VALUE GRADIENT FLOW: BEHAVIOR-REGULARIZED RL WITHOUT REGULARIZATION

Anonymous authors
Paper under double-blind review

ABSTRACT

We study behavior-regularized reinforcement learning (RL), which encompasses offline RL and RL from human feedback (RLHF). In both settings, regularization toward a reference distribution (offline data in offline RL or the supervised-finetuned policy in RLHF) is essential to prevent value over-optimization caused by erroneous out-of-distribution extrapolation. Existing methods typically add distance or divergence penalties on the learning objective, which introduces optimization challenges and over-conservatism. In this paper, we propose Value Gradient Flow (VGF), a new paradigm for behavior-regularized RL. VGF formulates an optimal transport problem from the reference distribution to the optimal policy distribution induced by the value function. This problem is solved via discrete gradient flow, where value gradients guide particles sampled from the reference distribution. Our theoretical analysis shows that an implicit behavior regularization is imposed by controlling the transport budget. This formulation avoids unnecessary restrictions on the optimization problem, enabling better reward maximization. Moreover, VGF operates without explicit policy parameterization while remaining expressive and flexible, allowing adaptively test-time scaling by adjusting the transport budget. Extensive experiments demonstrate that VGF significantly outperforms prior methods, achieving state-of-the-art results on offline RL benchmarks (D4RL, OGBench) and challenging RLHF tasks.

1 INTRODUCTION

Reinforcement learning (RL) has provided a powerful framework for solving sequential decision-making problems in complex environments. These methods have been successfully applied in diverse domains, ranging from robotics (Levine et al., 2016) to game playing (Mnih et al., 2013; Silver et al., 2017), and have recently become instrumental in fine-tuning large language models (LLMs) to align with human preferences and instructions (Ouyang et al., 2022) and enhancing the reasoning capabilities of LLMs (Shao et al., 2024; DeepSeek-AI et al., 2025). While these successes highlight the broad potential of RL, they also expose a key challenge: policies must often be regularized toward a reference distribution to remain stable and reliable. This challenge arises in both offline RL, where erroneous extrapolation beyond fixed datasets can cause severe value overestimation, and RLHF, where deviating too far from the supervised policy risks reward hacking. In these settings, naïve value maximization alone is insufficient. As a result, recent research in both offline RL (Kumar et al., 2020; Fujimoto et al., 2019; Wu et al., 2019; Xu et al., 2023) and RLHF (Ouyang et al., 2022; Wang et al., 2024) has increasingly converged on the paradigm of *behavior-regularized RL*, which balances value maximization with adherence to reliable reference distributions.

The most common approach to behavior-regularized RL is to add explicit divergence or distance penalties (e.g., KL divergence) to the RL learning objective (Ouyang et al., 2022; Touvron et al., 2023; Gao et al., 2023). While this constrains policies to remain close to the reference distribution, it also introduces several limitations. First, jointly optimizing for reward maximization and distributional proximity creates an optimization conflict, often leading to unstable training and overly conservative policies (e.g., stay within the support of the reference distribution (Wu et al., 2025; Korbak et al., 2022)). Second, selecting an appropriate penalty strength is difficult: too strong a constraint prevents full exploration of in-distribution regions, while too weak a constraint risks out-of-distribution overestimation (Moskovitz et al., 2024). Alternative strategies, such as maximum reward clipping (Moskovitz et al., 2024), uncertainty quantification (Zhang et al., 2024), or ensemble-based estimates (Coste et al., 2023), bring their own drawbacks, including poor generalization to OOD regions (Nalisnick et al., 2018) and high computational cost (Eisenstein et al., 2023). More broadly, mechanisms designed to guard against OOD extrapolation often also suppress in-distribution actions, limiting the policy's ability to fully exploit the value function. In fact, we argue that any explicit distance or divergence constraint places unnecessary restrictions on the optimization problem and the resulting policy, motivating the search for a more flexible and principled alternative paradigm.

Behavior-regularized RL without regularization.[1] We introduce **Value Gradient Flow (VGF)**, which casts the behavior-regularized RL problem as the optimal transport from an (estimated) reference distribution to the optimal policy distribution induced by the value function. Instead of adding an explicit distance/divergence penalty or maintaining a separate parameterized policy, VGF starts from samples of the reference distribution and gradually nudges them toward higher value regions using a small, fixed number of guidance steps. The distribution after these steps serves as an implicit policy. The transport budget itself (i.e., how far and how often we move) acts as an implicit behavior regularization, limiting deviation from the reference distribution during training while preserving flexibility to enable adaptively scaling at inference. VGF removes the need to balance the optimization conflict between reward maximization and derivation penalties, avoids brittle coefficient hyperparameter tuning, and better preserves multimodal structure inherited from the reference distribution. In RLHF, VGF yields inference-time control that steers a supervised finetuned policy using first-order value gradient, sidestepping RL-style optimization and reducing both compute and engineering complexity. Across extensive experiments, VGF consistently outperforms strong behavior-regularized baselines, attaining state-of-the-art results on standard offline RL suites (D4RL, OGBench) and delivering substantial gains on challenging RLHF tasks.

To summarize, the contributions of this paper are as follows:

- A new paradigm that casts behavior-regularized RL as bounded transport from a reference distribution to value-preferred regions, yielding an implicit policy with reduced conservatism.
- An unified and scalable framework for both offline RL and RLHF, offering several distinctive benefits and achieving state-of-the-art empirical performance.

2 PRELIMINARIES

MDP and value functions. We consider the RL problem presented as a Markov Decision Process (MDP) (Sutton et al., 1998), which is specified by a tuple $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \mathcal{P}, d_0, r, \gamma \rangle$. Here $\mathcal{S}$ and $\mathcal{A}$ are state and action space, $\mathcal{P}(s'|s,a)$ and d_0 denote transition dynamics and initial state distribution, $r(s,a)$ and γ represent reward function and discount factor, respectively. The goal of RL is to find a policy $\pi(a|s)$ which maximizes expected return $J(\pi) = \mathbb{E}_\pi[\sum_{t=0}^{\infty} \gamma^t \cdot r(s_t, a_t)]$. In the offline setting, interaction with the environment is prohibited and one needs to learn an optimal π from a static replay buffer $\mathcal{D} = \{s_i, a_i, r_i, s'_i\}_{i=1}^N$ collected from unknown policies. The dataset can be heterogeneous and suboptimal, we denote the empirical behavior policy of $\mathcal{D}$ as $\pi_\mathcal{D}$, which represents the conditional distribution $p(a|s)$ observed in the dataset.

RL methods based on approximate dynamic programming typically maintain an action-value function (Q-function) and, optionally, a state-value function (V-function), referred to as $Q(s,a)$ and $V(s)$ respectively (Haarnoja et al., 2017; Nachum et al., 2017; Kumar et al., 2020; Kostrikov et al., 2021b). Define $Q^\pi : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$, where $Q^\pi(s,a) = \mathbb{E}_\pi\left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) | s_0 = s, a_0 = a\right]$. The value function is learned by satisfying single-step Bellman consistencies. Let $\mathcal{T}^\pi$ be the Bellman operator with policy π such that $(\mathcal{T}^\pi Q)(s,a) := r(s,a) + \gamma \mathbb{E}_{s'|s,a} \mathbb{E}_{a' \sim \pi}\left[Q(s',a')\right]$. Then Q are learned by $\min_Q J(Q) = \frac{1}{2}\mathbb{E}_{(s,a)\sim\mathcal{D}}\left[(\mathcal{T}^\pi Q - Q)(s,a)^2\right]$.

Behavior-regularized RL. In general, behavior-regularized RL considers the following constraint optimization problem with a reference policy μ:

$$\pi^* = \arg\max_\pi \; \mathbb{E}_{s\sim\mathcal{D}, a\sim\pi(\cdot|s)}\big[R(s,a)\big] \;\; \text{s.t.} \;\; \mathbb{E}_{s\sim\mathcal{D}}\big[M\left(\pi(\cdot|s), \mu(\cdot|s)\right)\big] \le \epsilon, \tag{1}$$

where $R(s,a)$ is a differentiable function and M is some distance or divergence measure (e.g., KL-divergence, L_2 distance). In offline RL we choose μ as $\pi_\mathcal{D}$ and $R(s,a)$ to be the Q-function,

[1] Although VGF uses transport budget as an implicit regularization, here we use "without regularization" to emphassize that VGF doesn't involve any explicit regularization during optimization.

whereas in RLHF μ is the LM after supervised finetuning and $R(s,a)$ is set to be the reward model trained on pairwise preference data (Bradley and Terry, 1952). Depending on π and M, several different approaches can be used to extract the optimal policy based on Equation (1).

(1) Policy gradient with π reparameterized. The most straightforward approach is to guide the policy to directly maximize function R with reparameterized gradients. The constraint term will be added as a regularization term with a coefficient β to balance these two gradients.

$$\max_{\pi} \mathbb{E}_{s\sim\mathcal{D}}\Big[\mathbb{E}_{a\sim\pi}\big[R(s,a)\big] - \beta\cdot M\big(\pi(\cdot|s),\mu(\cdot|s)\big)\Big]. \tag{2}$$

Reparameterized policy gradient is commonly used in offline RL with Gaussian policies (Wu et al., 2019; Fujimoto and Gu, 2021; Tarasov et al., 2024) and has recently been extended to more expressive policy classes like Diffusion (Wang et al., 2023) and Flow models (Park et al., 2025b), with M being the Diffusion or Flow matching loss. One issue of this type of method is that it uses first-order gradient information from $R(s,a)$ where actions are sampled from π, this brings over-estimation errors to the policy update without an appropriately selected β, and further affects value learning due to the *deadly-triad* issue (Van Hasselt et al., 2018). Another issue is that computing $\nabla_\theta\, \mathbb{E}_{a\sim\pi_\theta}\left[R(s,a)\right]$ is unstable, requiring backpropagation through time when π is a generative model, and cannot be applied to the LLM setting due to the discrete nature of language generation (Rafailov et al., 2023).

(2) Weighted behavior cloning or best-of-N with $M = \mathrm{KL}$. To consider the LLM RL setting, we choose M to be the KL divergence. Using KL-divergence gives Equation (1) a closed-form solution which can be optimized by doing weighted behavior cloning (BC) (Peng et al., 2019; Xu et al., 2023) where actions are sampled from the reference policy, as follows.

$$\max_{\pi} \mathbb{E}_{s\sim\mathcal{D},a\sim\mu}\big[\exp(R(s,a)/\beta)\cdot\log\pi(a|s)\big]. \tag{3}$$

Although simple and easy to implement, using weighted BC tends to be mode-covering, only amplifying weak signals from the reference distribution without extracting new skills or knowledge (Wu et al., 2025). In fact, a simple best-of-N sampling policy (Nakano et al., 2021), where N i.i.d. samples are drawn from the reference distribution and one with the highest $R(s,a)$ is returned, is theoretically near optimal for this KL-constrained RL problem (Beirami et al., 2024; Yang et al., 2024).

$$\pi^* = \arg\max_{a_i\sim\mu, i\in[N]} R(s,a_i). \tag{4}$$

3 Value Gradient Flow

VGF is designed to solve the above-mentioned challenges and provide a **scalable and unified** solution to Equation (1), and we give a detailed introduction in this section.

3.1 Behavior-Regularized RL as Optimal Transport

We first consider a surrogate optimization objective that augments the value maximization objective in Equation (1) with a policy entropy maximization term: $\mathbb{E}_{a\sim\pi}\left[R(s,a)\right] + \alpha H(\pi(\cdot|s))$, where $H(\pi(\cdot|s)) \triangleq \mathbb{E}_\pi\left[-\log\pi(a|s)\right]$ is the causal entropy of the policy π at state s. This Maximum-Entropy (MaxEnt) formulation of RL is well-known to enhance the exploration and robustness of the policy (Haarnoja et al., 2018; Garg et al., 2021; Eysenbach and Levine, 2021). However, our intuition here is that optimizing this MaxEnt objective turns the optimal policy distribution from greedy max to softmax over the whole action space, resulting in a variational distribution as the Boltzmann distribution over the value function $R(s,a)$ (Ziebart, 2010; Bloem and Bambos, 2014):

$$\pi_R^*(a|s) = \frac{1}{Z_s}\exp\left(R(s,a)/\alpha\right), \tag{5}$$

where Z_s is the normalization factor given as $\sum_{a'}\exp\left(R\left(s,a'\right)/\alpha\right)$.

Particle-based gradient flow. We reframe the value-maximization problem as an optimal transport problem that transports probability mass from distribution μ to distribution π_R^* defined in Equation (5). A natural way to formalize this transport is as a gradient flow of the functional $F(q) = \mathrm{KL}(q \,\|\, \pi_R^\star)$ on the space of probability measures endowed with the Wasserstein metric (Jordan et al., 1998; Ambrosio et al., 2008; Peyré and Cuturi, 2019). The resulting continuous-time evolution q_t follows the continuity equation $\partial_t q_t + \nabla \cdot (q_t v_t) = 0$ with the steepest-descent velocity field $v_t = \nabla \log \pi_R^\star - \nabla \log q_t$, so that $F(q_t)$ decreases monotonically and the stationary distribution is $\pi_R^\star$. However, directly solving q_t is intractable, so we adopt the Jordan-Kinderlehrer-Otto (JKO) minimizing-movement scheme (Jordan et al., 1998) to obtain a discrete gradient flow:

$$q_{k+1} = \arg\min_q \ \mathrm{KL}(q \,\|\, \pi_R^\star) + \frac{1}{2h} W_2^2(q,\, q_k)\,, \tag{6}$$

where $h > 0$ is the step size and W_2 is the 2-Wasserstein distance (Peyré and Cuturi, 2019). In Euclidean space, Equation (6) reduces to gradient descent on the function landscape. However, it is intractable as, in general, q_k is infinite-dimensional.

To obtain a practical solver, we approximate q_k by an empirical measure over N particles in action space (for a fixed state s), $q_k \approx \frac{1}{N}\sum_{i=1}^N \delta_{a_i^{(k)}}$, and seek an update rule for $\{a_i^{(k)}\}_{i=1}^N$ that decreases Equation (6). By restricting the velocity field v to the unit ball of a vector-valued reproducing kernel Hilbert space (RKHS), we get a solver that can be derived as the nonparametric functional gradient method that most rapidly decreases $\mathrm{KL}(q \,\|\, \pi_R^\star)$ within the RKHS (Liu and Wang, 2016; Liu, 2017). This yields a particle-based gradient flow solver that approximates the discrete gradient flow as

$$a_i^{(l+1)} = a_i^{(l)} + \epsilon \cdot \phi\big(a_i^{(l)}\big), \quad \phi(x) \;=\; \frac{1}{N}\sum_{j=1}^N \Big[k(a_j, x)\, \underbrace{\nabla_{a_j} \log \pi_R^\star(a_j|s)}_{=\nabla_{a_j} R(s,a_j)/\alpha} + \nabla_{a_j} k(a_j, x) \Big]. \tag{7}$$

Here, ϵ is the step size and $k(\cdot,\cdot)$ is the kernel function (e.g., RBF kernel with $k(a_i, a_j) = \exp(||a_i - a_j||^2/2\sigma^2)$. The first term in $\phi(a_i)$ drives the particles toward the high probability regions of π_R^* (i.e., with high $R(s,a)$), while the second term serves as a repulsive force to encourage dispersion and preserve multi-modality of the particles. The temperature α controls the relative strength of these two terms. Convergence of this particle-based gradient flow to the target distribution can be easily proved as $\mathbb{E}_{a^L \sim \pi_R^*(\cdot|s)}[\phi(a^L)] = 0$.

Note that an **implicit behavior regularization** is imposed via controlling the transport budget (L, α and ϵ). Intuitively, Equation (7) performs a kernel-smoothed transport to each particle in the action space, resulting in a controlled derivation from the reference distribution. Theoretically, we show in the following that the Maximum Mean Discrepancy (MMD) distance between the initial particles sampled from the reference policy and the particles generated by VGF is bounded.

Theorem 1. *Assume the value function $R(s,a)$ is c-Lipschitz w.r.t the input action a. Define the implicit policy that performs Equation (7) for L steps with N particles as π_N^L. We have*

$$\mathrm{MMD}^2(\mu, \pi_N^L) = \mathrm{MMD}^2(\pi_N^0, \pi_N^L) \le \frac{2\epsilon L}{\sigma\sqrt{e}} \left(\frac{c}{\alpha} + \frac{1}{\sigma\sqrt{e}} \right).$$

VGF in the LLM setting. In the LLM RL setting, at time step t, the action a_t is a discrete token and the state s_t is the token sequence $s_t = (x_0, \ldots, x_L, a_0, \ldots, a_{t-1})$, where $x = (x_0, \ldots, x_L)$ is the input prompt and $y = (a_0, \ldots, a_{t-1})$ are the generated tokens up to step $t-1$. The transition function P updates the state deterministically via concatenation: $s_{t+1} = P(s_t, a_t) = s_t \,\|\, a_t$.

A direct application of Eq. (7) to tokens is infeasible because tokens are discrete. We therefore perform VGF in a **continuous surrogate space** and decode back to the discrete token space only at the end of the gradient flow. Let u be a differentiable representation of a full response y. The representation could either be the token-embedding matrix $u \in \mathbb{R}^{T\times d}$ or a latent vector $u = z \in \mathbb{R}^m$ of a flow or diffusion language model with $y = \mathrm{Dec}(z)$. Denote $y_i^{(l)} = \mathrm{Dec}(u_i^{(l)})$. Because the reward model is differentiable with respect to its input embeddings, response-level gradient $\nabla_y R(x,y)$ can be back-propagated to the surrogate via the chain rule as follows.

$$\nabla_{u_i} \log \pi_R^*\big(y_i^{(l)}|x\big) = \frac{1}{\alpha}\, J_i^\top\, \nabla_y R\big(x, y_i^{(l)}\big), \quad J_i := \frac{\partial \mathrm{Dec}\big(u_i^{(l)}\big)}{\partial u_i^{(l)}}. \tag{8}$$

One motivation to use VGF is that the SFT policy is far from random, with most probability mass concentrating on a small subset of tokens and modes. Note that VGF utilizes first-order gradient

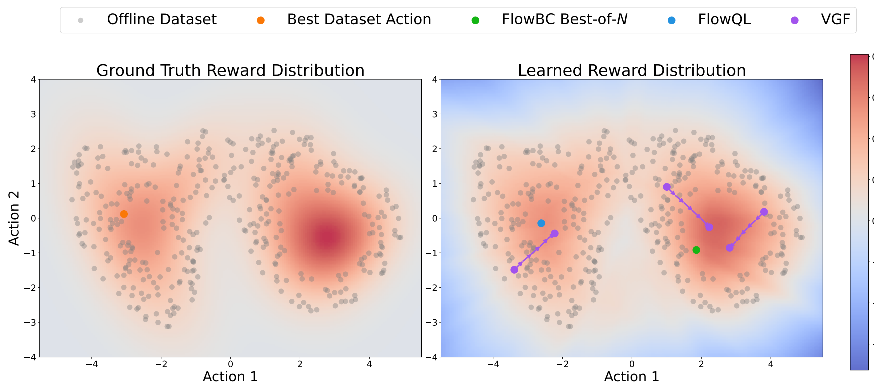

Figure 1: **Toycase results**. VGF generates actions with higher ground-truth reward than other methods.

guidance from R, this avoids high-variance PPO-style optimization (Ouyang et al., 2022) and enables inference-only control similar to best-of-N sampling. However, the difference is that VGF steers particles toward high-reward modes, the resulting implicit policy need not remain within the support of the reference distribution, as shown in the following theorem.

Theorem 2. *Define the ϵ-support of a distribution P as* $\mathrm{supp}_\epsilon(P) := \{x : p(x) \geq \epsilon\}$. *We have*

$$\mathrm{supp}_\epsilon(\pi_N^L(\cdot|s)) \not\subseteq \mathrm{supp}_\epsilon(\mu(\cdot|s)).$$

This indicates that, unlike the methods discussed in Section 2 which result in $\mathrm{supp}_\epsilon(\pi(\cdot|s)) \subseteq \mathrm{supp}_\epsilon(\mu(\cdot|s))$ (Wu et al., 2025), VGF breaks the over-conservative behavior constraint, enabling the discovery and exploitation of novel behaviors beyond the reference distribution.

3.2 Discussion

Below, we outline several distinctive advantages of VGF and discuss the connection with prior work.

(1) Stable optimization with reduced conservatism. Compared with methods based on reparameterized policy gradient (Fujimoto and Gu, 2021; Wang et al., 2023; Park et al., 2025b), VGF removes the need to balance the optimization conflict between reward maximization and derivation penalties. Instead, VGF is optimized to find the best reward-maximization policy within a fixed behavior constraint, which is more aligned with Equation (1). Furthermore, since there is no function approximation error introduced by the policy parameterization, VGF alleviates the deadly-triad issue (Van Hasselt et al., 2018) in actor-critic style RL algorithms, enjoying more stable training.

(2) Implicit policy with multimodal expressivity. While bypassing explicit policy parameterization, the implicit policy in VGF is still expressive enough to capture a multimodal distribution. Owing to the usage of gradient flow, VGF naturally preserves and sharpens multiple high-value modes from the reference distribution instead of collapsing to a single one. This is different from BCQ (Fujimoto et al., 2019) where a Gaussian residual policy with limited expressivity is learned on top of the reference policy. Note that in the offline RL setting, VGF remains versatile to the usage of different advanced generative models, e.g., Diffusion models (Song et al., 2021) or Flow models (Lipman et al., 2022), to generate samples from the reference distribution given only an offline dataset.

(3) Adaptive scaling during test-time. One intriguing property of VGF is that it enables adaptive test-time scaling via varying the transport budget **without** any retraining. For example, when the value function R can generalize well, the performance will scale with the number of test-time flow steps, which could be different from the number of train-time flow steps. However, when the value function has large extrapolation errors, by setting the test-time flow steps to 0, VGF reduces to Best-of-N sampling methods (Chen et al., 2023; Hansen-Estruch et al., 2023). One difference in this case is that VGF learns the value function by TD learning (since the train-time flow step is not 0) instead of in-sample learning (Kostrikov et al., 2021b; Xu et al., 2023). We find in practice that TD learning enables better stitching and generalization. This difference makes VGF **fundamentally different** from Diffusion-based methods (Mao et al., 2024; Frans et al., 2025) that can also do adaptive generation via adjusting the guidance weight but rely on in-sample value learning.

Practical consideration. In offline RL, given the offline dataset, we train BC flow models to generate samples from μ. The Q-function is updated using TD-learning with the double Q-learning trick (Fujimoto et al., 2018) to enhance training stability, and we average over all particles when computing the target Q-values. At test time, since we need to choose one particle to do evaluation, we use best-of-N sampling from all VGF particles based on the value function. We summarize the pseudo-code in Algorithm 1.

Algorithm 1 Value Gradient Flow (VGF)

function VGF($s, \hat{\mu}, R, L_{\text{test}}$)
  Get $a_N^0 \sim \hat{\mu}(\cdot|s)$
  for $l = 0, 1, \ldots, L_{\text{test}} - 1$ **do**
    Get a_N^{l+1} using R and a_N^l by Eq. (7)
  return $a_N^{L_{\text{test}}}$

Require: $\mathcal{D}$, L_{train}, L_{test}, α, ϵ
▷ Value Training (offline RL)
for $t = 1, 2, \cdots, M$ **do**
  Sample transitions $(s, a, r, s') \sim \mathcal{D}$
  Train BC flow model $\hat{\mu}$
  Get $a_N^{L_{\text{train}}} = \text{VGF}(s', \hat{\mu}, Q, L_{\text{train}})$
  Train Q using $a_N^{L_{\text{train}}}$ by TD learning
▷ Evaluation (offline RL and RLHF)
Get initial state s, set d as False
while not d **do**
  Get $a_N^{L_{\text{test}}} = \text{VGF}(s, \hat{\mu}, R, L_{\text{test}})$
  Get best-of-N a^* from $a_N^{L_{\text{test}}}$ by Eq. (4)
  Roll out a^* and get (s', r, d)
  Set $s \leftarrow s'$

A toy example. We use a toy example to illustrate the mechanism of VGF. We construct a 2-D continuous control bandit task with a bimodal ground-truth reward distribution, where the offline dataset is generated from sampling from sub-optimal reward regions, as demonstrated in Figure 1. We are interested in studying the behavior of the following three different behavior-regularized RL methods. Note that all three methods fit a learned reward model using L_2 loss and a BC flow model using flow-matching loss. **FlowQL** (Park et al., 2025b): This method represents the first group of methods in section 2. FlowQL additionally trains a one-step flow model as the policy, which is used during evaluation. We carefully tune the coefficient β to ensure the best performance. **FlowBC Best-of-N**: This method represents the second group of methods in section 2. In this case, we sample $N = 20$ actions from the BC flow model and do best-of-N sampling using the learned reward function. **VGF**: For this task, we set particle number $N = 3$, $L_{\text{test}} = 5$ and $\alpha = 0.1$.

We plot the action generated by FlowQL and FlowBC Best-of-N, along with the flow trajectories of particles generated by VGF, in Figure 1. As shown, even if the learned reward model gets some error, the implicit policy in VGF demonstrates successful and effective exploration of the area with high ground-truth reward. By contrast, FlowQL is shown to be misled by the error of the learned reward model, generating actions with suboptimal values. We conjecture that this is because the actor loss of FlowQL consists of both one-step distillation loss and reward-maximization loss, and jointly optimizing these losses leads to erroneous generalization when both the behavior is suboptimal and the learned reward function is inaccurate. Although best-of-N sampling from the FlowBC model could improve the best dataset action, it still falls within the support of the suboptimal behavior distribution due to over-conservatism, which aligns well with the theoretical analysis.

4 Related Work

Offline RL. To address distributional shift, most model-free offline RL methods augment off-policy learning with behavior regularization. This can appear explicitly as divergence penalties that constrain the learned policy toward the dataset distribution (Wu et al., 2019; Kumar et al., 2019; Fujimoto and Gu, 2021), implicitly via weighted behavior cloning and advantage-weighted updates (Wang et al., 2020; Nair et al., 2020), or through careful policy parameterization that restricts actions to data-supported regions (Fujimoto et al., 2019; Zhou et al., 2020). A complementary line modifies value learning to penalize OOD actions and promote pessimism (Kumar et al., 2020; Kostrikov et al., 2021a), with related approaches leveraging uncertainty estimates or learned distance functions to discourage extrapolation (An et al., 2021; Li et al., 2022). To improve the expressiveness of the policy, recent work adopts powerful generative models as the policy: Decision Transformer (Chen et al., 2021); diffusion-based policies such as DQL, SfBC, QGPO, and IDQL (Wang et al., 2023; Chen et al., 2023; Hansen-Estruch et al., 2023); as well as CVAE- and consistency-model policies (Zhou et al., 2020; Ding and Jin, 2023). Compared with these methods, VGF bypasses explicit policy parameterization, greatly reducing the training cost while remaining expressive and multimodal.

Reinforcement Learning from Human Feedback. In RLHF, policy models can exploit imperfections in learned reward models, a phenomenon often termed reward over-optimization (Gao et al., 2023) and also discussed as reward hacking or reward gaming (Amodei et al., 2016; Skalse et al.,

Table 1: **D4RL offline RL results**. Scores are averaged over the final 10 evaluations across 5 seeds with standard deviation reported, we highlight the best score in integer-level. VGF demonstrates superior performance on most tasks, especially those challenging ones.

	Gaussian Policy			Diffusion/Flow Policy			w/o Policy
Dataset	TD3+BC	IQL	IVR	Diffusion-QL	SfBC	FQL	VGF (ours)
halfcheetah-m	48.3	47.4 ±0.2	48.3 ±0.2	51.1 ±0.5	45.9 ±2.2	55.6 ±0.2	57.1 ±0.1
hopper-m	59.3	66.3 ±5.7	75.5 ±3.4	90.5 ±4.6	57.1 ±4.1	60.6 ±0.1	97.9 ±2.0
walker2d-m	83.7	72.5 ±8.7	84.2 ±4.6	87.0 ±0.9	77.9 ±2.5	65.9 ±0.3	89.4 ±0.8
halfcheetah-m-r	44.6	44.2 ±1.2	44.8 ±0.7	47.8 ±0.3	37.1 ±1.7	48.3 ±0.3	49.1 ±0.1
hopper-m-r	60.9	95.2 ±8.6	99.7 ±3.3	101.3 ±0.6	86.2 ±9.1	50.7 ±2.7	99.0 ±1.1
walker2d-m-r	81.8	76.1 ±7.3	81.2 ±3.8	95.5 ±1.5	65.1 ±5.6	38.8 ±1.1	97.8 ±1.6
halfcheetah-m-e	90.7	86.7 ±5.3	94.0 ±0.4	96.8 ±0.3	92.6 ±0.5	102.1 ±0.6	99.1 ±0.3
hopper-m-e	98.0	101.5 ±7.3	111.8 ±2.2	111.1 ±1.3	108.6 ±2.1	76.7 ±0.6	98.3 ±3.3
walker2d-m-e	110.1	110.6 ±1.0	110.0 ±0.8	110.1 ±0.3	109.8 ±0.2	102.6 ±0.2	110.5 ±1.5
antmaze-u	78.6	85.5 ±1.9	92.2 ±1.4	93.4 ±3.4	92.0 ±2.1	96 ±1.6	98.0 ±1.8
antmaze-u-d	71.4	66.7 ±4.0	74.0 ±2.3	66.2 ±8.6	85.3 ±3.6	89 ±2.3	94.3 ±1.4
antmaze-m-p	10.6	72.2 ±5.3	80.2 ±3.7	76.6 ±10.8	81.3 ±2.6	78.0 ±2.6	92.4 ±3.1
antmaze-m-d	3.0	71.0 ±3.2	79.1 ±4.2	78.6 ±10.3	82.0 ±3.1	71.0 ±3.4	93.7 ±2.8
antmaze-l-p	0.2	39.6 ±4.5	53.2 ±4.8	46.4 ±8.3	59.3 ±14.3	84.0 ±2.9	82.5 ±3.6
antmaze-l-d	0.0	47.5 ±4.4	52.3 ±5.2	56.6 ±7.6	45.5 ±6.6	83.0 ±3.8	83.8 ±4.5

2022; Pang et al., 2023). Many studies analyze this effect in synthetic setups that substitute expensive human evaluation with strong "gold" models for labeling and assessment (Gao et al., 2023; Moskovitz et al., 2024; Coste et al., 2023). A prevailing mitigation strategy augments the reward or training objective with a KL penalty to a supervised-finetuned reference model (Kullback and Leibler, 1951; Stiennon et al., 2020; Ouyang et al., 2022; Bai et al., 2022). Other approaches employ ensembles or early-stopping-style constraints to curb over-optimization while controlling KL (Coste et al., 2023; Moskovitz et al., 2024). However, explicit penalties inevitably introduce a reward-KL trade-off that is sensitive to coefficient tuning and can be overly conservative towards the reference support.

Optimal transport in RL. Optimal transport (OT) provides a geometry over distributions that has proved useful in multiple RL settings. In distributional RL, Wasserstein metrics underpin return-distribution learning via quantile-regression objectives, improving stability and control (Dabney et al., 2018). OT has also been used to align occupancy or trajectory distributions for imitation and offline learning. For example, Sinkhorn-based matching or primal Wasserstein formulations that shape rewards and enable cross-domain alignment (Dadashi et al., 2021). Beyond matching, an OT viewpoint motivates transporting probability mass toward value-preferred regions, inspiring flow/particle-style policy updates and robust formulations that explicitly constrain distributional shift. Our work follows this trajectory but emphasizes transport from the reference distribution to the optimal policy distribution using value gradients. This yields an implicit policy and turns the transport budget itself into behavior regularization without tightly coupled policy-optimization loops (Peyré and Cuturi, 2019). Other work like PPL (Asadulaev et al., 2024) considers transport between states and partial action distributions, whereas VGF operates in the action space.

5 Experiments

The goal of our experiments is to evaluate the efficacy of VGF in improving offline RL and RLHF. We evaluate the performance of VGF on D4RL and OGBench and compare it with prior methods. We also provide an ablation study on important hyperparameters and investigate the adaptive scaling behavior during test time in VGF to gain a deeper understanding of its mechanism.

5.1 Offline RL Results

D4RL Benchmark Datasets. We evaluate the performance of VGF on the D4RL benchmark (Fu et al., 2020), and compare it with several algorithms based on Gaussian policy, diffusion policy and flow policy. Gaussian-policy-based baselines include TD3+BC (Fujimoto and Gu, 2021), IQL (Kostrikov et al., 2021a), and IVR (Xu et al., 2023). We also select Diffusion-QL (Wang et al., 2023) and SfBC (Chen et al., 2023) as diffusion-policy-based baselines, and FQL (Park et al., 2025b) as a flow-policy-based baseline. The evaluation tasks include MuJoCo, a set of locomotion tasks, and

Table 2: **OGBench offline RL results**. Scores are averaged over the final 10 evaluations across 5 seeds with standard deviation reported, we highlight the best score in integer-level. VGF achieves competitive or superior performance compared to prior approaches, especially on hard tasks.

	Gaussian Policy			Diffusion/Flow Policy			w/o Policy
Dataset (5 tasks each)	BC	IQL	ReBRAC	FBRAC	IDQL	FQL	VGF (ours)
antmaze-giant	0 ± 0	4 ± 1	26 ± 8	4 ± 4	0 ± 0	9 ± 6	3 ± 1
humanoidmaze-medium	2 ± 1	33 ± 2	22 ± 8	38 ± 5	1 ± 0	58 ± 5	72 ± 1
humanoidmaze-large	1 ± 0	2 ± 1	2 ± 1	2 ± 0	1 ± 0	4 ± 2	10 ± 2
antsoccer-arena	1 ± 0	8 ± 2	0 ± 0	16 ± 1	12 ± 4	60 ± 2	63 ± 4
cube-single	5 ± 1	83 ± 3	91 ± 2	79 ± 7	95 ± 2	96 ± 1	96 ± 1
cube-double	2 ± 1	7 ± 1	12 ± 1	15 ± 3	15 ± 6	29 ± 2	61 ± 8
scene	5 ± 1	28 ± 1	41 ± 3	45 ± 5	46 ± 3	56 ± 2	59 ± 1
puzzle-3x3	2 ± 0	9 ± 1	21 ± 1	14 ± 4	10 ± 2	30 ± 1	77 ± 4
puzzle-4x4	0 ± 0	7 ± 1	14 ± 1	13 ± 1	29 ± 3	17 ± 2	46 ± 4

AntMaze, a series of navigation tasks. The results in Table 1 show that VGF outperforms all of the baselines in most datasets. We would like to highlight that VGF achieves much higher scores than prior methods in AntMaze tasks that are widely acknowledged to be more challenging.

OGBench Datasets. Another benchmark that we use to evaluate is OGBench (Park et al., 2025a), which provides a variety of goal-conditioned tasks and datasets across robotic locomotion and manipulation. Among the reward-based single-task settings, we select 4 locomotion and 5 manipulation environments, each of which consists of 5 tasks. We take the average score of the 5 tasks to be the indicator of performance in these environments. We compare our results with several of the state-of-the-art algorithms, including ReBRAC (Tarasov et al., 2023), Flow BRAC (Wu et al., 2019), IDQL (Hansen-Estruch et al., 2023) and FQL (Park et al., 2025b). ReBRAC leverages a monolithic Q network and a Gaussian policy and achieves competitive performance on offline RL datasets. Flow BRAC replaces Gaussian policy with flow policy in behavior-regularized actor-critic algorithms. IDQL trains a diffusion BC model along with a value function learned by IQL to do best-of-N sampling. FQL is a recently proposed method that introduces a one-step flow policy that distills from the BC flow policy to avoid unstable gradient backpropagation through time. Table 2 summarizes the results of VGF and its comparison with baselines on OGBench. VGF achieves better performance than prior methods in most of the environments, especially those hard ones where FQL attains performance below 50% success rate (`cube-double`, `puzzle-3x3` and `puzzle-4x4`).

5.2 RLHF Results

We report results on the TL;DR Summarize (Stiennon et al., 2020) and Anthropic Helpful and Harmless Dialogue (Bai et al., 2022) datasets. The training split of `TL;DR` dataset contains 116k human-written instructions and 93k human-annotated preference pairs. The preprocessed `Anthropic-HH` dataset contains 112k training preference pairs. Both the reward model and the SFT policy are initialized from the same Pythia-2.8B base model, trained on either human demonstrations (`TL;DR`) or the chosen responses (`Anthropic-HH`). For evaluation metrics, we calculate win rates (WR%) judged by GPT-4 comparing outputs from the initialization and aligned models. As shown in Table 3, VGF outperforms all baseline RLHF methods by a large margin.

	TL;DR	Anthropic-HH
Model	WR% (vs ref)	WR% (vs chosen)
Pythia-SFT	48.5	46.2
PPO	57.3	45.5
DPO	61.2	51.5
Best-of-N	58.3	49.0
VGF (Ours)	68.1	59.0

Table 3: **RLHF results**. VGF outperforms baseline RLHF methods by having higher win-rates on TL;DR and Anthropic-HH dataset.

5.3 Understanding the Properties and Behavior of VGF

To better understand the mechanism and behavior of VGF, we investigate the importance of hyperparameter selection and the test-time scaling property by conducting some ablation study.

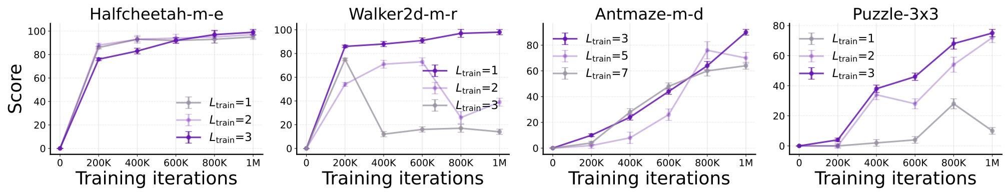

Figure 2: Ablation study on VGF train-time flow steps L_{train}.

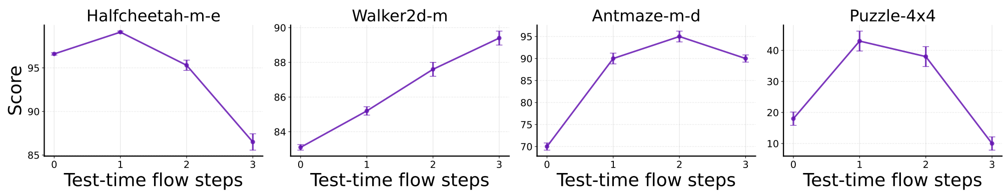

Figure 3: VGF enables adaptive test-time scaling behavior by adjusting test-time flow steps L_{test}.

What are the important hyperparameters of VGF?

There are three hyperparameters in VGF: train-time flow steps L_{train}, temperature α and particle number N, with L_{train} being the most important one. Below we provide a brief explanation for their importance and provide an ablation study on the effect of L_{train} in Figure 2. Train-time flow steps L_{train} is the number of flow steps we adopt during training. L is directly related to the distance between the reference policy and the learned policy. Intuitively, a larger L means deviating more from the reference policy. In the ablation study, we show that optimal L_{train} needs to be tuned per task to achieve the best performance. Temperature α in Equation (7) serves as the trade-off between reward maximization and dispersing particles. Specifically, a smaller α drives the actions towards the area with higher reward, but with a risk of collapsing to a single mode. The value of α is tuned for each task to seek a balance between reward maximization and dispersion. Particle number N affects the multi-modality of the implicit policy in VGF. We find that choosing a small number is enough to represent an expressive distribution. We choose $N = 5$ in our experiments and find that setting it to 10 or more has no big difference.

Can VGF enable adaptive test-time scaling behavior?

The answer is yes by controlling the test-time flow steps L_{test}. However, the optimal L_{test} depends on the specific dataset. In general, scaling up the test-time flow steps is helpful when the value function generalizes well to OOD regions and the offline dataset is of low quality, which requires improving over the reference policy to achieve the best performance. Note that even when the value function has large extrapolation error, by setting the test-time flow steps to 0, VGF reduces to the best-of-N sampling method, but it can still outperform the reference policy, enabling in-distribution generalization owing to the training of a value function via TD learning.

6 Limitations and Future Work

In this paper, we propose VGF, a new scalable paradigm that casts behavior-regularized RL as optimal transport from the reference distribution to the optimal policy distribution induced by the value function, where the transport budget serves as the implicit behavior regularization. VGF uses particle-based gradient flow as the practical solution, yielding an implicit policy with multimodal expressivity while bypassing the need of policy reparameterization. VGF is easy to implement, enabling adaptive test-time scaling and achieves strong empirical results on both offline RL and RLHF tasks. We believe that VGF represents a concrete step toward building unified and scalable behavior-regularized RL algorithms. One limitation of VGF lies in its ability to handle scenarios where the reference distribution is heavily skewed toward suboptimal behavior. Enhancing performance under such settings remains for future work.

7 Reproducibility Statement

To ensure the reproducibility of this paper, we detail the theoretical and empirical parts of our results in the main paper and the appendix. In Section 3, we introduce the basic notations used in the theoretical analysis and establish theories to support our claim. Furthermore, in Appendix A, we provide the detailed proofs of the theoretical results. For empirical details, we briefly introduce the setup of our toy case in Section 3. In Appendix B, we elaborate on the benchmark environments, the network architecture and hyperparameters in our experiments, and provide a more detailed version of empirical results on OGBench. In Appendix C, we provide a simple implementation of our method. We will release the code after acceptance.

References

Luigi Ambrosio, Nicola Gigli, and Giuseppe Savaré. Gradient Flows: In Metric Spaces and in the Space of Probability Measures. Birkhäuser, Basel, 2nd edition, 2008. doi: 10.1007/978-3-7643-8722-8.

Dario Amodei, Chris Olah, Jacob Steinhardt, Paul Christiano, John Schulman, and Dan Mané. Concrete problems in ai safety. arXiv preprint arXiv:1606.06565, 2016.

Gaon An, Seungyong Moon, Jang-Hyun Kim, and Hyun Oh Song. Uncertainty-based offline reinforcement learning with diversified q-ensemble. Proc. of NeurIPS, 2021.

Arip Asadulaev, Rostislav Korst, Aleksandr Korotin, Vage Egiazarian, Andrey Filchenkov, and Evgeny Burnaev. Rethinking optimal transport in offline reinforcement learning. Advances in Neural Information Processing Systems, 37:123592–123607, 2024.

Yuntao Bai, Saurav Kadavath, Sandipan Kundu, Amanda Askell, et al. Constitutional ai: Harmlessness from ai feedback. arXiv preprint arXiv:2212.08073, 2022.

Ahmad Beirami, Alekh Agarwal, Jonathan Berant, Alexander D'Amour, Jacob Eisenstein, Chirag Nagpal, and Ananda Theertha Suresh. Theoretical guarantees on the best-of-n alignment policy. arXiv preprint arXiv:2401.01879, 2024.

Michael Bloem and Nicholas Bambos. Infinite time horizon maximum causal entropy inverse reinforcement learning. In Proceedings of the 53rd IEEE Conference on Decision and Control (CDC), pages 4911–4916, 2014. doi: 10.1109/CDC.2014.7040156.

Ralph Allan Bradley and Milton E Terry. Rank analysis of incomplete block designs: I. the method of paired comparisons. Biometrika, 39(3/4):324–345, 1952.

Huayu Chen, Cheng Lu, Chengyang Ying, Hang Su, and Jun Zhu. Offline reinforcement learning via high-fidelity generative behavior modeling. In International Conference on Learning Representations (ICLR), 2023.

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. Proc. of NeurIPS, 2021.

Thomas Coste, Usman Anwar, Robert Kirk, and David Krueger. Reward model ensembles help mitigate overoptimization. arXiv preprint arXiv:2310.02743, 2023. URL https://arxiv.org/abs/2310.02743.

Will Dabney, Mark Rowland, Marc G. Bellemare, and Rémi Munos. Distributional reinforcement learning with quantile regression. In Proceedings of the AAAI Conference on Artificial Intelligence, volume 32, 2018.

Robert Dadashi, Léonard Hussenot, Matthieu Geist, and Olivier Pietquin. Primal wasserstein imitation learning. In International Conference on Learning Representations (ICLR), 2021. URL https://openreview.net/pdf?id=TtYSU29zgR.

DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. arXiv preprint arXiv:2501.12948, 2025. doi: 10.48550/arXiv.2501.12948. URL https://arxiv.org/abs/2501.12948.

Zihan Ding and Chi Jin. Consistency models as a rich and efficient policy class for reinforcement learning. arXiv preprint arXiv:2309.16984, 2023.

Jacob Eisenstein, Chirag Nagpal, Alekh Agarwal, Ahmad Beirami, Alex D'Amour, DJ Dvijotham, Adam Fisch, Katherine Heller, Stephen Pfohl, Deepak Ramachandran, Peter Shaw, and Jonathan Berant. Helping or herding? reward model ensembles mitigate but do not eliminate reward hacking. arXiv preprint arXiv:2312.09244, 2023. URL https://arxiv.org/abs/2312.09244.

Benjamin Eysenbach and Sergey Levine. Maximum entropy rl (provably) solves some robust rl problems. arXiv preprint arXiv:2103.06257, 2021.

Kevin Frans, Seohong Park, Pieter Abbeel, and Sergey Levine. Diffusion guidance is a controllable policy improvement operator. arXiv preprint arXiv:2505.23458, 2025.

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

Scott Fujimoto and Shixiang Shane Gu. A minimalist approach to offline reinforcement learning. ArXiv preprint, 2021.

Scott Fujimoto, Herke van Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In Proc. of ICML, pages 1582–1591, 2018.

Scott Fujimoto, David Meger, and Doina Precup. Off-policy deep reinforcement learning without exploration. In Proc. of ICML, pages 2052–2062, 2019.

Leo Gao, John Schulman, and Jacob Hilton. Scaling laws for reward model overoptimization. In Proceedings of the 40th International Conference on Machine Learning, volume 202 of Proceedings of Machine Learning Research, 2023. URL https://proceedings.mlr.press/v202/gao23h.html. See also arXiv:2210.10760.

Divyansh Garg, Shuvam Chakraborty, Chris Cundy, Jiaming Song, and Stefano Ermon. Iq-learn: Inverse soft-q learning for imitation. Advances in Neural Information Processing Systems, 34:4028–4039, 2021.

Tuomas Haarnoja, Haoran Tang, Pieter Abbeel, and Sergey Levine. Reinforcement learning with deep energy-based policies. In Proc. of ICML, pages 1352–1361, 2017.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In Proc. of ICML, pages 1856–1865, 2018.

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.

Richard Jordan, David Kinderlehrer, and Felix Otto. The variational formulation of the fokkerâĂŞplanck equation. SIAM Journal on Mathematical Analysis, 29(1):1–17, 1998. doi: 10.1137/S0036141096303359.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Proc. of ICLR, 2015.

Tomasz Korbak, Ethan Perez, and Christopher L Buckley. Rl with kl penalties is better viewed as bayesian inference. arXiv preprint arXiv:2205.11275, 2022.

Ilya Kostrikov, Rob Fergus, Jonathan Tompson, and Ofir Nachum. Offline reinforcement learning with fisher divergence critic regularization. In Proc. of ICML, pages 5774–5783, 2021a.

Ilya Kostrikov, Ashvin Nair, and Sergey Levine. Offline reinforcement learning with implicit q-learning. ArXiv preprint, 2021b.

Solomon Kullback and Richard A. Leibler. On information and sufficiency. Annals of Mathematical Statistics, 22(1):79–86, 1951. doi: 10.1214/aoms/1177729694.

Aviral Kumar, Justin Fu, Matthew Soh, George Tucker, and Sergey Levine. Stabilizing off-policy q-learning via bootstrapping error reduction. In Proc. of NeurIPS, pages 11761–11771, 2019.

Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative q-learning for offline reinforcement learning. In Proc. of NeurIPS, 2020.

Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. End-to-end training of deep visuomotor policies. Journal of Machine Learning Research, 2016.

Jianxiong Li, Xianyuan Zhan, Haoran Xu, Xiangyu Zhu, Jingjing Liu, and Ya-Qin Zhang. When data geometry meets deep function: Generalizing offline reinforcement learning. In The Eleventh International Conference on Learning Representations, 2022.

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

Qiang Liu. Stein variational gradient descent as gradient flow. Advances in neural information processing systems, 30, 2017.

Qiang Liu and Dilin Wang. Stein variational gradient descent: A general purpose bayesian inference algorithm. Advances in neural information processing systems, 29, 2016.

Liyuan Mao, Haoran Xu, Weinan Zhang, Xianyuan Zhan, and Amy Zhang. Diffusion-dice: In-sample diffusion guidance for offline reinforcement learning. arXiv preprint arXiv:2407.20109, 2024.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. arXiv preprint arXiv:1312.5602, 2013.

Ted Moskovitz, Aaditya K. Singh, DJ Strouse, Tuomas Sandholm, Ruslan Salakhutdinov, Anca D. Dragan, and Stephen McAleer. Confronting reward model overoptimization with constrained rlhf. In International Conference on Learning Representations, 2024. URL https://arxiv.org/abs/2310.04373.

Ofir Nachum, Mohammad Norouzi, Kelvin Xu, and Dale Schuurmans. Bridging the gap between value and policy based reinforcement learning. In Proc. of NeurIPS, pages 2775–2785, 2017.

Ashvin Nair, Murtaza Dalal, Abhishek Gupta, and Sergey Levine. Accelerating online reinforcement learning with offline datasets. ArXiv preprint, 2020.

Reiichiro Nakano, Jacob Hilton, Suchir Balaji, Jeff Wu, Long Ouyang, Christina Kim, Christopher Hesse, Shantanu Jain, Vineet Kosaraju, William Saunders, et al. Webgpt: Browser-assisted question-answering with human feedback. arXiv preprint arXiv:2112.09332, 2021.

Eric Nalisnick, Akihiro Matsukawa, Yee Whye Teh, Dilan Görür, and Balaji Lakshminarayanan. Do deep generative models know what they don't know? arXiv preprint arXiv:1810.09136, 2018. URL https://arxiv.org/abs/1810.09136.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback. In Advances in Neural Information Processing Systems, 2022. URL https://arxiv.org/abs/2203.02155.

Richard Yuanzhe Pang, Vishakh Padmakumar, Thibault Sellam, Ankur Parikh, and He He. Reward gaming in conditional text generation. In Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics, pages 4746–4763. Association for Computational Linguistics, 2023. doi: 10.18653/v1/2023.acl-long.262.

Seohong Park, Kevin Frans, Benjamin Eysenbach, and Sergey Levine. Ogbench: Benchmarking offline goal-conditioned rl. In International Conference on Learning Representations (ICLR), 2025a.

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

Xue Bin Peng, Aviral Kumar, Grace Zhang, and Sergey Levine. Advantage-weighted regression: Simple and scalable off-policy reinforcement learning. ArXiv preprint, 2019.

Gabriel Peyré and Marco Cuturi. Computational Optimal Transport, volume 11. Foundations and Trends® in Machine Learning, 2019. doi: 10.1561/2200000073.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. Advances in neural information processing systems, 36:53728–53741, 2023.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Mingchuan Zhang, Y. K. Li, Y. Wu, Daya Guo, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. arXiv preprint arXiv:2402.03300, 2024. URL https://arxiv.org/abs/2402.03300.

David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of go without human knowledge. nature, 2017.

Joar Skalse, Nikolaus Howe, Dmitrii Krasheninnikov, and David Krueger. Defining and characterizing reward hacking. In Advances in Neural Information Processing Systems (NeurIPS), 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/hash/3d719fee332caa23d5038b8a90e81796-Abstract-Conference.html.

Yang Song, Conor Durkan, Iain Murray, and Stefano Ermon. Maximum likelihood training of score-based diffusion models. Advances in neural information processing systems, 34:1415–1428, 2021.

Nisan Stiennon, Long Ouyang, Jeff Wu, Daniel M. Ziegler, Ryan Lowe, Chelsea Voss, Alec Radford, Dario Amodei, and Paul Christiano. Learning to summarize with human feedback. In Advances in Neural Information Processing Systems (NeurIPS), 2020.

Richard S Sutton, Andrew G Barto, et al. Introduction to reinforcement learning. MIT press Cambridge, 1998.

Denis Tarasov, Vladislav Kurenkov, Alexander Nikulin, and Sergey Kolesnikov. Revisiting the minimalist approach to offline reinforcement learning. In NeurIPS, 2023.

Denis Tarasov, Vladislav Kurenkov, Alexander Nikulin, and Sergey Kolesnikov. Revisiting the minimalist approach to offline reinforcement learning. Advances in Neural Information Processing Systems, 36, 2024.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurélien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. Llama: Open and efficient foundation language models. arXiv preprint arXiv:2302.13971, 2023. doi: 10.48550/arXiv.2302.13971. URL https://arxiv.org/abs/2302.13971.

Hado Van Hasselt, Yotam Doron, Florian Strub, Matteo Hessel, Nicolas Sonnerat, and Joseph Modayil. Deep reinforcement learning and the deadly triad. ArXiv preprint, 2018.

Binghai Wang, Rui Zheng, Lu Chen, Yan Liu, Shihan Dou, Caishuang Huang, Senjie Jin, Enyu Zhou, Chenyu Shi, Songyang Gao, Nuo Xu, Yuhao Zhou, Xiaoran Fan, Zhiheng Xi, Jun Zhao, Xiao Wang, Tao Ji, Hang Yan, Lixing Shen, Zhan Chen, Tao Gui, Qi Zhang, Xipeng Qiu, Xuanjing Huang, Zuxuan Wu, and Yu-Gang Jiang. Secrets of rlhf in large language models part ii: Reward modeling. arXiv preprint arXiv:2401.06080, 2024. URL https://arxiv.org/abs/2401.06080.

Xiangsen Wang, Haoran Xu, Yinan Zheng, and Xianyuan Zhan. Offline multi-agent reinforcement learning with implicit global-to-local value regularization. In Thirty-seventh Conference on Neural Information Processing Systems, 2023.

Ziyu Wang, Alexander Novikov, Konrad Zolna, Josh Merel, Jost Tobias Springenberg, Scott E. Reed, Bobak Shahriari, Noah Y. Siegel, Çaglar Gülçehre, Nicolas Heess, and Nando de Freitas. Critic regularized regression. In Proc. of NeurIPS, 2020.

Fang Wu, Weihao Xuan, Ximing Lu, Zaid Harchaoui, and Yejin Choi. The invisible leash: Why rlvr may not escape its origin. arXiv preprint arXiv:2507.14843, 2025.

Yifan Wu, George Tucker, and Ofir Nachum. Behavior regularized offline reinforcement learning. ArXiv preprint, 2019.

Haoran Xu, Li Jiang, Jianxiong Li, Zhuoran Yang, Zhaoran Wang, Victor Wai Kin Chan, and Xianyuan Zhan. Offline rl with no ood actions: In-sample learning via implicit value regularization. arXiv preprint arXiv:2303.15810, 2023.

Joy Qiping Yang, Salman Salamatian, Ziteng Sun, Ananda Theertha Suresh, and Ahmad Beirami. Asymptotics of language model alignment. In 2024 IEEE International Symposium on Information Theory (ISIT), pages 2027–2032. IEEE, 2024.

Xiaoying Zhang, Jean-François Ton, Wei Shen, Hongning Wang, and Yang Liu. Mitigating reward overoptimization via lightweight uncertainty estimation. In Advances in Neural Information Processing Systems, 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/94bbcb744bbada8808fda05b9d9290d6-Paper-Conference.pdf.

Wenxuan Zhou, Sujay Bajracharya, and David Held. Latent action space for offline reinforcement learning. In Conference on Robot Learning, 2020.

Brian D. Ziebart. Modeling Purposeful Adaptive Behavior with the Principle of Maximum Causal Entropy. PhD thesis, Carnegie Mellon University, Pittsburgh, PA, 2010. CMU-ML-10-110.

A PROOF

A.1 PROOF OF THEOREM 1

Theorem 1. Assume the value function $R(s,a)$ is c-Lipschitz w.r.t the input action a. Define the implicit policy that performs Equation (7) for L steps with N particles as π_N^L. We have

$$\mathrm{MMD}^2(\mu, \pi_N^L) = \mathrm{MMD}^2(\pi_N^0, \pi_N^L) \leq \frac{2\epsilon L}{\sigma\sqrt{e}}\left(\frac{c}{\alpha} + \frac{1}{\sigma\sqrt{e}}\right).$$

Proof. When $R(s,a)$ is c-Lipschitz, we know that $\|\nabla_{a_j} R(s,a_j)\|_\infty \leq c$. Additionally, we have another key observation that the kernel k also has Lipschitz property. Formally,

$$k(x+\delta, y) - k(x,y) = \exp(-\|x-y\|^2/2\sigma^2) - \exp(-\|x+\delta-y\|^2/2\sigma^2) \leq \frac{\|\delta\|_\infty}{\sigma\sqrt{e}}.$$

In other words, the kernel k is $\frac{1}{\sigma\sqrt{e}}$-Lipschitz. Combining these two properties, we have

$$\begin{aligned}\|h(s,x)\|_\infty &= \|\mathbb{E}_{a_j\sim\mu_0}\left[k(a_j,x)\nabla_{a_j}R(s,a_j)/\alpha + \nabla_{a_j}k(a_j,x)\right]\|_\infty \\ &\leq \sup_{a_j} k(a_j,x)\cdot\frac{c}{\alpha} + \sup_{a_j}\|\nabla_{a_j}k(a_j,x)\|_\infty \\ &\leq \frac{c}{\alpha} + \frac{1}{\sigma\sqrt{e}}.\end{aligned}$$

The square of MMD distance is calculated as follows:

$$\mathrm{MMD}^2(\pi_N^L, \pi_N^0) = \mathbb{E}_{x,y\sim\pi_N^0}k(x,y) + \mathbb{E}_{x,y\sim\pi_N^L}k(x,y) - 2\mathbb{E}_{x\sim\pi_N^0, y\sim\pi_N^L}k(x,y).$$

Define $K := \frac{1}{\sigma\sqrt{e}}$ and $H := \frac{c}{\alpha} + \frac{1}{\sigma\sqrt{e}}$. Let us consider the condition of a single pair of particles x, y after one iteration. We have

$$\begin{aligned}&k(x,y) + k(x+\epsilon h(x), y+\epsilon h(y)) - k(x+\epsilon h(x), y) - k(x, y+\epsilon h(y)) \\ &\leq |k(x,y) - k(x+\epsilon h(x), y)| + |k(x+\epsilon h(x), y+\epsilon h(y)) - k(x, y+\epsilon h(y))| \\ &\leq 2\epsilon KH.\end{aligned}$$

Denote $x^{(k)}$ as the particle x after the k-th iteration, we have

$$\begin{aligned}\mathrm{MMD}^2(\pi_N^L, \pi_N^0) &= \mathbb{E}_{x,y\sim\pi_N^0}k(x,y) + \mathbb{E}_{x,y\sim\pi_N^L}k(x,y) - 2\mathbb{E}_{x\sim\pi_N^0, y\sim\pi_N^L}k(x,y) \\ &= \frac{1}{n^2}\sum_{x,y}\left[k(x^{(0)},y^{(0)}) + k(x^{(L)},y^{(L)}) - k(x^{(L)},y^{(0)}) - k(x^{(0)},y^{(L)})\right] \\ &\leq 2\epsilon KHL.\end{aligned}$$

□

A.2 PROOF OF THEOREM 2

Theorem 2. Define the ϵ-support of a distribution P as $\mathrm{supp}_\epsilon(P) := \{x : p(x) \geq \epsilon\}$. We have

$$\mathrm{supp}_\epsilon(\pi_N^L(\cdot|s)) \not\subseteq \mathrm{supp}_\epsilon(\mu(\cdot|s)).$$

The proof can be divided into two settings, where the policy distribution is discrete and continuous. We first present the simple proof for the discrete setting.

Proof. Denote the support of the behavioral policy as $\mathrm{supp}(\pi_N^0) = \{a_1, ..., a_N\}$. If the support of the learned policy from one-step SVGD $\mathrm{supp}(\pi_N^1)$ is a subset of $\mathrm{supp}(\pi_N^0)$, then we know that a_1 is updated to $a_i \in \mathrm{supp}(\pi_N^0)$. This indicates that

$$a_i = a_1 + \epsilon h(a_1) = a_1 + \epsilon\mathbb{E}_{a_j\sim\pi_N^0}\left[k(a_j,a_1)\nabla_{a_j}R(s,a_j)/\alpha + \nabla_{a_j}k(a_j,a_1)\right].$$

Note that a little disturbance in any of the dimensions of $\nabla_{a_j}R(s,a_j)$ will make the equation invalid when the other derivatives of R are fixed. Therefore, the equation is almost surely invalid. □

As for the continuous setting, we typically assume that the policies before and after an SVGD update both follow Gaussian distributions, which is a widely used assumption. Formally, we assume that $\pi_N^0 \sim \mathcal{N}(\mu_1, \sigma_1^2), \pi_N^1 \sim \mathcal{N}(\mu_2, \sigma_2^2)$.

Proof. We consider the gradient of an arbitrary particle a sampled from π_N^0. We know that

$$h(a) = \mathbb{E}_{a_j \sim \pi_N^0} \left[k(a_j, a) \nabla_{a_j} R(s, a_j)/\alpha + \nabla_{a_j} k(a_j, a) \right].$$

Let us investigate the first term. We also know that

$$\begin{aligned}
\nabla_{a_j} R(s, a_j) &= \nabla_{a_j} \log \pi_N^1(a_j|s) \\
&= \nabla_{a_j} \log \left[\frac{1}{\sqrt{2\pi}\sigma_2} \exp(-\frac{(a_j - \mu_2)^2}{2\sigma_2^2}) \right] \\
&= \frac{-\frac{1}{\sqrt{2\pi}\sigma_2} \exp(-\frac{(a_j-\mu_2)^2}{2\sigma_2^2}) \frac{a_j-\mu_2}{\sigma_2^2}}{\frac{1}{\sqrt{2\pi}\sigma_2} \exp(-\frac{(a_j-\mu_2)^2}{2\sigma_2^2})} \\
&= \frac{\mu_2 - a_j}{\sigma_2^2}.
\end{aligned}$$

Then, we calculate the second term:

$$\nabla_{a_j} k(a_j, a) = k(a_j, a) \frac{a_j - a}{\sigma_1^2}.$$

Combining both terms, we have

$$\begin{aligned}
h(a) &= \mathbb{E}_{a_j \sim \pi_N^0} \left[k(a_j, a) (\frac{\mu_2 - a_j}{\alpha\sigma_2^2} + \frac{a_j - a}{\sigma_1^2}) \right] \\
&= \mathbb{E}_{a_j \sim \pi_N^0} \left[k(a_j, a) \frac{\mu_1 - a_j}{\alpha\sigma_2^2} \right] + \mathbb{E}_{a_j \sim \pi_N^0} \left[k(a_j, a) (\frac{\mu_2 - \mu_1}{\alpha\sigma_2^2} + \frac{a_j - a}{\sigma_1^2}) \right] \\
&= \mathbb{E}_{a_j \sim \pi_N^0} \left[k(a_j, a) (\frac{\mu_2 - \mu_1}{\alpha\sigma_2^2} + \frac{a_j - a}{\sigma_1^2}) \right].
\end{aligned}$$

The last equation above is because of the symmetry of Gaussian distributions. We now focus on the first dimension of particles. Without loss of generality, we assume that $\mu_{1,1} \leq \mu_{2,1}$. From the ϵ-support of π_N^0, which is a closed region, we choose a with the smallest value in the first dimension, i.e., $(a - x)_1 \leq 0$ for any $x \in \text{supp}(\pi_N^0)$. This indicates that the first dimension of $h(a)$ is strictly greater than zero, which means that the updated particle of a is out of the ϵ-support of π_N^0. $\square$

B Experimental Details

B.1 Offline RL Evaluation Details

Environments, tasks, and datasets. In the offline setting, VGF is evaluated on different kinds of datasets from various environments.

For MuJoCo environments, we have the following datasets.

- `halfcheetah/hopper/walker2d-m` (medium): Collected by a policy with moderate performance, typically reaching around one-third of expert returns. These datasets represent structured but suboptimal behavior.
- `halfcheetah/hopper/walker2d-m-r` (medium-replay): Contains the replay buffer of the mediocre SAC policy. It includes a wide range of off-policy transitions, many of which are suboptimal or noisy.
- `halfcheetah/hopper/walker2d-m-e` (medium-expert): A 50-50 mixture of medium and expert trajectories. These datasets are designed to test whether algorithms can leverage near-optimal data when it is partially present.

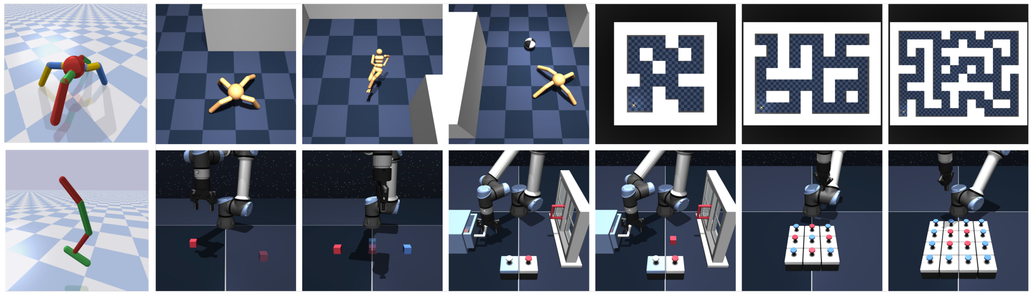

Figure 4: Visualization of offline RL tasks.

The AntMaze environments involve a quadruped ant navigating through a 2D maze using sparse goal-based rewards. The agent has a 29-dimensional state space and an 8-dimensional action space, corresponding to joint positions, velocities, and target location encoding. The tasks are particularly challenging due to long-horizon planning and sparse supervision.

- `antmaze-u` (umaze): A small maze where the agent must reach a fixed goal location using sparse rewards. The environment is relatively easy due to short trajectories.
- `antmaze-u-d` (umaze-diverse): Similar to `umaze`, but with broader trajectory diversity collected from random exploration.
- `antmaze-m-p` (medium-play): A medium-sized maze where data is collected via a play policy. The task is harder due to longer horizons and sparse goal rewards.
- `antmaze-m-d` (medium-diverse): Features more diverse and noisy behavior than `medium-play`, increasing exploration coverage but decreasing consistency.
- `antmaze-l-p` (large-play): A large maze with random play data. The agent must navigate long distances, making the task especially difficult under sparse reward signals.
- `antmaze-l-d` (large-diverse): Similar to `large-play`, but with broader and more varied behavior. It is one of the most challenging offline datasets due to the size of the environment and variability in data.

OGBench not only substantially extends the original AntMaze environment provided by the D4RL benchmark, but also introduces more challenging tasks, such as humanoid control and object manipulation. We elaborate on our selected environments below.

- `antmaze-giant-navigate`: An 8-DoF ant agent needs to reach a goal location in a 2-D maze, the size of which is substantially larger than those of D4RL datasets.
- `humanoid-medium/large-navigate`: This task involves full-body control of a 21-DoF Humanoid agent, which requires long-horizon reasoning.
- `antsoccer-arena-navigate`: This task involves controlling an ant agent to dribble a soccer ball. The agent is required to approach the ball and dribble it to random locations in an arena.
- `cube-single/double-play`: This task involves pick-and-place manipulation of several different cubes. The agent is required to complete tasks by moving, stacking, swapping, or permuting cubes. The options "single" and "double" refer to the number of cubes.
- `scene-play`: The agent's goal to manipulate the two buttons to determine the statuses of a window and a drawer to finish a specific task, which requires sequential reasoning.
- `puzzle-3×3/4×4-play`: The goal is to solve a "light-out" puzzle with a robot arm. At each step, the agent can press a button to change the states of the pressed button and its neighbors. We select 3×3 and 4×4 as the puzzle size.

Network Architecture and hyperparameters. In this part, we provide details of the hyperparameters used in our experiments on D4RL datasets and OGBench. Since most of the hyperparameters remain the same for all of our experiments, we list them in Table 4 and demonstrate our choice of

Table 4: Default Hyperparameters for VGF.

Hyperparameter	Value
Critic learning rate	0.0003
Actor learning rate	0.0003
Gradient steps	1000000
Batch size	256
Critic Hidden dimensions	$[256, 256, 256, 256]$
Discount factor γ	0.99
BC flow steps	5
Q value Aggregation	$\min$ for D4RL, mean for OGBench
Particle number N	5
Train particle select	mean
Eval particle select	$\max$

the important ones in Table 5 and Table 6. Specifically, we regard VGF learning rate ϵ, temperature parameter α, and training VGF steps L as the important hyperparameters.

We use a 4-layer MLP with 256 hidden units and Adam optimizer (Kingma and Ba, 2015) with a learning rate of 3×10^{-4} for both the behavior cloning network and the critic network in all tasks. We also use a target network with soft update weight 5×10^{-3} for critic update. We ran VGF for 10^6 gradient steps for all our experiments with batch size 256. We set BC flow steps to 5 for D4RL and 10 for OGBench, because we observe that changing BC flow steps within a dataset does not make much difference. In addition, we set particle number N to 5 because increasing N will bring about extra computational costs but hardly any improvement in performance. Our results are averaged across 5 seeds for all our experiments, with standard deviation reported. In Table 7, we further provide detailed results on each of the $5 \times 9 = 45$ tasks in OGBench, which further verifies VGF's efficacy across most of the environments.

B.2 RLHF Experiment Details

We evaluate on the TL;DR Summarize corpus (Stiennon et al., 2020) and the Anthropic Helpful & Harmless (HH) corpus (Bai et al., 2022). To reduce degenerate generations that miss an eos token, we filter out overly long prompts before training and evaluation: we discard prompts longer than 448 tokens for TL;DR and 348 tokens for HH (lengths measured after tokenization with the base model's tokenizer). For TL;DR we use the dedicated SFT split provided by the dataset. Unless otherwise noted, both the reward model and the policy are initialized from the same SFT checkpoint. For Pythia models we train the SFT stage for 2 epochs with an initial learning rate of 2e-5 on both summarization and dialogue tasks. We train the reward model for 1 epoch with an initial learning rate of 1e-5 using the same train splits as the policy initialization (TL;DR references or HH chosen responses). For both SFT and reward model training we use cosine learning-rate decay with a warm-up ratio of 0.03. All other implementation details are kept identical across tasks unless explicitly stated.

Table 5: Hyperparameter selection of VGF in D4RL datasets.

Env	Train VGF steps L	VGF learning rate ϵ	Temperature parameter α
halfcheetah-medium-v2	3	0.1	0.5
hopper-medium-v2	1	0.05	1.0
walker2d-medium-v2	1	0.1	1.0
halfcheetah-medium-replay-v2	3	0.1	1.0
hopper-medium-replay-v2	1	0.1	0.1
walker2d-medium-replay-v2	1	0.1	1.0
halfcheetah-medium-expert-v2	3	0.1	1.0
hopper-medium-expert-v2	1	0.05	1.0
walker2d-medium-expert-v2	1	0.05	1.0
antmaze-umaze-v2	5	0.2	0.2
antmaze-umaze-diverse-v2	3	0.1	0.2
antmaze-medium-play-v2	5	0.2	0.2
antmaze-medium-diverse-v2	3	0.1	0.2
antmaze-large-play-v2	5	0.2	0.2
antmaze-large-diverse-v2	5	0.2	0.2

Table 6: Hyperparameter selection of VGF in OGBench datasets.

Env	Train VGF steps L	VGF learning rate ϵ	Temperature parameter α
antmaze-giant-navigate-singletask-v0	5	0.05	0.2
humanoidmaze-medium-navigate-singletask-v0	1	0.05	1.0
humanoidmaze-large-navigate-singletask-v0	1	0.05	1.0
antsoccer-arena-navigate-singletask-v0	2	0.05	1.0
cube-single-play-singletask-v0	2	0.05	1.0
cube-double-play-singletask-v0	2	0.05	1.0
scene-play-singletask-v0	2	0.05	1.0
puzzle-3x3-play-singletask-v0	3	0.1	1.0
puzzle-4x4-play-singletask-v0	3	0.05	1.0

Table 7: OGBench results (all tasks). VGF performs comparable or superior to the baselines on most tasks. (*) denotes the default task per environment (Park et al., 2025b).

	Gaussian Policy		Diffusion/Flow Policy				w/o Policy (Ours)
Environment (5 tasks each)	BC	IQL	ReBRAC	FBRAC	IDQL	FQL	VGF
antmaze-giant-task1(*)	0 ±0	0 ±0	27 ±22	0 ±1	0 ±0	4 ±5	0 ±0
antmaze-giant-task2	0 ±0	1 ±1	16 ±17	4 ±7	0 ±0	9 ±7	9 ±3
antmaze-giant-task3	0 ±0	0 ±0	34 ±22	0 ±0	0 ±0	0 ±1	0 ±0
antmaze-giant-task4	0 ±0	0 ±0	5 ±12	9 ±4	0 ±0	14 ±23	0 ±0
antmaze-giant-task5	1 ±1	19 ±7	49 ±22	6 ±10	0 ±1	16 ±28	6 ±2
hmmaze-medium-task1(*)	1 ±0	32 ±7	16 ±9	25 ±8	1 ±1	19 ±12	86 ±1
hmmaze-medium-task2	1 ±0	41 ±9	18 ±16	76 ±10	1 ±1	94 ±3	92 ±2
hmmaze-medium-task3	6 ±2	25 ±5	36 ±13	27 ±11	0 ±1	74 ±18	82 ±2
hmmaze-medium-task4	0 ±0	0 ±1	15 ±16	1 ±2	1 ±1	3 ±4	0 ±0
hmmaze-medium-task5	2 ±1	66 ±4	24 ±20	63 ±9	1 ±1	97 ±2	100 ±0
hmmaze-large-task1(*)	0 ±0	3 ±1	2 ±1	0 ±1	0 ±0	7 ±6	18 ±4
hmmaze-large-task2	0 ±0	0 ±0	0 ±0	0 ±0	0 ±0	0 ±0	2 ±1
hmmaze-large-task3	1 ±1	7 ±3	8 ±4	10 ±2	3 ±1	11 ±7	12 ±6
hmmaze-large-task4	1 ±0	1 ±0	1 ±1	0 ±0	0 ±0	2 ±3	2 ±1
hmmaze-large-task5	0 ±1	1 ±1	2 ±2	1 ±1	0 ±0	1 ±3	14 ±5
antsoccer-arena-task1	2 ±1	14 ±5	0 ±0	17 ±3	44 ±12	77 ±4	76 ±3
antsoccer-arena-task2	2 ±2	17 ±7	0 ±1	8 ±2	15 ±12	88 ±3	75 ±2
antsoccer-arena-task3	0 ±0	6 ±4	0 ±0	16 ±3	0 ±0	61 ±6	62 ±9
antsoccer-arena-task4(*)	1 ±0	3 ±2	0 ±0	24 ±4	0 ±1	39 ±6	58 ±8
antsoccer-arena-task5	0 ±0	2 ±2	0 ±0	15 ±4	0 ±0	36 ±9	44 ±8
cube-single-task1	10 ±5	88 ±3	89 ±5	73 ±33	95 ±2	97 ±2	98 ±2
cube-single-task2(*)	3 ±1	85 ±8	92 ±4	83 ±13	96 ±2	97 ±2	100 ±0
cube-single-task3	9 ±3	91 ±5	93 ±3	82 ±12	99 ±1	98 ±2	100 ±0
cube-single-task4	2 ±1	73 ±6	92 ±3	79 ±20	93 ±4	94 ±3	94 ±3
cube-single-task5	3 ±3	78 ±9	87 ±8	76 ±33	90 ±6	93 ±3	88 ±4
cube-double-task1	8 ±3	27 ±5	45 ±6	47 ±11	39 ±19	61 ±9	80 ±6
cube-double-task2(*)	0 ±0	1 ±1	7 ±3	22 ±12	16 ±10	36 ±6	70 ±10
cube-double-task3	0 ±0	0 ±0	4 ±1	4 ±2	17 ±8	22 ±5	66 ±8
cube-double-task4	0 ±0	0 ±0	1 ±1	0 ±1	0 ±1	5 ±2	22 ±4
cube-double-task5	0 ±0	4 ±3	4 ±2	2 ±2	1 ±1	19 ±10	66 ±11
scene-task1	19 ±6	94 ±3	95 ±2	96 ±8	100 ±0	100 ±0	100 ±0
scene-task2(*)	1 ±1	12 ±3	50 ±13	46 ±10	33 ±14	76 ±9	96 ±2
scene-task3	1 ±1	32 ±7	55 ±16	78 ±14	94 ±4	98 ±1	94 ±2
scene-task4	2 ±2	0 ±1	3 ±3	4 ±4	4 ±3	5 ±1	2 ±2
scene-task5	0 ±0	0 ±0	0 ±0	0 ±0	0 ±0	0 ±0	2 ±1
puzzle-3x3-task1	5 ±2	33 ±6	97 ±4	63 ±19	52 ±12	90 ±4	100 ±0
puzzle-3x3-task2	1 ±1	4 ±3	1 ±1	2 ±2	0 ±1	16 ±5	71 ±6
puzzle-3x3-task3	1 ±1	3 ±2	3 ±1	1 ±1	0 ±0	10 ±3	62 ±7
puzzle-3x3-task4(*)	1 ±1	2 ±1	2 ±1	2 ±2	0 ±0	16 ±5	75 ±6
puzzle-3x3-task5	1 ±0	3 ±2	5 ±3	2 ±2	0 ±0	16 ±3	76 ±6
puzzle-4x4-task1	1 ±1	12 ±2	26 ±4	32 ±9	48 ±5	34 ±8	85 ±8
puzzle-4x4-task2	0 ±0	7 ±4	12 ±4	5 ±3	14 ±5	16 ±5	26 ±3
puzzle-4x4-task3	0 ±0	9 ±3	15 ±3	20 ±10	34 ±5	18 ±5	65 ±6
puzzle-4x4-task4(*)	0 ±0	5 ±2	10 ±3	5 ±1	26 ±6	11 ±3	43 ±6
puzzle-4x4-task5	0 ±0	4 ±1	7 ±3	4 ±3	24 ±11	7 ±3	12 ±2

C PSEUDOCODE OF VGF

```
import jax
import jax.numpy as jnp

def rbf_kernel(X, Y, sigma=None):
    """X: [B, n, d], Y: [B, m, d], returns K_XY: [B, n, m]"""
    X2 = jnp.sum(X * X, axis=-1, keepdims=True)
    Y2 = jnp.sum(Y * Y, axis=-1, keepdims=True).transpose(0, 2, 1)
    XY = jnp.matmul(X, Y.transpose(0, 2, 1))
    dnorm2 = X2 + Y2 - 2.0 * XY
    dnorm2 = jnp.maximum(dnorm2, 0.0)
    if sigma is None:
        # median heuristic per batch
        h = jnp.median(dnorm2, axis=(1,2))
        h /= (2.0 * jnp.log(X.shape[1] + 1.0))
        sigma_val = jnp.sqrt(jnp.maximum(h, 1e-12))
        sigma_val = sigma_val[:, None, None]
    else:
        sigma_val = jnp.asarray(sigma)
        if sigma_val.ndim == 0:
            sigma_val = jnp.broadcast_to(sigma_val, (X.shape[0], 1, 1))
    gamma = 1.0 / (1e-6 + 2.0 * (sigma_val ** 2))
    K_XY = jnp.exp(-gamma * dnorm2)
    return K_XY

class VGF:
    def __init__(self, q, alpha, optimizer):
        self.q = q
        self.alpha = alpha
        self.optim = optimizer
        self.opt_state = None

    def init(self, particles):
        self.opt_state = self.optim.init(particles)
        return particles, self.opt_state

    def phi(self, obs, particles):
        # obs: [B, D], particles: [B, N, D]
        # score terms
        def sum_q(action):
            obs_flatten = obs.reshape(-1, obs.shape[-1])
            action_flatten = action.reshape(-1, action.shape[-1])
            qs = self.q(obs_flatten, action_flatten)
            q = jnp.mean(qs, axis=0)
            return jnp.sum(q)
        score = jax.grad(sum_q)(particles)
        # kernel terms
        particles_stop = jax.lax.stop_gradient(particles)
        K_xx = rbf_kernel(particles, particles_stop)
        def sum_K(x):
            return jnp.sum(rbf_kernel(x, particles_stop))
        grad_q = jax.lax.stop_gradient(K_xx) @ score
        grad_K = -jax.grad(sum_K)(particles)
        phi_val = (grad_q / self.alpha + grad_K) / particles.shape[1]
        return phi_val

    def step(self, obs, particles, opt_s):
        grads = self.phi(obs, particles)
        updates, new_opt_s = self.optim.update(-grads, opt_s, particles)
        new_particles = optax.apply_updates(particles, updates)
        return new_particles, new_opt_s
```

Figure 5: A simple implementation of the VGF process.