>	51.00	51.00	30.00	11.50	**62.37 $\pm$ 1.38**
Average	51.50	<u>53.27</u>	48.20	31.70	12.73	**62.17**

Table 5: Normalized scores on the D4RL Adroit tasks. For VIMLE, results are reported as the mean and standard deviation over 10 evaluations across 5 random seeds. For the Adroit experiments, the latent dimension is set to 32; all other hyperparameters follow Table 8. The best and second-best results in each row are shown in bold and underlined.

Dataset	BC	TD3+BC	IQL	CQL	SAC-RND	BEAR	RORL	VIMLE
Pen-human	34.40	<u>81.80</u>	81.50	37.50	5.60	-1.00	33.70	**82.19 $\pm$ 7.34**
Pen-cloned	56.90	61.40	<u>77.20</u>	39.20	2.50	26.50	35.70	**79.08 $\pm$ 4.57**
Pen-expert	85.10	**146.00**	133.60	107.00	45.40	105.90	130.30	<u>140.69 $\pm$ 2.63</u>
Door-human	0.50	-0.10	3.10	<u>9.90</u>	0.00	-0.30	3.78	**14.00 $\pm$ 1.84**
Door-cloned	-0.10	0.10	<u>0.80</u>	0.40	0.20	-0.10	-0.10	**3.30 $\pm$ 0.65**
Door-expert	34.90	84.60	**105.30**	101.50	73.60	103.40	104.90	<u>105.00 $\pm$ 0.10</u>
Hammer-human	1.50	0.40	2.50	<u>4.40</u>	-0.10	0.30	2.30	**5.23 $\pm$ 2.03**
Hammer-cloned	0.80	0.80	1.10	<u>2.10</u>	0.10	0.30	1.70	**2.70 $\pm$ 0.91**
Hammer-expert	125.60	117.00	<u>129.60</u>	86.70	24.80	127.30	**132.20**	127.60 $\pm$ 0.39
Relocate-human	0.00	-0.20	0.10	<u>0.20</u>	0.00	-0.30	0.00	**0.59 $\pm$ 0.27**
Relocate-cloned	-0.10	-0.10	<u>0.20</u>	-0.10	0.00	-0.30	0.00	**0.53 $\pm$ 0.46**
Relocate-expert	101.30	<u>107.30</u>	106.50	95.00	3.40	98.60	47.80	**108.30 $\pm$ 0.48**
Average	36.73	49.92	<u>53.46</u>	40.32	12.96	38.36	41.02	**55.77**

G ABLATION STUDIES ON ENSEMBLE SIZE AND REFERENCE SAMPLE COUNT

In Section 4.4, we discussed the effect of ensemble-based behavioral consistency filtering on the performance of VIMLE. This section further examines the effects of ensemble size and reference sample count.

G.1 EFFECT OF ENSEMBLE SIZE

We conducted experiments with ensemble sizes of 1, 3, and 5 on the HalfCheetah, Hopper, and Walker2d environments. The results are presented in Table 6. As the ensemble size increases, the average performance improves from 88.41 to 90.32. When comparing $M = 5$ with $M = 1$,

performance improves on five of the six Hopper and Walker2d tasks, whereas it decreases on all three HalfCheetah tasks. This degradation may arise because consensus-based filtering becomes overly conservative and removes useful candidates on HalfCheetah. Despite this exception, the overall improvement indicates that larger ensembles are generally beneficial. In the main experiments, we use the same $M = 5$ across all tasks without environment-specific tuning, as a practical compromise between average performance and computational cost.

Table 6: Ablation study on ensemble size. For each VIMLE variant, results are reported as the mean and standard deviation over 10 evaluations across 5 random seeds. The best in each row are shown in bold.

Environment	VIMLE-1	VIMLE-3	VIMLE-5
halfcheetah-medium-expert-v2	102.75 ± 0.59	$\mathbf{104.28 \pm 0.75}$	100.43 ± 1.90
halfcheetah-medium-replay-v2	67.70 ± 0.25	$\mathbf{69.06 \pm 0.32}$	57.58 ± 0.38
halfcheetah-medium-v2	$\mathbf{73.43 \pm 0.10}$	72.88 ± 0.23	60.19 ± 0.21
hopper-medium-expert-v2	102.18 ± 1.58	104.92 ± 0.61	$\mathbf{112.18 \pm 0.26}$
hopper-medium-replay-v2	$\mathbf{106.43 \pm 5.52}$	97.31 ± 12.82	102.90 ± 14.52
hopper-medium-v2	89.99 ± 1.08	85.91 ± 0.92	$\mathbf{90.78 \pm 0.94}$
walker2d-medium-expert-v2	102.61 ± 0.15	103.26 ± 0.27	$\mathbf{109.80 \pm 0.40}$
walker2d-medium-replay-v2	66.91 ± 3.78	80.12 ± 2.17	$\mathbf{88.56 \pm 2.41}$
walker2d-medium-v2	83.71 ± 0.86	84.71 ± 0.94	$\mathbf{90.51 \pm 0.48}$
Average	88.41	89.16	**90.32**

G.2 Effect of Reference Sample Count

We additionally study the number of reference actions R used to compute the ensemble consistency score, while fixing the ensemble size to $M = 5$ and retaining the remaining settings in Table 8. Table 7 reports the results on the same nine non-expert MuJoCo tasks used in Table 6. Reducing R from 32 to 16 lowers the average score from 90.32 to 83.64. Increasing R to 64 yields only a marginal average improvement to 90.73 while doubling the number of reference actions. We therefore use $R = 32$ as a practical balance between performance and inference cost.

Table 7: Ablation study on the number of reference actions. Results are reported as the mean $\pm$ standard deviation over 10 evaluations across 5 random seeds. The best result in each row is shown in bold.

Environment	$R = 16$	$R = 32$	$R = 64$
halfcheetah-medium-expert-v2	94.20 ± 2.10	100.43 ± 1.90	$\mathbf{101.60 \pm 1.65}$
halfcheetah-medium-replay-v2	50.80 ± 0.55	57.58 ± 0.38	$\mathbf{58.80 \pm 0.42}$
halfcheetah-medium-v2	53.41 ± 0.42	60.19 ± 0.21	$\mathbf{61.00 \pm 0.31}$
hopper-medium-expert-v2	109.50 ± 0.35	112.18 ± 0.26	$\mathbf{112.30 \pm 0.28}$
hopper-medium-replay-v2	88.00 ± 12.80	102.90 ± 14.52	$\mathbf{103.80 \pm 12.10}$
hopper-medium-v2	84.20 ± 1.25	90.78 ± 0.94	$\mathbf{91.70 \pm 0.89}$
walker2d-medium-expert-v2	105.10 ± 0.55	109.80 ± 0.40	$\mathbf{110.20 \pm 0.43}$
walker2d-medium-replay-v2	81.00 ± 2.66	88.56 ± 2.41	$\mathbf{89.40 \pm 2.15}$
walker2d-medium-v2	86.52 ± 0.47	$\mathbf{90.51 \pm 0.48}$	87.78 ± 1.00
Average	83.64	90.32	**90.73**

H Detailed Model Hyperparameters

Table 8 summarizes the main hyperparameters used in VIMLE across all tasks, including the IMLE generator, value learning, ensemble-based action selection, and common training settings.

Table 8: Main hyperparameters of VIMLE.

Module	Name	Value
IMLE generator	Batch size	256
	Learning rate	3×10^{-4}
	Number of layers	3
	Hidden dimension	256
	Latent dimension	16
Value learning	Learning rate	3×10^{-4}
	Expectile τ_{exp}	0.9
	Target update rate τ_{target}	0.005
	Training steps	1×10^{6}
Ensemble selection	Ensemble size	5
	IMLE candidates	3
	Action candidates	64
	Reference samples	32
	Retained actions	16
Common	Discount factor γ	0.99
	Behavior training steps	1×10^{6}
	Evaluation episodes	10