Kernel Value Regression in Offline Reinforcement Learning

Jongyeon Lee[1]* **Jaehyoung Jeon**[2]* **Haneol Kim**[2] **Myungjoo Kang**[1,2,3,4]†

[1]Interdisciplinary Program in Artificial Intelligence, Seoul National University
[2]Department of Mathematical Sciences, Seoul National University
[3]Interdisciplinary Program in Computational Sciences and Technology, Seoul National University
[4]Research Institute of Mathematics, Seoul National University
{aileejy, jan4021, haneol.kijm, mkang}@snu.ac.kr

Abstract

Value overestimation is a central challenge in offline reinforcement learning (RL), arising when Bellman backups propagate out-of-distribution (OOD) errors that lead to catastrophic inflation of Q-values. Prior approaches address this issue by encouraging the learned policy to stay close to the behavior policy, but they often result in suboptimal policies or require complex training procedures. More recent work attempts to tackle this issue through minimal modifications to the critic network, yet it remains limited in either theoretical grounding or practical implementation. In this work, we present Kernel Value Regression (KVR), a simple critic regularization method based on kernel regression. KVR builds on the vanishing extrapolation property of kernel ridge regression, which encourages value estimates to decay toward zero away from the data support. This inductive bias naturally suppresses Q-values for OOD actions without auxiliary regularizers, providing a structural mechanism for addressing value overestimation. Using random Fourier features, KVR can be incorporated into standard actor–critic algorithms with only minimal modifications. We empirically demonstrate that these properties persist in complex offline RL environments, leading to strong performance across diverse tasks in the D4RL and OGBench benchmarks.

1 Introduction

Offline Reinforcement Learning (RL) enables agents to learn from static datasets with little to no environment interaction, making it particularly useful for domains where data collection is inefficient or risky. However, standard off-policy methods often suffer from value overestimation in offline RL, as Bellman backups propagate erroneous estimates for out-of-distribution (OOD) actions [18]. In response to this issue, earlier approaches either conservatively regularize value estimates, often resulting in overly pessimistic policies [17, 23], or restrict action queries to the in-distribution (IND) support [45, 25], typically requiring additional generative models that increase learning complexity. Critic regularization methods [13, 44] offer a simpler alternative with minimal modifications to the critic network, yet lack theoretical guarantees for suppressing extrapolated values outside the data support. Consequently, they still rely on auxiliary penalty modules to explicitly suppress OOD actions [13].

This raises a natural question: *Is there a mathematical mechanism that provides theoretical guarantees against value overestimation while remaining simple to implement?* We find that kernel ridge

*Equal contribution.
†Corresponding author.

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

regression (KRR) offers a promising answer. For shift-invariant kernels whose correlations decay with distance, KRR exhibits vanishing behavior, where predictions collapse toward zero far from the training support. Although this behavior is often viewed as a limitation for generalization, we reinterpret it as a structural safeguard against value overestimation. By defining the Q-function within the reproducing kernel Hilbert space (RKHS) generated by such a kernel, we can endow the critic with an inductive bias whose extrapolation geometry follows a flat vanishing profile. As a result, the critic naturally assigns lower Q-values to regions away from the data support, thereby mitigating overestimation.

However, directly applying KRR to offline RL introduces two practical challenges. First, although positive ridge regularization is assumed in our theoretical analysis, explicit ridge penalties can overly shrink Q-values and degrade performance in practice. Second, standard random Fourier features (RFF) with fixed Fourier frequencies can be sensitive to the initial random samples and kernel bandwidth. We address these challenges by adopting ridgeless optimization with implicit regularization and making Fourier frequencies learnable to reduce sensitivity to random sampling. Empirically, we observe that the desired extrapolation behavior persists under these adaptations.

To this end, we propose **Kernel Value Regression (KVR)**, a novel algorithm that intrinsically mitigates overestimation while maintaining a simple implementation. Our contributions are threefold:

- **Simple implementation.** KVR introduces a kernel-based inductive bias by replacing the critic's output layer with an RFF layer, enabling simple integration into standard actor-critic algorithms.
- **Theoretical basis and empirical validation.** KVR builds on the vanishing property of KRR, which provides a structural mechanism for mitigating value overestimation, and we show that this behavior persists in our practical implementation.
- **Strong empirical performance.** We evaluate KVR on diverse benchmarks, including D4RL [4] and OGBench [29], and demonstrate that it achieves competitive performance against existing methods.

To the best of our knowledge, KVR is the first offline RL method to explicitly leverage the vanishing extrapolation behavior of kernel regression as an inductive bias for addressing value overestimation.

2 Preliminaries

Offline RL. We consider a standard Markov decision process $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{T}, r, \rho, \gamma)$, where $\mathcal{S} \subset \mathbb{R}^p$ and $\mathcal{A} \subset \mathbb{R}^q$ denote the state and action spaces, $\mathcal{T} : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$ is the transition distribution, r is the reward function, ρ is the initial state distribution, and $\gamma \in [0, 1)$ is the discount factor. The goal of offline RL is to find a policy $\pi_\eta(a|s)$ maximizing the expected discounted return $\mathcal{R}(\pi_\eta) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)\right]$ using only a static dataset $\mathcal{D} = \{(s_i, a_i, r_i, s'_i)\}_{i=1}^N$. A key challenge in this setting is value overestimation. Bellman backups such as $y = r + \gamma \max_{a' \in \mathcal{A}} Q(s', a')$ may evaluate actions that are poorly supported by the dataset. Since the Q-function is not trained on these OOD actions, there is no corrective feedback for these erroneous estimates. This leads to accumulated bias and catastrophic overestimation.

Kernel Ridge Regression. KRR is a non-parametric regression method that minimizes regularized empirical risk in an RKHS [34]. Given a dataset $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^N$ with $\mathbf{x}_i \in \mathcal{X} \subset \mathbb{R}^d$ and $y_i \in \mathbb{R}$, let $\mathcal{H}_k$ denote the RKHS generated by a continuous positive semidefinite kernel $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$. The KRR predictor $f^\star : \mathcal{X} \to \mathbb{R}$ is defined as the minimizer of the regularized empirical risk in $\mathcal{H}_k$:

$$f^\star = \operatorname*{argmin}_{f \in \mathcal{H}_k} \left\{ \sum_{i=1}^{N} \big(f(\mathbf{x}_i) - y_i\big)^2 + \lambda \|f\|_{\mathcal{H}_k}^2 \right\}, \tag{1}$$

where $\lambda > 0$ is a regularization parameter. By the Representer Theorem [35], the solution admits a finite expansion:

$$f^\star(\mathbf{x}) = \sum_{i=1}^{N} \alpha_i \, k(\mathbf{x}, \mathbf{x}_i), \tag{2}$$

where $\alpha_i = ((\mathbf{K} + \lambda \mathbf{I})^{-1} \mathbf{y})_i$, $\mathbf{K} \in \mathbb{R}^{N \times N}$ is a kernel matrix with entries $\mathbf{K}_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$ and $\mathbf{y} = [y_1, \ldots, y_N]^\top$. For universal kernels, such as the Gaussian kernel, $\mathcal{H}_k$ is dense in the space of

continuous functions over any compact subset of $\mathcal{X}$ [38]. This ensures that $f^\star$ can approximate any continuous target function given sufficient data.

Random Fourier Features. Although KRR provides strong approximation fidelity, its $\mathcal{O}(N^3)$ complexity for inverting the kernel matrix is prohibitive for large datasets. RFF addresses this limitation by providing a scalable approximation to shift-invariant kernels [31]. Based on Bochner's theorem [33], RFF constructs an explicit finite-dimensional feature map whose inner product approximates the target kernel.

Specifically, let $\boldsymbol{\xi}_1, \ldots, \boldsymbol{\xi}_l$ be independently sampled from the probability measure associated with the shift-invariant kernel via Bochner's theorem. RFF approximates the kernel as $\hat{k}(\mathbf{x}, \mathbf{y}) = \phi(\mathbf{x})^\top \phi(\mathbf{y})$, where the Fourier feature map $\phi : \mathbb{R}^d \to \mathbb{R}^{2l}$ is defined as

$$\phi(\mathbf{x}) = \frac{1}{\sqrt{l}} \left[\cos(\boldsymbol{\xi}_1^\top \mathbf{x}), \sin(\boldsymbol{\xi}_1^\top \mathbf{x}), \ldots, \cos(\boldsymbol{\xi}_l^\top \mathbf{x}), \sin(\boldsymbol{\xi}_l^\top \mathbf{x})\right]^\top. \tag{3}$$

Let $\Phi \in \mathbb{R}^{N \times 2l}$ be a feature matrix whose i-th row is $\phi(\mathbf{x}_i)^\top$. Then the RFF approximation replaces the kernel matrix $\mathbf{K}$ with $\hat{\mathbf{K}} = \Phi\Phi^\top$, and the approximated KRR predictor can be written as

$$\hat{f}(\mathbf{x}) = \sum_{i=1}^{N} \hat{\alpha}_i \, \hat{k}(\mathbf{x}, \mathbf{x}_i) = \mathbf{w}^\top \phi(\mathbf{x}), \qquad \hat{\alpha} = (\hat{\mathbf{K}} + \lambda \mathbf{I})^{-1} \mathbf{y}, \quad \mathbf{w} = (\Phi^\top \Phi + \lambda \mathbf{I}_{2l})^{-1} \Phi^\top \mathbf{y}.$$

When $2l \ll N$, this formulation avoids full $N \times N$ kernel matrix inversion. Moreover, $\mathbf{w}$ can be learned by iterative optimization, effectively reducing RFF-based KRR to regularized linear regression over Fourier features. Further details are provided in Appendix B.1.

3 Kernel Value Regression

Despite KRR's strong expressivity and scalability, its extrapolation behavior has often been viewed as a limitation for generalization, especially when the training and test distributions are mismatched. This limitation stems from an intrinsic property of KRR: its predictions asymptotically decay toward zero away from the data support. We reinterpret this vanishing behavior as a desirable mechanism for addressing value overestimation in offline RL. By leveraging this property, we develop a novel offline RL algorithm that naturally assigns lower values to regions outside the data support.

In what follows, we first analyze the vanishing property of KRR and its RFF approximations. We then discuss the practical challenges of applying KRR to offline RL and present our implementation solutions. Next, through a simple didactic example, we empirically show that the desirable properties of KRR are preserved in practice. Finally, we introduce our main algorithm, Kernel Value Regression.

3.1 Theoretical Analysis and Statistical Guarantees

We begin by formalizing the vanishing property of KRR and then extend the analysis to its RFF approximation. Given a training dataset $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^N$, we define the maximum kernel similarity between a query $\mathbf{x}$ and the training inputs as $m_k(\mathbf{x}) := \max_{1 \le i \le N} |k(\mathbf{x}, \mathbf{x}_i)|$. For a shift-invariant kernel $k(\mathbf{x}, \mathbf{x}') = \kappa(\mathbf{x} - \mathbf{x}')$ satisfying $|\kappa(\mathbf{t})| \to 0$ as $\|\mathbf{t}\|_2 \to \infty$, the kernel similarity $m_k(\mathbf{x})$ vanishes as the query moves arbitrarily far from the empirical training set. Thus, $m_k(\mathbf{x})$ measures how strongly a query is correlated with the training data through the kernel.

Lemma 1. *Consider a shift-invariant positive definite kernel $k(\mathbf{x}, \mathbf{x}') = \kappa(\mathbf{x} - \mathbf{x}')$ with $\kappa(\mathbf{0}) = 1$ and $|\kappa(\mathbf{t})| \to 0$ as $\|\mathbf{t}\|_2 \to \infty$. Given a dataset $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^N$ with $\mathbf{x}_i \in \mathbb{R}^d$ and $y_i \in \mathbb{R}$, let $f^\star \in \mathcal{H}_k$ be the KRR predictor on $\mathcal{D}$ with regularization $\lambda > 0$, and let $\mathbf{y} = [y_1, \ldots, y_N]^\top$. Then, for any $\mathbf{x} \in \mathbb{R}^d$,*

$$|f^\star(\mathbf{x})| \le \|\mathbf{y}\|_\infty \min\left\{\sqrt{N} M_1, \; N M_2 m_k(\mathbf{x})\right\},$$

where

$$M_1 = \max_{i=1,\ldots,N} \frac{\sqrt{\sigma_i}}{\lambda + \sigma_i}, \qquad M_2 = \max_{i=1,\ldots,N} \frac{1}{\lambda + \sigma_i},$$

$\mathbf{K} = (k(\mathbf{x}_i, \mathbf{x}_j))_{i,j=1}^N$ is the kernel matrix, and $\{\sigma_i\}_{i=1}^N$ are its eigenvalues.

Lemma 1 bounds the magnitude of the KRR prediction at $\mathbf{x}$ by a constant multiple of $m_k(\mathbf{x})$. As the query point $\mathbf{x}$ moves farther away from the dataset, this correlation decreases, which in turn suppresses the output prediction. That is, the KRR predictor asymptotically vanishes away from the data support. We refer to this behavior as the *vanishing property*.

Theorem 1 (Finite-feature bound for RFF-KRR)**.** *Suppose that the conditions of Lemma 1 and Lemma 2 in Appendix B.3 hold. Let $\hat{k}_l$ be the RFF approximation of k using l random Fourier frequencies, and define*

$$\hat{f}_l(\mathbf{x}) = \sum_{i=1}^{N} \hat{\alpha}_{l,i}\, \hat{k}_l(\mathbf{x}, \mathbf{x}_i), \qquad \hat{\boldsymbol{\alpha}}_l = (\hat{\mathbf{K}}_l + \lambda \mathbf{I})^{-1}\mathbf{y},$$

where $\hat{\mathbf{K}}_l$ is the approximate kernel matrix induced by $\hat{k}_l$. Then the following holds.

For any compact $\mathcal{C} \subset \mathbb{R}^d$ containing $\{\mathbf{x}_i\}_{i=1}^N$ and any $\delta \in (0,1)$, there exist $C_1, C_2 > 0$, independent of l, such that, with probability at least $1-\delta$,

$$|\hat{f}_l(\mathbf{x})| \le C_1 m_k(\mathbf{x}) + C_2 l^{-1/2}, \qquad \forall\, \mathbf{x} \in \mathcal{C}.$$

Theorem 1 establishes that, with high probability, RFF-KRR predictions are bounded in absolute value by two terms: one proportional to the kernel similarity $m_k(\mathbf{x})$ and the other representing the approximation error induced by finite Fourier features, which decays at a rate of $O(l^{-1/2})$. As an input moves farther from the training data, the kernel similarity $m_k(\mathbf{x})$ decreases, and the bound approaches the finite-feature approximation error.

Extension to neural representations. The preceding analysis considers kernels defined directly on the input space. However, KVR applies these kernels in the latent space of a neural network. A sufficient condition for preserving the vanishing property in the input space is that inputs moving arbitrarily far from the training set are also mapped arbitrarily far from the latent representations of the training samples. We examine how this condition can be satisfied by two architectures: ReLU-based MLPs and residual networks with layer normalization (LN). For ReLU-based MLPs, their piecewise-affine structure makes the mapping asymptotically affine along any fixed input direction, ensuring the required distance relationship under suitable non-degeneracy conditions [8, 46]. For residual networks, LN bounds the residual branches, while skip connections provide a lower bound on latent distances in terms of input distances. We can therefore express the prediction bound in terms of the input-space distance to the training set.

Corollary 1 (Finite-feature bound under neural representations)**.** *Suppose that the conditions of Theorem 1 hold in the latent space. Let $g : \mathbb{R}^d \to \mathbb{R}^h$ be a neural map satisfying the sufficient conditions in Appendix B.4. Define the RFF-KRR predictor on the latent representations as*

$$\hat{f}_{\mathrm{NN},l}(\mathbf{x}) := \sum_{i=1}^{N} \hat{\alpha}_{l,i} \hat{k}_l(g(\mathbf{x}), g(\mathbf{x}_i)),$$

where $\hat{\boldsymbol{\alpha}}_l = (\hat{\mathbf{K}}_{g,l} + \lambda \mathbf{I})^{-1}\mathbf{y}$ and $(\hat{\mathbf{K}}_{g,l})_{ij} = \hat{k}_l(g(\mathbf{x}_i), g(\mathbf{x}_j))$.

Then there exists a non-increasing function $\psi_g : [0,\infty) \to [0,\infty)$ satisfying $\psi_g(t) \to 0$ as $t \to \infty$, such that, for any compact set $\mathcal{C} \subset \mathbb{R}^d$ containing the training inputs and any $\delta \in (0,1)$, there exist constants $C_1, C_2 > 0$, independent of l, such that, for each $l \ge 1$, with probability at least $1-\delta$,

$$|\hat{f}_{\mathrm{NN},l}(\mathbf{x})| \le C_1 \psi_g(d_{\mathcal{D}}(\mathbf{x})) + C_2 l^{-1/2}, \qquad \forall\, \mathbf{x} \in \mathcal{C},$$

where $d_{\mathcal{D}}(\mathbf{x}) := \min_{1 \le i \le N} \|\mathbf{x} - \mathbf{x}_i\|_2$.

Corollary 1 extends Theorem 1 to neural representations. The prediction magnitude is bounded by a term proportional to a non-increasing function ψ_g of the input-space distance to the training set, together with a finite-feature approximation error of order $O(l^{-1/2})$. Under suitable conditions, both ReLU-based MLPs and residual networks with LN admit a function ψ_g that vanishes as the distance increases, thereby preserving the vanishing behavior in the input space up to the finite-feature approximation error. The explicit form of ψ_g, the corresponding conditions, and the proofs are provided in Appendix B.4.

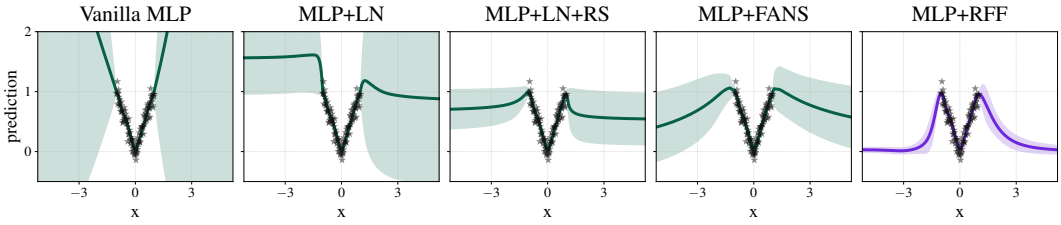


Figure 1: **Results of the didactic example (mean $\pm$ std, 10 runs).** In each run, the model is trained on 100 noisy samples ($y = |x| + \epsilon$) within the support $x \in [-1, 1]$. While all models exhibit reasonable interpolation within the support, only the RFF-based model achieves stable and desirable extrapolation.

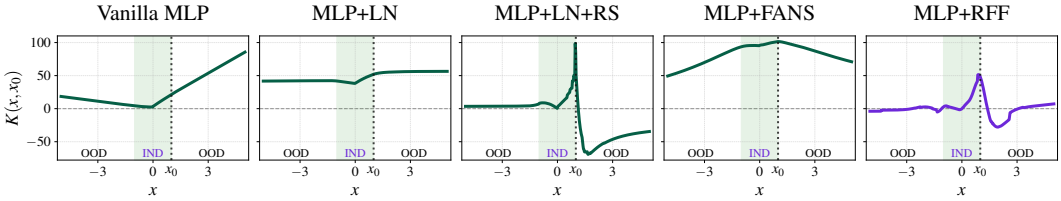


Figure 2: **NTK analysis for the didactic example.** We plot the NTK correlations between the reference point $x_0 = 1$ and x in the full input space for each trained model.

3.2 Practical Challenges and Solutions

While the vanishing property of KRR is theoretically desirable for addressing value overestimation, directly applying KRR to offline RL introduces two practical challenges. First, the preceding analysis assumes explicit ridge regularization, which can produce overly conservative Q-values, thereby degrading performance in practice. Second, standard RFF approximations use fixed Fourier frequencies sampled from the spectral distribution associated with a prescribed kernel, making their behavior sensitive to the random sampling and kernel hyperparameters. To address these challenges, we employ ridgeless optimization and slowly adapted Fourier frequencies.

Ridgeless Regression via Implicit Regularization. Although our theoretical analysis assumes $\lambda > 0$, explicit ridge regularization can excessively shrink Q-values and degrade offline RL performance, as shown in Figure 6. With the RFF-KRR predictor $\hat{f}(\mathbf{x}) = \mathbf{w}^\top \phi(\mathbf{x})$, ridge regularization penalizes $\|\mathbf{w}\|_2^2$. Since $|\hat{f}(\mathbf{x})| \leq \|\mathbf{w}\|_2 \|\phi(\mathbf{x})\|_2 = \|\mathbf{w}\|_2$, shrinking $\|\mathbf{w}\|_2$ globally suppresses critic values, potentially reducing useful IND values as well. We therefore adopt a ridgeless regime ($\lambda = 0$) leveraging finite-step gradient descent for implicit regularization. As shown in Appendices B.5 and C.4, finite-step gradient descent induces a spectral shrinkage effect analogous to explicit ridge regularization. This observation motivates our use of ridgeless optimization in the practical algorithm.

Learnable Spectral Frequencies. In standard RFF, the Fourier frequencies $\boldsymbol{\xi}_1, \ldots, \boldsymbol{\xi}_l$ are typically sampled once and then kept fixed. This can make the approximation sensitive to the initial random sampling and to hyperparameters such as the bandwidth. We instead optimize them with a small learning rate, motivated by prior work showing that adaptive random features can reduce sampling sensitivity and improve optimization [36, 20, 19]. As demonstrated in the following sections, this adaptation improves performance while preserving the desired extrapolation behavior.

3.3 Didactic Example

To examine whether the vanishing behavior persists under our practical design choices, we consider a 1D regression problem ($y = |x|$) following Ball et al. [2], where training data is restricted to the interval $[-1, 1]$ with noise $\epsilon \sim \mathcal{N}(0, 0.1)$. This didactic setup reflects the core challenge of offline RL: accurate interpolation within the data support while ensuring conservative extrapolation in unseen regions. As shown in Figure 1, while all models achieve high fidelity within the support, their extrapolation behaviors diverge significantly. Vanilla MLP and MLP with LN (MLP+LN) exhibit large predictions outside the support, accompanied by high variance, indicating severe sensitivity

to noise and numerical instability. Reward scaling (MLP+LN+RS) and FANS partially suppress such extrapolation, yet remain vulnerable to noise. In contrast, an MLP with an RFF layer achieves both accurate interpolation and desirable extrapolation, with low-variance predictions that smoothly decay toward zero outside the data support. Notably, this behavior emerges without explicit ridge regularization ($\lambda = 0$), suggesting that the kernel-inspired extrapolation bias can persist in the ridgeless setting. Detailed descriptions of the baseline algorithms are provided in Appendix C.1.

Neural Tangent Kernel analysis. To further investigate these extrapolation behaviors, we analyze the Neural Tangent Kernel (NTK), $k_{f_\theta}(\mathbf{x}, \mathbf{x}') = \langle \nabla_\theta f_\theta(\mathbf{x}), \nabla_\theta f_\theta(\mathbf{x}') \rangle$, centered at the boundary point $x = 1$. As shown in Figure 2, the MLP+LN+RS and MLP+RFF models exhibit strongly localized kernel structures, where high NTK values are concentrated near $x = 1$ and rapidly decay elsewhere. This *local correlation* confines learning signals to the supported regions, preventing signal leakage into unseen regions. Conversely, other models show non-local NTK distributions that maintain high correlations even in distant regions, leading to uncontrolled extrapolation.

Bandwidth regulation. Our approach provides an explicit mechanism for regulating the model's extrapolation behavior by adjusting the kernel bandwidth. Larger bandwidths allow broader extrapolation, whereas smaller bandwidths induce faster decay of predictions outside the support. As shown in Figure 3, the model maintains high fidelity within the data support across different bandwidths, while its extrapolation behavior varies systematically with the bandwidth. This provides a way to control the learned policy's preference for actions outside the data support.

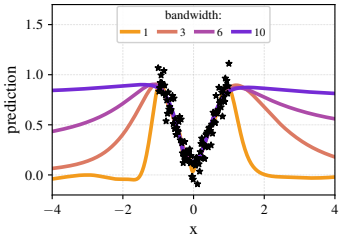


Figure 3: **Effect of kernel bandwidth on extrapolation.**

3.4 Main Algorithm

We are now ready to introduce **Kernel Value Regression (KVR)**, a simple kernel-based offline RL algorithm for mitigating value overestimation. KVR is motivated by the vanishing extrapolation property of KRR, under which predictions are increasingly attenuated as inputs move away from the data support. This structural bias provides a natural safeguard against spurious value inflation. We incorporate this inductive bias into a standard actor-critic framework through a simple modification to the critic, replacing its final linear layer with an RFF layer.

Base model. We employ TD3BC [5] as the base actor–critic algorithm, which extends TD3 [6] to the offline setting by incorporating a behavior cloning (BC) regularizer.

RFF-KRR critic. We construct the critic using an RFF layer, which consists of a Fourier feature mapping followed by a linear output projection. Let $g_\theta : \mathcal{S} \times \mathcal{A} \to Z$ be a neural network mapping the state-action space to a latent space $Z \subset \mathbb{R}^h$. The Q-function is then defined by an RFF expansion:

$$Q_\Theta(\mathbf{s}, \mathbf{a}) = \frac{1}{\sqrt{l}} \sum_{i=1}^{l} w_{2i-1} \cos(\boldsymbol{\xi}_i^\top g_\theta(\mathbf{s}, \mathbf{a})) + w_{2i} \sin(\boldsymbol{\xi}_i^\top g_\theta(\mathbf{s}, \mathbf{a})),$$

where $\Theta = \{\theta, \{\boldsymbol{\xi}_i\}_{i=1}^l, \mathbf{w}\}$ denotes the critic parameters, $\{\boldsymbol{\xi}_i\}_{i=1}^l$ are Fourier frequencies independently initialized from $\mathrm{Unif}([-\beta^{-1}, \beta^{-1}]^h)$ with bandwidth β, and $\mathbf{w} = \{w_i\}_{i=1}^{2l}$ denotes the coefficient vector. All critic parameters are jointly optimized without an explicit ridge penalty, with the Fourier frequencies updated using a smaller learning rate. This design reduces sensitivity to random frequency initialization and improves training stability while maintaining desirable extrapolation behavior.

We consider three architectures for g_θ: a ReLU-based MLP, its variant with LN applied immediately before the RFF layer, and a residual network with LN (Figure 18). Under suitable conditions, the vanilla MLP and the residual network can preserve the vanishing behavior in the input space (Corollary 1). Although applying LN to the output of the vanilla MLP, immediately before the RFF layer, prevents the latent representation from satisfying these conditions, it improves extrapolation stability, as further analyzed in Appendix D.10. We refer to these three variants as KVR-MLP (vanilla MLP), KVR-LN (MLP with LN immediately before the RFF layer), and KVR-RES (residual network with LN), respectively. Unless otherwise specified, KVR refers to KVR-LN, our default architecture.

Critic ensemble and reward scaling. Following RLPD [2], we employ a critic ensemble to improve training stability and performance. We also use reward scaling [13, 9], a common technique in RL. In

Table 1: **Performance results on benchmarks.** KVR-LN and KVR-MLP scores are task-averaged means ± estimated SD (24 seeds per task for D4RL, 12 for OGBench). Baselines are from official reports unless noted. Green and orange denote the best and second-best scores. Full results are provided in Tables 15 and 16.

GYM-MuJoCo	IQL	EDAC	TD3BC	ReBRAC	PARS	FANS	KVR-LN	KVR-MLP
random (3 tasks)	8.8	23.4	7.0	18.7	25.9	–	**27.7** ± 0.8	**27.8** ± 0.7
medium (3 tasks)	64.0	86.7	63.8	83.4	**88.5**	**90.7**	**88.5** ± 0.7	86.6 ± 0.6
medium-replay (3 tasks)	70.9	83.1	62.4	75.5	**85.3**	**85.8**	84.5 ± 3.6	84.7 ± 1.3
medium-expert (3 tasks)	95.9	**110.6**	99.6	106.6	109.3	**111.4**	109.6 ± 2.2	109.3 ± 0.8
Average	59.9	76.0	58.2	71.0	**77.3**	–	**77.6** ± 1.1	77.1 ± 0.5

AntMaze	IQL	EDAC	TD3BC	ReBRAC	PARS	FANS	KVR-LN	KVR-MLP
umaze (2 tasks)	74.9	0.0	75.0	93.1	91.9	80.1	**97.2** ± 1.3	**97.9** ± 1.4
medium-maze (2 tasks)	70.6	0.0	6.8	80.2	91.6	4.6	**93.6** ± 2.0	**92.1** ± 3.4
large-maze (2 tasks)	43.6	0.0	0.1	57.4	84.0	0.0	**91.4** ± 2.2	**89.6** ± 3.0
Average	63.0	0.0	27.3	76.9	89.2	28.2	**94.0** ± 1.1	**93.2** ± 1.6

OGBench singletask (5 tasks)	IQL	ReBRAC	IFQL	FQL	GFP	KVR-LN	KVR-MLP
antmaze-large-navigate	53.4	80.8	28.0	78.6	**93.8**	**96.5** ± 0.8	92.8 ± 6.1
antmaze-giant-navigate	4.0	26.2	2.6	8.6	27.9	**86.3** ± 2.0	**62.4** ± 10.9
humanoidmaze-medium-navigate	32.8	21.8	60.4	57.4	**72.0**	70.0 ± 6.9	**70.1** ± 11.2
humanoidmaze-large-navigate	2.4	2.6	11.0	4.2	**17.7**	13.4 ± 3.7	**64.1** ± 13.5
scene-play	27.6	40.6	30.4	55.8	53.5	**75.6** ± 4.5	**81.1** ± 3.9
puzzle-3x3-play	9.0	21.6	19.0	29.6	23.1	**95.4** ± 1.5	**98.0** ± 0.6
Average	21.5	32.3	25.2	39.0	48.0	**72.9** ± 1.6	**78.1** ± 3.6

sparse-reward and long-horizon tasks, reward scaling helps propagate reward information from goal states to distant regions of the state space.

4 Experiments

4.1 Experimental Setup

Benchmarks. To evaluate the versatility and robustness of KVR, we conduct extensive experiments on two widely used offline RL benchmarks: D4RL [4] and OGBench [29]. For D4RL, we consider two domains: (1) Gym-MuJoCo, which consists of dense-reward locomotion tasks with datasets of varying quality, and (2) AntMaze, which consists of challenging sparse-reward navigation tasks. For OGBench, we evaluate KVR on four maze navigation tasks and two robotic manipulation tasks, each comprising five distinct subtasks to ensure statistical reliability. Further details on the benchmarks are provided in Appendix D.1.

Baselines. We evaluate our models against a broad range of baselines, including an ensemble approach (EDAC [1]), critic regularizers (PARS [13], FANS [44]), and generative policy models (IFQL, FQL [30], GFP [42]). We prioritize officially reported scores, so the compared baselines vary across benchmarks depending on result availability. In the main text, we report results for KVR-LN and KVR-MLP, while results for all three variants are provided in Appendix D.10.2.

Number of seeds. To alleviate the seed search trap, where hyperparameter tuning can exploit stochastic variance to report misleadingly high scores, we significantly increase the number of learning seeds. This issue is closely related to the distractor policy phenomenon [11], in which policies may achieve high peak performance despite low average performance. Specifically, we use 24 seeds for D4RL and 12 seeds for OGBench. Some additional experiments use 5 seeds, so their results may differ slightly from the main benchmark scores. When aggregating results across K tasks, we report the mean ± estimated SD, with $\text{SD} = \sqrt{\sum_{i=1}^{K} \sigma_i^2}/K$ assuming independent tasks, where σ_i is the standard deviation for task i.

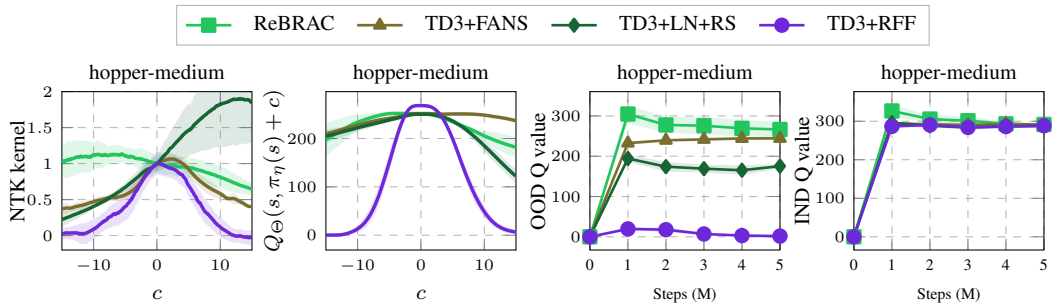


Figure 4: **Effective extrapolation on the D4RL hopper-medium task (mean $\pm$ std, 5 seeds).** From left to right: critic's NTK map, Q-value curves for perturbed actions, Q-values for infeasible actions, and Q-values for IND state-action pairs. TD3 with an RFF layer effectively separates OOD regions from IND regions. Refer to D.3 for the detailed experimental settings.

4.2 Benchmark Results

The evaluation results are summarized in Table 1. On MuJoCo, both KVR-LN and KVR-MLP achieve performance comparable to competitive baselines, with KVR-LN attaining a slightly higher average score than PARS. On AntMaze, both variants outperform the strongest baseline on average, with KVR-LN achieving consistently high scores across all tasks. On OGBench, both variants substantially outperform existing baselines on average, despite using standard MLP policies rather than more expressive generative policies. In particular, KVR-MLP performs well on the challenging humanoidmaze-large environment, averaging 64.1 across its five tasks, compared with 17.7 for GFP.

Importantly, KVR-MLP achieves competitive performance against strong baselines despite not using LN, which is widely used to mitigate overestimation and stabilize critic estimates. This suggests that LN is not essential for achieving strong performance with KVR. We further investigate this observation through ablation studies in the following section.

4.3 Research Q&A

To elucidate the properties and effectiveness of KVR, we present several research questions, with a particular focus on whether the vanishing property of KRR persists in our practical implementation.

Q1. Does KVR preserve the desired extrapolation behavior in complex offline RL tasks?

A: Our practical implementation relaxes several theoretical assumptions by removing ridge regularization, making Fourier frequencies learnable, and applying LN immediately before the RFF layer in KVR-LN. We therefore examine whether the desirable extrapolation behavior of KRR persists in complex offline RL tasks.

We first investigate the extrapolation behavior of KVR-MLP. We measure Q-values for perturbed actions, $Q_\Theta\big(\mathbf{s}, \pi_\eta(\mathbf{s}) + c \cdot \mathbb{1}\big)$, where π_η is the learned policy, $\mathbf{s}$ is sampled from the dataset, $c \in \mathbb{R}$, and $\mathbb{1} = [1, \ldots, 1]^\top$. As $|c|$ increases, the perturbed action moves farther from the policy output, providing a simple probe of extrapolation. We additionally compare Q-values for IND state-action pairs $(\mathbf{s}, \mathbf{a})$ from the dataset and OOD pairs $(\mathbf{s}, \mathbf{a}')$, where $\mathbf{a}'$ is sampled outside the valid action space following PARS [13]. As shown in Figure 4, KVR-MLP exhibits an overall decay in Q-values as the perturbed actions move farther from the policy output (second plot). KVR-MLP also consistently assigns lower Q-values to sampled OOD actions than the baselines, while maintaining high values on IND samples (third and fourth plots).

We further analyze the locality of the critic through its NTK. For a fixed state, we compute the gradient inner product $\langle \nabla_\Theta Q_\Theta(\mathbf{s}, \pi_\eta(\mathbf{s})), \nabla_\Theta Q_\Theta(\mathbf{s}, \pi_\eta(\mathbf{s}) + c \cdot \mathbb{1}) \rangle$, normalized by its value at $c = 0$. As shown in the first plot of Figure 4, the NTK of KVR-MLP peaks at the origin and decays as $|c|$ increases, indicating a localized correlation structure around the reference action. This localization suggests that learning signals at the reference action have a limited influence on the Q-values of distant actions.

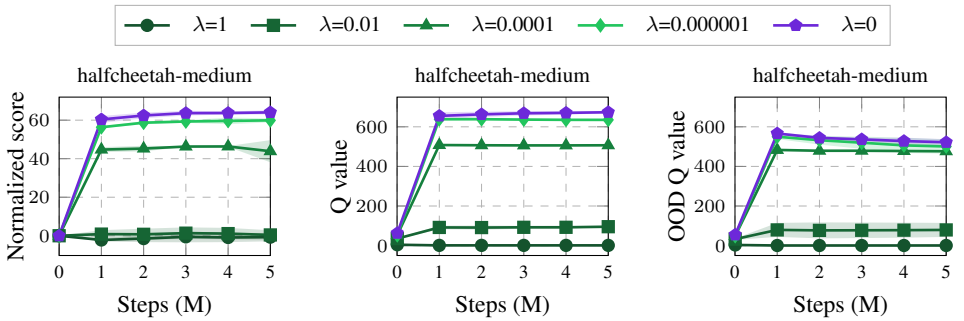


Figure 6: **Overly conservative Q-values induced by ridge regularization (mean ± std, 5 seeds).** The plots illustrate results on HalfCheetah-medium across varying ridge coefficients. See Table 13 for full results.

Next, we examine whether KVR-MLP retains the bandwidth-dependent extrapolation behavior of kernel regression in complex offline RL tasks. Following the preceding experiment, we measure Q-values for perturbed actions $\pi_\eta(\mathbf{s}) + c \cdot \mathbb{1}$ at randomly sampled dataset states $\mathbf{s}$. As shown in Figure 5, Q-values remain consistent across bandwidths near $c = 0$, but exhibit distinct extrapolation behaviors as $|c|$ increases, with smaller bandwidths inducing faster decay and larger bandwidths allowing broader extrapolation. These results suggest that KVR-MLP retains the characteristic bandwidth-dependent extrapolation behavior of KRR despite the relaxation of several theoretical assumptions.

Further analysis of KVR-LN is provided in Appendix D.10.

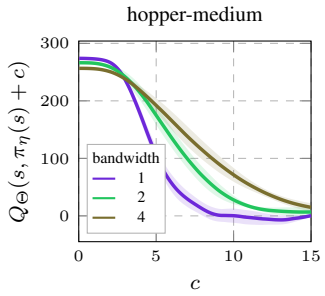


Figure 5: **Effect of bandwidth on extrapolation behavior (mean ± std, 5 seeds).**

Q2. Is ridge regularization a necessity or a bottleneck?

A: We investigate the effect of the explicit ridge λ on the Bellman update:

$$\min_{\Theta} \mathbb{E}_{(\mathbf{s},\mathbf{a},r,\mathbf{s}')\sim\mathcal{D}} \left[\left\| Q_\Theta(\mathbf{s},\mathbf{a}) - \left(r + \gamma \min_{i \le N_Q} \mathbb{E}_{\mathbf{a}'\sim\pi_\eta(\cdot|\mathbf{s}')} \left[\hat{Q}^i(\mathbf{s}',\mathbf{a}') \right] \right) \right\|_2^2 + \lambda \|\mathbf{w}\|_2^2 \right],$$

where N_Q denotes the ensemble size, $\hat{Q}_i$ denotes the i-th target critic in the ensemble. We evaluate five ridge settings, $\lambda \in \{1, 10^{-2}, 10^{-4}, 10^{-6}, 0\}$, on six D4RL tasks. As shown in Figure 6, the ridgeless setting ($\lambda = 0$) achieves the highest normalized score, while larger ridge coefficients generally degrade performance. This degradation appears to stem from excessive shrinkage of Q-values under larger explicit ridge penalties, leading to overly conservative value estimates.

Q3. Can learnable Fourier frequencies improve KVR without sacrificing extrapolation?

A: As discussed in Section 3.2, we make Fourier frequencies learnable to alleviate sampling dependency. To evaluate this design choice, we examine both performance and Q-value estimation on four D4RL AntMaze tasks: medium-diverse, medium-play, large-diverse, and large-play. First, we compare KVR with fixed and learnable Fourier frequencies. As shown in Figure 7, learnable frequencies consistently improve performance across all four tasks, yielding an average gain of approximately 10 points. We further examine whether this adaptation maintains low Q-value estimates for unsupported actions. Figure 17 shows that Q-values for infeasible actions remain consistently lower than those for IND state-action pairs throughout training across all four tasks. These re-

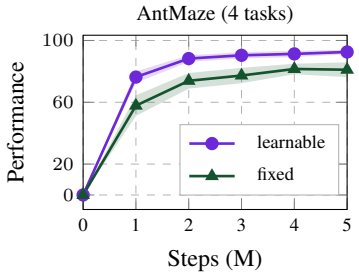


Figure 7: **Fixed vs learnable Fourier frequencies on AntMaze medium and large tasks (mean ± SD, 5 seeds).**

sults indicate that learnable Fourier frequencies improve KVR's performance while maintaining effective suppression of values for unsupported actions.

Q4. Does KVR's OGBench performance persist under matched experimental settings?

A: As shown in Table 1, both KVR-LN and KVR-MLP achieve strong performance relative to existing baselines on OGBench. To examine whether these gains persist under matched experimental settings, we conduct an additional comparison using KVR-LN. Specifically, we use the same reward settings and a training budget of 1M gradient steps for all methods, and remove KVR-LN's additional critic ensemble to match the non-ensemble baselines. We also retune and rerun each baseline following the guidelines of its original implementation. As reported in Table 10, KVR achieves the highest average performance among the evaluated methods. These results suggest that its gains are not solely attributable to differences in auxiliary experimental settings. Further details are provided in Appendix D.9.

Q5. What happens without the RFF layer?

A: To assess the contribution of the RFF layer to KVR's performance and its effectiveness in mitigating value overestimation, we conduct an ablation study by removing the RFF layer from KVR-MLP. We use KVR-MLP to isolate the effect of the RFF layer from that of LN, which can help mitigate overestimation. As shown in Figure 8, removing the RFF layer substantially degrades performance across D4RL and OGBench, leading to catastrophic performance drops on several tasks. We then examine critic estimates on a task exhibiting severe performance degradation (Table 2). Without the RFF layer, Q-values explode beyond 10^{14}, indicating severe value overestimation that may contribute to the performance collapse. In contrast, KVR-MLP maintains stable critic estimates, with Q-values for IND actions remaining higher than those for OOD actions. Notably, removing the RFF layer leads to exploding Q-values on 22 out of 48 evaluated tasks, whereas KVR-MLP maintains stable estimates across all tasks.

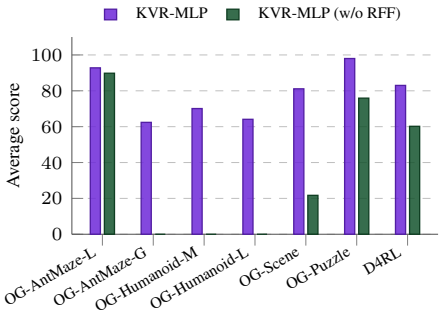


Figure 8: **Ablation study on the RFF layer.**

Together, these results demonstrate the critical role of the RFF layer in mitigating value overestimation and achieving strong performance in offline RL. Further details of the ablation setup are provided in Appendix D.16.

Table 2: **Ablation of the RFF layer on hopper-medium.**

Metric	KVR-MLP	KVR-MLP (w/o RFF layer)
Normalized score	104.3 $\pm$ 0.4	0.7 $\pm$ 0.0
IND Q value	287.9 $\pm$ 3.4	(1.1 $\pm$ 0.6)$\times 10^{14}$
OOD Q value	37.2 $\pm$ 13.2	(1.4 $\pm$ 1.9)$\times 10^{14}$

5 Conclusion and Limitations

In this work, we introduced Kernel Value Regression (KVR), a kernel-inspired offline RL method for mitigating value overestimation. KVR is motivated by the vanishing extrapolation property of KRR and incorporates this inductive bias into the critic through an RFF-based parameterization. Our evaluations show that KVR is competitive on MuJoCo and achieves the highest average scores among the compared methods on AntMaze and OGBench. Additional analyses show that KVR consistently maintains lower Q-values in OOD regions, while NTK analysis reveals the desirable local correlation structure of its predictions.

Despite these advantages, KVR has the following limitation. KVR may require an appropriate reward transformation when the reward setting is not well aligned with its inductive bias. For example, when IND Q-values are predominantly negative, near-zero OOD estimates can become larger than values on the data support, causing the policy to favor unsupported OOD actions. In future work, we aim to develop a general reward transformation method that aligns diverse reward settings with KVR's inductive bias.

Acknowledgments and Disclosure of Funding

This work was supported by grants from the National Research Foundation of Korea (NRF; RS-2024-00406127, RS-2026-25477522, RS-2026-25608312) and the Institute of Information & communications Technology Planning & Evaluation (IITP; RS-2021-II211343, Artificial Intelligence Graduate School Program at Seoul National University), funded by the Korea government (MSIT).

References

[1] Gaon An, Seungyong Moon, Jang-Hyun Kim, and Hyun Oh Song. Uncertainty-based offline reinforcement learning with diversified q-ensemble. *Advances in neural information processing systems*, 34:7436–7447, 2021.

[2] Philip J Ball, Laura Smith, Ilya Kostrikov, and Sergey Levine. Efficient online reinforcement learning with offline data. In *International Conference on Machine Learning*, pages 1577–1594. PMLR, 2023.

[3] James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Yash Katariya, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: composable transformations of Python+NumPy programs, 2018. URL `http://github.com/jax-ml/jax`.

[4] 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.

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

[6] Scott Fujimoto, Herke Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *International conference on machine learning*, pages 1587–1596. PMLR, 2018.

[7] John Guibas, Morteza Mardani, Zongyi Li, Andrew Tao, Anima Anandkumar, and Bryan Catanzaro. Adaptive fourier neural operators: Efficient token mixers for transformers. *arXiv preprint arXiv:2111.13587*, 2021.

[8] Matthias Hein, Maksym Andriushchenko, and Julian Bitterwolf. Why relu networks yield high-confidence predictions far away from the training data and how to mitigate the problem. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 41–50. IEEE, 2019.

[9] Peter Henderson, Riashat Islam, Philip Bachman, Joelle Pineau, Doina Precup, and David Meger. Deep reinforcement learning that matters. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

[10] Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*, 2016.

[11] Matthew Thomas Jackson, Uljad Berdica, Jarek Liesen, Shimon Whiteson, and Jakob Nicolaus Foerster. A clean slate for offline reinforcement learning. *arXiv preprint arXiv:2504.11453*, 2025.

[12] Ke Jiang, Wen Jiang, and Xiaoyang Tan. Variational ood state correction for offline reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pages 22327–22335, 2026.

[13] Jeonghye Kim, Yongjae Shin, Whiyoung Jung, Sunghoon Hong, Deunsol Yoon, Youngchul Sung, Kanghoon Lee, and Woohyung Lim. Penalizing infeasible actions and reward scaling in reinforcement learning with offline data. *arXiv preprint arXiv:2507.08761*, 2025.

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

[15] Young Su Ko, Jonathan Parkinson, Cong Liu, and Wei Wang. Tuna: an uncertainty-aware transformer model for sequence-based protein–protein interaction prediction. *Briefings in Bioinformatics*, 25(5):bbae359, 2024.

[16] Cevahir Koprulu, Franck Djeumou, et al. Neural stochastic differential equations for uncertainty-aware offline rl. In *The Thirteenth International Conference on Learning Representations*, 2025.

[17] 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.

[18] 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.

[19] Alexander Li and Deepak Pathak. Functional regularization for reinforcement learning via learned fourier features. *Advances in Neural Information Processing Systems*, 34:19046–19055, 2021.

[20] Yang Li, Si Si, Gang Li, Cho-Jui Hsieh, and Samy Bengio. Learnable fourier features for multi-dimensional spatial positional encoding. *Advances in Neural Information Processing Systems*, 34:15816–15829, 2021.

[21] Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Fourier neural operator for parametric partial differential equations. *arXiv preprint arXiv:2010.08895*, 2020.

[22] Jeremiah Liu, Zi Lin, Shreyas Padhy, Dustin Tran, Tania Bedrax Weiss, and Balaji Lakshminarayanan. Simple and principled uncertainty estimation with deterministic deep learning via distance awareness. *Advances in neural information processing systems*, 33:7498–7512, 2020.

[23] Jiafei Lyu, Xiaoteng Ma, Xiu Li, and Zongqing Lu. Mildly conservative q-learning for offline reinforcement learning. *Advances in Neural Information Processing Systems*, 35:1711–1724, 2022.

[24] Neil Mallinar, James Simon, Amirhesam Abedsoltan, Parthe Pandit, Misha Belkin, and Preetum Nakkiran. Benign, tempered, or catastrophic: Toward a refined taxonomy of overfitting. *Advances in neural information processing systems*, 35:1182–1195, 2022.

[25] Yixiu Mao, Hongchang Zhang, Chen Chen, Yi Xu, and Xiangyang Ji. Supported value regularization for offline reinforcement learning. *Advances in Neural Information Processing Systems*, 36:40587–40609, 2023.

[26] Yixiu Mao, Qi Wang, Chen Chen, Yun Qu, and Xiangyang Ji. Offline reinforcement learning with ood state correction and ood action suppression. *Advances in Neural Information Processing Systems*, 37:93568–93601, 2024.

[27] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. *Communications of the ACM*, 65(1):99–106, 2021.

[28] Mitsuhiko Nakamoto, Simon Zhai, Anikait Singh, Max Sobol Mark, Yi Ma, Chelsea Finn, Aviral Kumar, and Sergey Levine. Cal-ql: Calibrated offline rl pre-training for efficient online fine-tuning. *Advances in Neural Information Processing Systems*, 36:62244–62269, 2023.

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

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

[31] Ali Rahimi and Benjamin Recht. Random features for large-scale kernel machines. *Advances in neural information processing systems*, 20, 2007.

[32] Yongming Rao, Wenliang Zhao, Zheng Zhu, Jiwen Lu, and Jie Zhou. Global filter networks for image classification. *Advances in neural information processing systems*, 34:980–993, 2021.

[33] Walter Rudin. *Fourier analysis on groups*. Courier Dover Publications, 2017.

[34] Bernhard Schölkopf and Alexander J. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press, 2002.

[35] Bernhard Schölkopf, Ralf Herbrich, and Alexander J. Smola. A generalized representer theorem. In *Computational Learning Theory*, pages 416–426. Springer, 2001.

[36] Aman Sinha and John C Duchi. Learning kernels with random features. *Advances in neural information processing systems*, 29, 2016.

[37] Bharath Sriperumbudur and Zoltán Szabó. Optimal rates for random fourier features. *Advances in neural information processing systems*, 28, 2015.

[38] Ingo Steinwart. On the influence of the kernel on the consistency of support vector machines. *Journal of Machine Learning Research*, 2:67–93, 2001.

[39] HJ Terry Suh, Glen Chou, Hongkai Dai, Lujie Yang, Abhishek Gupta, and Russ Tedrake. Fighting uncertainty with gradients: Offline reinforcement learning via diffusion score matching. In *Conference on Robot Learning*, pages 2878–2904. PMLR, 2023.

[40] Matthew Tancik, Pratul Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan Barron, and Ren Ng. Fourier features let networks learn high frequency functions in low dimensional domains. *Advances in neural information processing systems*, 33:7537–7547, 2020.

[41] 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.

[42] Franki Nguimatsia Tiofack, Théotime Le Hellard, Fabian Schramm, Nicolas Perrin-Gilbert, and Justin Carpentier. Guided flow policy: Learning from high-value actions in offline reinforcement learning. *arXiv preprint arXiv:2512.03973*, 2025.

[43] Gia-Lac Tran, Edwin V Bonilla, John Cunningham, Pietro Michiardi, and Maurizio Filippone. Calibrating deep convolutional gaussian processes. In *The 22nd international conference on artificial intelligence and statistics*, pages 1554–1563. PMLR, 2019.

[44] Da Wang, Yi Ma, Ting Guo, Hongyao Tang, Wei Wei, and Jiye Liang. Fans: A flatness-aware network structure for generalization in offline reinforcement learning. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.

[45] Jialong Wu, Haixu Wu, Zihan Qiu, Jianmin Wang, and Mingsheng Long. Supported policy optimization for offline reinforcement learning. *Advances in Neural Information Processing Systems*, 35:31278–31291, 2022.

[46] K Xu, M Zhang, J Li, SS Du, K-i Kawarabayashi, and S Jegelka. How neural networks extrapolate: From feedforward to graph neural networks, arxiv. *arXiv preprint arXiv:2009.11848*, 2020.

[47] Felix Xinnan X Yu, Ananda Theertha Suresh, Krzysztof M Choromanski, Daniel N Holtmann-Rice, and Sanjiv Kumar. Orthogonal random features. *Advances in neural information processing systems*, 29, 2016.

Appendix

A Related Work

Fourier Features in Deep Learning Prior work on integrating Fourier features into neural networks can be categorized by the point of integration: input-level, layer-wise, and output-level. First, input-level approaches apply Fourier features directly to the input data, enabling networks to learn high-frequency functions over low-dimensional domains [40, 27]. Second, layer-wise approaches incorporate Fourier transforms across multiple layers to represent latent features in the frequency domain [21, 32, 7]. Third, output-level approaches apply Fourier features near the final layer, often to approximate Gaussian processes (GPs) via RFF [22, 43, 15].

SNGP [22] is a representative model of the third approach, addressing the distance-agnostic nature of ReLU-based networks by introducing a distance-aware neural model through GP approximation. Our method is related to this line of work, as the GP posterior mean shares the same functional form as KRR. However, we differ in two key aspects: we omit the posterior variance to simplify the algorithmic implementation, and we adopt a ridgeless framework that operates without a Gaussian prior, departing from the traditional motivation of GP-based approaches.

Offline Reinforcement Learning. Research in offline RL has largely focused on two central challenges: distributional shift and value overestimation. Distributional shift arises from the discrepancy between the training and testing distributions [26, 12]. Value overestimation, on the other hand, occurs when Bellman backups evaluate actions that are poorly supported by the dataset, leading to overestimated Q-values [18]. Our work focuses on the latter problem. Early methods addressed overestimation by adding conservative penalty terms to lower Q-values across the entire domain [17] or by incorporating behavior cloning objectives to constrain the action query space of the policy [5]. While these approaches can prevent extreme value instability, their restrictive nature often leads to suboptimal performance. To relax these constraints, methods such as SVR [25] and Mildly CQL [23] apply Bellman updates only within the dataset support. Distance-aware model-based approaches, including NUNO [16] and SGP [39], assign penalties according to the distance from the dataset distribution. Although these methods achieve strong performance on D4RL benchmarks, they often require additional generative or likelihood-based models, which substantially increases training complexity.

Critic network regularization in offline RL. These approaches aim for simple implementations by introducing only minimal modifications to standard actor–critic algorithms, avoiding the complex training pipelines required by support-based methods. For instance, RLPD [2] applies layer normalization before the final layer of a standard ReLU-MLP, which bounds the output magnitude by the norm of the last-layer weights. Similarly, PARS [13] introduces reward scaling during Bellman backups, artificially scaling target Q-values to encourage higher-resolution feature representations and reduce gradient influence in OOD regions. More recently, FANS [44] replaces ReLU with Gaussian activations, suppressing feature responses far from the origin and improving extrapolation when combined with zero-mean input normalization.

Although these methods are simple and effective, they still have inherent limitations. In PARS, reward scaling alone provides insufficient control over extrapolation, requiring an auxiliary OOD penalty term (PA) [13]. FANS does not provide a formal guarantee that Q-values in OOD regions remain lower than those in IND regions. Our work follows the same philosophy of structural simplicity. However, our approach differs in its underlying principle: KVR is motivated by the vanishing extrapolation behavior of kernel regression induced by shift-invariant kernels. This allows the critic to fit target values within the data support while naturally lowering estimates away from it. Through an RFF approximation, this kernel-induced extrapolation bias can be implemented efficiently within a standard actor–critic framework.

B Proofs and Analysis

B.1 Detailed Preliminaries

Theorem 2 (Bochner's Theorem [33])**.** *Let $k(\mathbf{x}, \mathbf{x}')$ be a continuous shift-invariant kernel. Then k is positive definite if and only if k is the Fourier transform of a finite nonnegative measure μ.*

To simplify the exposition, we assume $k(\mathbf{0}) = 1$, which can be achieved by rescaling the kernel. Let $\mathbb{P}$ be the probability measure obtained by normalizing μ. Then the shift-invariant positive definite kernel k can be expressed as an expectation:

$$k(\mathbf{x}, \mathbf{x}') = k(\mathbf{x} - \mathbf{x}') = \int_{\mathbb{R}^d} e^{i\boldsymbol{\xi}^\top(\mathbf{x}-\mathbf{x}')} d\mathbb{P}(\boldsymbol{\xi}) = \mathbb{E}_{\boldsymbol{\xi}\sim\mathbb{P}}\left[e^{i\boldsymbol{\xi}^\top(\mathbf{x}-\mathbf{x}')}\right]. \tag{4}$$

Based on this representation, RFFs approximate the kernel by replacing the expectation with a finite Monte Carlo average. Let $\boldsymbol{\xi}_1, \ldots, \boldsymbol{\xi}_l \overset{\text{i.i.d.}}{\sim} \mathbb{P}$. For a real-valued feature representation, define

$$\phi(\mathbf{x}) = \frac{1}{\sqrt{l}}\left[\cos(\boldsymbol{\xi}_1^\top \mathbf{x}), \sin(\boldsymbol{\xi}_1^\top \mathbf{x}), \ldots, \cos(\boldsymbol{\xi}_l^\top \mathbf{x}), \sin(\boldsymbol{\xi}_l^\top \mathbf{x})\right]^\top \in \mathbb{R}^{2l}.$$

Then the kernel can be approximated by the inner product

$$\hat{k}(\mathbf{x}, \mathbf{x}') = \phi(\mathbf{x})^\top \phi(\mathbf{x}').$$

Given a training dataset $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^N$, let $\Phi \in \mathbb{R}^{N\times 2l}$ denote the RFF design matrix whose i-th row is $\phi(\mathbf{x}_i)^\top$. The RFF approximation of KRR can then be written as

$$\hat{f}(\mathbf{x}) = \mathbf{w}^\top \phi(\mathbf{x}),$$

where the coefficient vector $\mathbf{w} \in \mathbb{R}^{2l}$ is obtained by solving

$$\mathbf{w}^\star = \operatorname*{argmin}_{\mathbf{w}\in\mathbb{R}^{2l}} \left\{\|\Phi\mathbf{w} - \mathbf{y}\|_2^2 + \lambda\|\mathbf{w}\|_2^2\right\}.$$

Thus, once the Fourier frequencies $\{\boldsymbol{\xi}_i\}_{i=1}^l$ are sampled, RFF-based KRR reduces to a finite-dimensional linear regression problem with ridge regularization. For $\lambda > 0$, its unique closed-form solution is given by

$$\mathbf{w}^\star = (\Phi^\top\Phi + \lambda\mathbf{I}_{2l})^{-1}\Phi^\top\mathbf{y}.$$

B.2 Proof of Lemma 1

Lemma 1. *Consider a shift-invariant positive definite kernel $k(\mathbf{x}, \mathbf{x}') = \kappa(\mathbf{x} - \mathbf{x}')$ with $\kappa(\mathbf{0}) = 1$ and $|\kappa(\mathbf{t})| \to 0$ as $\|\mathbf{t}\|_2 \to \infty$. Given a dataset $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^N$ with $\mathbf{x}_i \in \mathbb{R}^d$ and $y_i \in \mathbb{R}$, let $f^\star \in \mathcal{H}_k$ be the KRR predictor on $\mathcal{D}$ with regularization $\lambda > 0$, and let $\mathbf{y} = [y_1, \ldots, y_N]^\top$. Then, for any $\mathbf{x} \in \mathbb{R}^d$,*

$$|f^\star(\mathbf{x})| \le \|\mathbf{y}\|_\infty \min\left\{\sqrt{N}M_1,\ NM_2 m_k(\mathbf{x})\right\},$$

where

$$M_1 = \max_{i=1,\ldots,N} \frac{\sqrt{\sigma_i}}{\lambda + \sigma_i}, \qquad M_2 = \max_{i=1,\ldots,N} \frac{1}{\lambda + \sigma_i},$$

$\mathbf{K} = (k(\mathbf{x}_i, \mathbf{x}_j))_{i,j=1}^N$ is the kernel matrix, and $\{\sigma_i\}_{i=1}^N$ are its eigenvalues.

Proof. We first derive an upper bound for the KRR predictor f without the kernel decay term, and then derive a bound that incorporates the kernel decay.

$$
\begin{aligned}
\left|f^\star(\mathbf{x})\right| &= \left|\langle f^\star, k(\mathbf{x},\cdot)\rangle_{\mathcal{H}_k}\right| \\
&\le \|f^\star\|_{\mathcal{H}_k} \cdot \|k(\mathbf{x},\cdot)\|_{\mathcal{H}_k} \quad (\because \text{ Cauchy-Schwarz Inequality}) \\
&= \|f^\star\|_{\mathcal{H}_k} \cdot \sqrt{k(\mathbf{x},\mathbf{x})} \\
&= \|f^\star\|_{\mathcal{H}_k} \quad (\because\ k(\mathbf{x},\mathbf{x}) = k(\mathbf{0}) = 1) \\
&= \sqrt{\mathbf{y}^\top (\mathbf{K}+\lambda\mathbf{I})^{-1}\mathbf{K}(\mathbf{K}+\lambda I)^{-1}\mathbf{y}} \\
&\le \|\mathbf{y}\|_2 \cdot \max_{i=1,\dots,N}\left\{\sqrt{\frac{\sigma_i}{(\lambda+\sigma_i)^2}}\right\} \quad (\because \text{ Rayleigh Quotient}) \\
&\le \|\mathbf{y}\|_\infty \cdot \sqrt{N} \cdot \max_{i=1,\dots,N}\left\{\sqrt{\frac{\sigma_i}{(\lambda+\sigma_i)^2}}\right\}
\end{aligned}
$$

$$
\begin{aligned}
\left|f^\star(\mathbf{x})\right| &= \Big|\sum_{i=1}^{N} \alpha_i k(\mathbf{x},\mathbf{x}_i)\Big| \\
&\le m_k(\mathbf{x}) \cdot \|\boldsymbol{\alpha}\|_1 \\
&\le m_k(\mathbf{x}) \cdot \sqrt{N} \cdot \|\boldsymbol{\alpha}\|_2 \quad (\because \text{ Cauchy-Schwarz Inequality}) \\
&\le m_k(\mathbf{x}) \cdot \sqrt{N} \cdot \sqrt{\mathbf{y}^\top (\mathbf{K}+\lambda\mathbf{I})^{-2}\mathbf{y}} \\
&\le m_k(\mathbf{x}) \cdot \sqrt{N} \cdot \|\mathbf{y}\|_2 \cdot \max_{i=1,\dots,N}\left\{\frac{1}{\lambda+\sigma_i}\right\} \quad (\because \text{ Rayleigh Quotient}) \\
&\le m_k(\mathbf{x}) \cdot N \cdot \|\mathbf{y}\|_\infty \cdot \max_{i=1,\dots,N}\left\{\frac{1}{\lambda+\sigma_i}\right\}
\end{aligned}
$$

where $\alpha = (\mathbf{K}+\lambda\mathbf{I})^{-1}\mathbf{y}$ is the vector of KRR coefficients. □

B.3 Proof of Theorem 1

As a key ingredient, we restate below Theorem 1 of Sriperumbudur and Szabó [37] in our notation.

Lemma 2 (Restatement of Theorem 1 in [37])**.** *Let $k(\cdot,\cdot)$ be a positive semi-definite and shift-invariant kernel on $\mathbb{R}^d \times \mathbb{R}^d$.*

$$\nu^2 := \int \|\boldsymbol{\xi}\|^2 \, d\mathbb{P}(\boldsymbol{\xi}) < \infty.$$

Then, for any $\tau > 0$ and any non-empty compact set $\mathcal{C} \subset \mathbb{R}^d$,

$$\mathbb{P}^l\left(\left\{(\boldsymbol{\xi}_i)_{i=1}^l : \|\hat{k} - k\|_{\mathcal{C}\times\mathcal{C}} \ge \frac{\rho(d,|\mathcal{C}|,\nu) + \sqrt{2\tau}}{\sqrt{l}}\right\}\right) \le e^{-\tau},$$

where

$$\rho(d,|\mathcal{C}|,\nu) := 32\sqrt{2d\log(2|\mathcal{C}|+1)} + 32\sqrt{2d\log(\nu+1)} + 16\sqrt{\frac{2d}{\log(2|\mathcal{C}|+1)}},$$

and $|\mathcal{C}|$ denotes the diameter of $\mathcal{C}$.

This lemma states that the uniform approximation error between the true kernel k and its RFF approximation $\hat{k}$ scales as $l^{-1/2}$, with logarithmic dependence on $|\mathcal{C}|$ and ν, where l denotes the number of sampled features.

Theorem 1. *Suppose that the conditions of Lemma 1 and Lemma 2 hold. Let $\hat{k}_l$ be the RFF approximation of k using l random Fourier frequencies, and define*

$$\hat{f}_l(\mathbf{x}) = \sum_{i=1}^{N} \hat{\alpha}_{l,i}\, \hat{k}_l(\mathbf{x}, \mathbf{x}_i), \qquad \hat{\boldsymbol{\alpha}}_l = (\hat{\mathbf{K}}_l + \lambda \mathbf{I})^{-1}\mathbf{y},$$

where $\hat{\mathbf{K}}_l$ is the approximate kernel matrix induced by $\hat{k}_l$. Then the following holds.

For any compact $\mathcal{C} \subset \mathbb{R}^d$ containing $\{\mathbf{x}_i\}_{i=1}^N$ and any $\delta \in (0,1)$, there exist $C_1, C_2 > 0$, independent of l, such that, with probability at least $1-\delta$,

$$|\hat{f}_l(\mathbf{x})| \le C_1 m_k(\mathbf{x}) + C_2 l^{-1/2}, \qquad \forall\, \mathbf{x} \in \mathcal{C}.$$

Proof. Let

$$\varepsilon_l := \frac{\rho(d, |\mathcal{C}|, \nu) + \sqrt{2\log(1/\delta)}}{\sqrt{l}}.$$

By Lemma 2, with probability at least $1-\delta$,

$$\sup_{\mathbf{x},\mathbf{x}' \in \mathcal{C}} |k(\mathbf{x}, \mathbf{x}') - \hat{k}_l(\mathbf{x}, \mathbf{x}')| \le \varepsilon_l.$$

Consequently,

$$\|\hat{\mathbf{K}}_l - \mathbf{K}\|_2 \le N\varepsilon_l.$$

Let $\boldsymbol{\alpha} = (\mathbf{K} + \lambda \mathbf{I})^{-1}\mathbf{y}$. Then

$$\hat{\boldsymbol{\alpha}}_l - \boldsymbol{\alpha} = (\hat{\mathbf{K}}_l + \lambda \mathbf{I})^{-1}(\mathbf{K} - \hat{\mathbf{K}}_l)(\mathbf{K} + \lambda \mathbf{I})^{-1}\mathbf{y}.$$

Since both $\mathbf{K}$ and $\hat{\mathbf{K}}_l$ are positive semidefinite and $\lambda > 0$,

$$\|\hat{\boldsymbol{\alpha}}_l - \boldsymbol{\alpha}\|_2 \le \frac{N}{\lambda^2}\varepsilon_l \|\mathbf{y}\|_2, \qquad \|\hat{\boldsymbol{\alpha}}_l\|_2 \le \frac{1}{\lambda}\|\mathbf{y}\|_2.$$

For any $\mathbf{x} \in \mathcal{C}$,

$$\begin{aligned}
&|f^\star(\mathbf{x}) - \hat{f}_l(\mathbf{x})| \\
&= \left| \sum_{i=1}^{N} (\alpha_i - \hat{\alpha}_{l,i}) k(\mathbf{x}, \mathbf{x}_i) + \sum_{i=1}^{N} \hat{\alpha}_{l,i} \big(k(\mathbf{x}, \mathbf{x}_i) - \hat{k}_l(\mathbf{x}, \mathbf{x}_i)\big) \right| \\
&\le N\|\boldsymbol{\alpha} - \hat{\boldsymbol{\alpha}}_l\|_2 m_k(\mathbf{x}) + N\|\hat{\boldsymbol{\alpha}}_l\|_2 \varepsilon_l \\
&\le \left(\frac{N^2}{\lambda^2} m_k(\mathbf{x}) + \frac{N}{\lambda} \right) \|\mathbf{y}\|_2 \varepsilon_l.
\end{aligned}$$

By Lemma 1 and $m_k(\mathbf{x}) \le 1$,

$$\begin{aligned}
|\hat{f}_l(\mathbf{x})| &\le |f^\star(\mathbf{x})| + |f^\star(\mathbf{x}) - \hat{f}_l(\mathbf{x})| \\
&\le NM_2\|\mathbf{y}\|_\infty m_k(\mathbf{x}) + \left(\frac{N^2}{\lambda^2} m_k(\mathbf{x}) + \frac{N}{\lambda} \right) \|\mathbf{y}\|_2 \varepsilon_l \\
&\le NM_2\|\mathbf{y}\|_2 m_k(\mathbf{x}) + \left(\frac{N^2}{\lambda^2} + \frac{N}{\lambda} \right) \|\mathbf{y}\|_2 \varepsilon_l.
\end{aligned}$$

Since $\varepsilon_l = O(l^{-1/2})$, the desired bound follows with constants $C_1, C_2 > 0$ independent of l. □

B.4 Proof of Corollary 1

To prove Corollary 1, we first derive lower bounds on the latent-space distance to the training representations for ReLU-based MLPs and residual networks with LN. Under suitable conditions, these bounds ensure that latent distances diverge as inputs move arbitrarily far from the training set. Since the kernel decays at large distances, we can then bound the latent kernel similarity by a function

ψ_g of the input-space distance to the training set. Finally, applying Theorem 1 in the latent space yields the prediction bound with an additional finite-feature approximation error.

We follow the setting and notation of Corollary 1. For the residual network with LN, we assume an affine input embedding followed by J residual blocks:

$$\mathbf{z}_0(\mathbf{x}) = A\mathbf{x} + \mathbf{b},$$
$$\mathbf{z}_j(\mathbf{x}) = \mathbf{z}_{j-1}(\mathbf{x}) + F_j(\mathbf{z}_{j-1}(\mathbf{x})), \quad j = 1, \dots, J, \qquad g(\mathbf{x}) = \mathbf{z}_J(\mathbf{x}).$$

Here $A \in \mathbb{R}^{h \times d}$ has full column rank, and each branch has the structure Linear-ReLU-LN-Linear:

$$F_j(\mathbf{z}) = W_{j,2}\,\mathrm{LN}_j(\mathrm{ReLU}(W_{j,1}\mathbf{z} + \mathbf{b}_{j,1})) + \mathbf{b}_{j,2}.$$

The layer LN_j has width m_j and stabilizer $\varepsilon_j > 0$, with no affine gain or shift.

For either architecture, define the distance to the training representations as $\Delta_g(\mathbf{x}) := \min_{1 \le i \le N} \|g(\mathbf{x}) - g(\mathbf{x}_i)\|_2$. The following proposition gives a common lower bound for the two architectures.

Proposition 1. *Under either of the following conditions, there exist constants $a_g > 0$ and $b_g \ge 0$, independent of $\mathbf{x}$, such that*

$$\Delta_g(\mathbf{x}) \ge [a_g d_{\mathcal{D}}(\mathbf{x}) - b_g]_+, \qquad \mathbf{x} \in \mathbb{R}^d, \tag{5}$$

where $[t]_+ := \max\{t, 0\}$.

1. ***ReLU-based MLP.*** *Suppose that g is a finite MLP constructed from affine layers and ReLU activations. Let g_∞ be the network obtained by setting all biases to zero, and assume*

$$c_g := \min_{\|\mathbf{v}\|_2 = 1} \|g_\infty(\mathbf{v})\|_2 > 0.$$

Then (5) *holds with $a_g = c_g$ and a finite constant $b_g = C_{\mathcal{D}}$ defined in the proof.*

2. ***Residual network with LN.*** *Suppose that g has the residual architecture specified above. Then* (5) *holds with*

$$a_g = \sigma_{\min}(A) := \sqrt{\lambda_{\min}(A^\top A)} > 0, \qquad b_g = 2B_{\mathrm{res}}, \qquad B_{\mathrm{res}} := \sum_{j=1}^{J} \sqrt{m_j} \|W_{j,2}\|_2.$$

Proof. **ReLU-based MLP.** Let ρ_j be ReLU or the identity, and write the layer outputs of g and its zero-bias counterpart as

$$\mathbf{h}_j = \rho_j(W_j \mathbf{h}_{j-1} + \mathbf{b}_j), \qquad \mathbf{h}_j^\infty = \rho_j(W_j \mathbf{h}_{j-1}^\infty), \qquad j = 1, \dots, L,$$

where $\mathbf{h}_0 = \mathbf{h}_0^\infty = \mathbf{x}$, $\mathbf{h}_L = g(\mathbf{x})$, and $\mathbf{h}_L^\infty = g_\infty(\mathbf{x})$. For $e_j := \|\mathbf{h}_j - \mathbf{h}_j^\infty\|_2$, we have

$$\begin{aligned}
e_j &= \|\rho_j(W_j \mathbf{h}_{j-1} + \mathbf{b}_j) - \rho_j(W_j \mathbf{h}_{j-1}^\infty)\|_2 \\
&\le \|W_j(\mathbf{h}_{j-1} - \mathbf{h}_{j-1}^\infty) + \mathbf{b}_j\|_2 \quad (\rho_j \text{ is 1-Lipschitz}) \\
&\le \|W_j\|_2 \|\mathbf{h}_{j-1} - \mathbf{h}_{j-1}^\infty\|_2 + \|\mathbf{b}_j\|_2 \\
&= \|W_j\|_2 e_{j-1} + \|\mathbf{b}_j\|_2 \\
&\le \sum_{r=1}^{j} \|\mathbf{b}_r\|_2 \prod_{k=r+1}^{j} \|W_k\|_2 \quad (\text{iteration with } e_0 = 0).
\end{aligned}$$

In particular,

$$\|g(\mathbf{x}) - g_\infty(\mathbf{x})\|_2 = e_L \le \sum_{j=1}^{L} \|\mathbf{b}_j\|_2 \prod_{k=j+1}^{L} \|W_k\|_2 =: B_g, \tag{6}$$

where an empty product equals 1. Set $R_{\mathcal{D}} := \max_i \|\mathbf{x}_i\|_2$ and $M_{\mathcal{D}} := \max_i \|g(\mathbf{x}_i)\|_2$. Then

$$\begin{aligned}
\Delta_g(\mathbf{x}) &= \min_i \|g(\mathbf{x}) - g(\mathbf{x}_i)\|_2 \\
&\geq \min_i \big(\|g(\mathbf{x})\|_2 - \|g(\mathbf{x}_i)\|_2\big) \quad \text{(reverse triangle inequality)} \\
&= \|g(\mathbf{x})\|_2 - M_{\mathcal{D}} \\
&\geq \|g_\infty(\mathbf{x})\|_2 - B_g - M_{\mathcal{D}} \quad \text{(by (6))} \\
&\geq c_g\|\mathbf{x}\|_2 - B_g - M_{\mathcal{D}} \quad \text{(positive homogeneity)} \\
&\geq c_g\big(d_{\mathcal{D}}(\mathbf{x}) - R_{\mathcal{D}}\big) - B_g - M_{\mathcal{D}} \\
&= c_g d_{\mathcal{D}}(\mathbf{x}) - C_{\mathcal{D}},
\end{aligned}$$

where the penultimate inequality uses $d_{\mathcal{D}}(\mathbf{x}) \leq \|\mathbf{x}\|_2 + R_{\mathcal{D}}$ and $C_{\mathcal{D}} := c_g R_{\mathcal{D}} + B_g + M_{\mathcal{D}}$. Since $\Delta_g(\mathbf{x}) \geq 0$,

$$\Delta_g(\mathbf{x}) \geq [c_g d_{\mathcal{D}}(\mathbf{x}) - C_{\mathcal{D}}]_+,$$

which proves the bound with $a_g = c_g$ and $b_g = C_{\mathcal{D}}$.

Residual network with LN. For $\mathbf{u} \in \mathbb{R}^{m_j}$, let $\mathbf{u}_c := \mathbf{u} - m_j^{-1}(\mathbf{1}^\top \mathbf{u})\mathbf{1}$. Since LN has no affine gain or shift,

$$\begin{aligned}
\|\operatorname{LN}_j(\mathbf{u})\|_2^2 &= \left\| \frac{\mathbf{u}_c}{\sqrt{m_j^{-1}\|\mathbf{u}_c\|_2^2 + \varepsilon_j}} \right\|_2^2 \\
&= \frac{\|\mathbf{u}_c\|_2^2}{m_j^{-1}\|\mathbf{u}_c\|_2^2 + \varepsilon_j} \\
&\leq m_j \quad (\varepsilon_j > 0).
\end{aligned}$$

Writing $r_j(\mathbf{z}) := \operatorname{ReLU}(W_{j,1}\mathbf{z} + \mathbf{b}_{j,1})$, we obtain

$$\begin{aligned}
\|F_j(\mathbf{z}) - F_j(\mathbf{z}')\|_2 &= \big\|W_{j,2}\big(\operatorname{LN}_j(r_j(\mathbf{z})) - \operatorname{LN}_j(r_j(\mathbf{z}'))\big)\big\|_2 \\
&\leq \|W_{j,2}\|_2 \|\operatorname{LN}_j(r_j(\mathbf{z})) - \operatorname{LN}_j(r_j(\mathbf{z}'))\|_2 \\
&\leq \|W_{j,2}\|_2 \big(\|\operatorname{LN}_j(r_j(\mathbf{z}))\|_2 + \|\operatorname{LN}_j(r_j(\mathbf{z}'))\|_2\big) \\
&\leq 2\sqrt{m_j}\|W_{j,2}\|_2.
\end{aligned}$$

For $\mathbf{z}_j = \mathbf{z}_j(\mathbf{x})$ and $\mathbf{z}'_j = \mathbf{z}_j(\mathbf{x}')$, expanding the residual updates gives

$$\begin{aligned}
\|g(\mathbf{x}) - g(\mathbf{x}')\|_2 &= \left\| A(\mathbf{x} - \mathbf{x}') + \sum_{j=1}^{J} \big(F_j(\mathbf{z}_{j-1}) - F_j(\mathbf{z}'_{j-1})\big) \right\|_2 \\
&\geq \|A(\mathbf{x} - \mathbf{x}')\|_2 - \sum_{j=1}^{J} \|F_j(\mathbf{z}_{j-1}) - F_j(\mathbf{z}'_{j-1})\|_2 \\
&\geq \sigma_{\min}(A)\|\mathbf{x} - \mathbf{x}'\|_2 - 2\sum_{j=1}^{J} \sqrt{m_j}\|W_{j,2}\|_2 \\
&= \sigma_{\min}(A)\|\mathbf{x} - \mathbf{x}'\|_2 - 2B_{\mathrm{res}}.
\end{aligned} \tag{7}$$

Taking the minimum over the training inputs and using $\Delta_g(\mathbf{x}) \geq 0$,

$$\begin{aligned}
\Delta_g(\mathbf{x}) &\geq \Big[\min_i \big\{\sigma_{\min}(A)\|\mathbf{x} - \mathbf{x}_i\|_2 - 2B_{\mathrm{res}}\big\}\Big]_+ \\
&= \Big[\sigma_{\min}(A) \min_i \|\mathbf{x} - \mathbf{x}_i\|_2 - 2B_{\mathrm{res}}\Big]_+ \\
&= [\sigma_{\min}(A) d_{\mathcal{D}}(\mathbf{x}) - 2B_{\mathrm{res}}]_+.
\end{aligned}$$

Thus, $a_g = \sigma_{\min}(A) > 0$ and $b_g = 2B_{\mathrm{res}}$. □

Corollary 1. *Under the common setup and either condition in Proposition 1, there exists a non-increasing function $\psi_g : [0,\infty) \to [0,\infty)$ satisfying $\psi_g(t) \to 0$ as $t \to \infty$ such that, for any compact set $\mathcal{C} \subset \mathbb{R}^d$ containing the training inputs and any $\delta \in (0,1)$, there exist constants $C_1, C_2 > 0$, independent of l, such that, for each $l \geq 1$, with probability at least $1-\delta$,*

$$|\hat{f}_{\mathrm{NN},l}(\mathbf{x})| \leq C_1 \psi_g(d_{\mathcal{D}}(\mathbf{x})) + C_2 l^{-1/2}, \qquad \forall \mathbf{x} \in \mathcal{C}.$$

Proof. Define

$$\bar{\kappa}(t) := \sup_{\substack{\mathbf{u}\in\mathbb{R}^h \\ \|\mathbf{u}\|_2 \geq t}} |\kappa(\mathbf{u})|, \qquad \psi_g(t) := \bar{\kappa}([a_g t - b_g]_+), \qquad t \geq 0.$$

By positive semidefiniteness and normalization,

$$\begin{aligned} |\kappa(\mathbf{u})| &= |k(\mathbf{u},\mathbf{0})| \\ &\leq \sqrt{k(\mathbf{u},\mathbf{u})k(\mathbf{0},\mathbf{0})} \\ &= 1. \end{aligned}$$

For $0 \leq s \leq t$, the definitions give

$$\begin{aligned} \bar{\kappa}(s) &= \sup_{\|\mathbf{u}\|_2 \geq s} |\kappa(\mathbf{u})| \geq \sup_{\|\mathbf{u}\|_2 \geq t} |\kappa(\mathbf{u})| = \bar{\kappa}(t), \\ \psi_g(s) &= \bar{\kappa}([a_g s - b_g]_+) \geq \bar{\kappa}([a_g t - b_g]_+) = \psi_g(t). \end{aligned}$$

The kernel decay assumption and $a_g > 0$ further imply

$$\bar{\kappa}(t) \longrightarrow 0, \qquad [a_g t - b_g]_+ \longrightarrow \infty, \qquad \psi_g(t) \longrightarrow 0 \quad (t \to \infty).$$

Thus, ψ_g is nonnegative, non-increasing, and vanishes at infinity; its definition is independent of l, $\mathcal{C}$, and δ.

For any $\mathbf{x} \in \mathbb{R}^d$,

$$\begin{aligned} \max_i |k(g(\mathbf{x}), g(\mathbf{x}_i))| &= \max_i |\kappa(g(\mathbf{x}) - g(\mathbf{x}_i))| \\ &\leq \max_i \bar{\kappa}(\|g(\mathbf{x}) - g(\mathbf{x}_i)\|_2) \\ &= \bar{\kappa}\Big(\min_i \|g(\mathbf{x}) - g(\mathbf{x}_i)\|_2\Big) \quad (\bar{\kappa} \text{ is non-increasing}) \\ &= \bar{\kappa}(\Delta_g(\mathbf{x})) \\ &\leq \bar{\kappa}([a_g d_{\mathcal{D}}(\mathbf{x}) - b_g]_+) \quad \text{(Proposition 1)} \\ &= \psi_g(d_{\mathcal{D}}(\mathbf{x})). \end{aligned}$$

Fix $\mathcal{C}$ and δ as in the statement. Since g is continuous, $g(\mathcal{C})$ is compact; choose a compact ball $\mathcal{Z}_{\mathcal{C}} \supset g(\mathcal{C})$ of positive radius, independently of the sampled frequencies. Applying Theorem 1 to the mapped dataset on $\mathcal{Z}_{\mathcal{C}}$ gives constants $C_1, C_2 > 0$, independent of l, such that, for each $l \geq 1$, with probability at least $1-\delta$,

$$\begin{aligned} |\hat{f}_{\mathrm{NN},l}(\mathbf{x})| &\leq C_1 \max_i |k(g(\mathbf{x}), g(\mathbf{x}_i))| + C_2 l^{-1/2} \\ &\leq C_1 \bar{\kappa}(\Delta_g(\mathbf{x})) + C_2 l^{-1/2} \\ &\leq C_1 \bar{\kappa}([a_g d_{\mathcal{D}}(\mathbf{x}) - b_g]_+) + C_2 l^{-1/2} \\ &= C_1 \psi_g(d_{\mathcal{D}}(\mathbf{x})) + C_2 l^{-1/2}, \qquad \forall \mathbf{x} \in \mathcal{C}. \end{aligned}$$

□

B.5 Relation Between Finite-Stopped Gradient Descent and Ridge Regression

We adopt the RFF regression formulation and notation introduced in Appendix B.1. The following result characterizes the relationship between gradient descent stopped after finitely many iterations and ridge regression.

Proposition 2. *Let w_t denote the parameter of a linear model trained by gradient descent with learning rate η for t iterations, and let $w^\star_{\text{ridge}}$ denote the parameter of the same linear model trained by ridge regression with regularization parameter λ. Assume that $w_0 = 0$, that $\Phi^\top\Phi$ has positive eigenvalues σ_i, and that η is sufficiently small so that $0 < 1 - 2\eta\sigma_i < 1$ for each i. Let $\Phi^\top\Phi = Q^\top\Sigma Q$, and define*

$$w'_t := Qw_t, \qquad w'_{\text{ridge}} := Qw^\star_{\text{ridge}}.$$

Then the i-th spectral component of the finite-stopped gradient descent solution coincides with that of the ridge regression solution:

$$w'_{t,i} = w'_{\text{ridge},i} \qquad \textit{if} \qquad \lambda = \frac{\sigma_i(1-2\eta\sigma_i)^t}{1-(1-2\eta\sigma_i)^t}.$$

Proof. First we express w_t in recursive form and then derive a closed-form expression for w_t in terms of the eigendecomposition of $\Phi^\top\Phi$.

$$\begin{aligned}
\mathcal{L} &= \|\mathbf{y} - \Phi w\|^2 \\
\nabla_w \mathcal{L} &= -2\Phi^\top(\mathbf{y} - \Phi w) \\
w_{t+1} &= w_t - \eta\nabla_w\mathcal{L} \\
&= w_t + 2\eta\Phi^\top(\mathbf{y} - \Phi w_t) \\
&= \underbrace{(I - 2\eta\Phi^\top\Phi)}_{=:A} w_t + \underbrace{2\eta\Phi^\top\mathbf{y}}_{=:B} \\
&= Aw_t + B
\end{aligned}$$

$$\begin{aligned}
w_0 &= 0 \\
w_1 &= B \\
w_2 &= Aw_1 + B \\
w_3 &= Aw_2 + B = A^2w_1 + AB + B \\
&\cdots \\
w_t &= \sum_{j=0}^{t-1} A^j B \\
(I - A)w_t &= (I - A^t)B
\end{aligned}$$

Let $\Phi^\top\Phi = Q^\top\Sigma Q$ be the eigendecomposition of $\Phi^\top\Phi$:

$$\begin{aligned}
(I - A) &= 2\eta\Phi^\top\Phi = 2\eta Q^\top\Sigma Q \\
(I - A^t) &= I - (I - 2\eta\Phi^\top\Phi)^t = Q^\top(I - (I - 2\eta\Sigma)^t)Q
\end{aligned}$$

$$\begin{aligned}
2\eta Q^\top\Sigma Q w_t &= 2\eta Q^\top(I - (I - 2\eta\Sigma)^t)Q\Phi^\top\mathbf{y} \\
\Sigma Q w_t &= (I - (I - 2\eta\Sigma)^t)Q\Phi^\top\mathbf{y} \\
\Sigma w'_t &= (I - (I - 2\eta\Sigma)^t)\mathbf{y}' \qquad \text{where} \quad w'_t = Qw_t,\ \mathbf{y}' = Q\Phi^\top\mathbf{y} \\
w'_{t,i} &= \frac{1 - (1 - 2\eta\sigma_i)^t}{\sigma_i} y'_i
\end{aligned}$$

For the ridge regression solution, we have

$$\begin{aligned} w^\star_{\text{ridge}} &= (\Phi^\top \Phi + \lambda I)^{-1} \Phi^\top \mathbf{y} \\ &= Q^\top (\Sigma + \lambda I)^{-1} Q \Phi^\top \mathbf{y} \\ Q w^\star_{\text{ridge}} &= (\Sigma + \lambda I)^{-1} Q \Phi^\top \mathbf{y} \\ w'_{\text{ridge},i} &= \frac{1}{\sigma_i + \lambda} y'_i \qquad \text{where} \quad w'_{\text{ridge}} = Q w^\star_{\text{ridge}}. \end{aligned}$$

Therefore, if $\lambda = \frac{\sigma_i (1-2\eta\sigma_i)^t}{1-(1-2\eta\sigma_i)^t}$, we have

$$\begin{aligned} w'_{t,i} &= \frac{1-(1-2\eta\sigma_i)^t}{\sigma_i} y'_i \\ &= \frac{1}{\sigma_i + \lambda} y'_i \\ &= w'_{\text{ridge},i} \end{aligned}$$

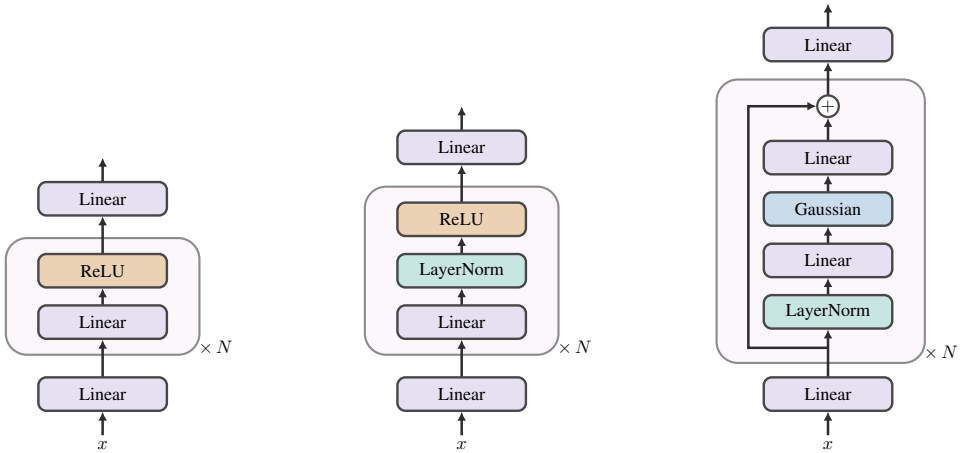
□

Proposition 2 shows that finite stopping in gradient descent induces an eigendirection-wise regularization effect analogous to ridge regression in the RFF approximation setting. In particular, along each eigendirection of $\Phi^\top \Phi$, the t-step gradient descent iterate applies a spectral shrinkage factor of $1-(1-2\eta\sigma_i)^t$, while ridge regression applies the shrinkage factor $\sigma_i/(\sigma_i+\lambda)$. The Proposition identifies the corresponding regularization level under which these two quantities coincide for the i-th spectral component. This characterization provides a spectral interpretation of finite stopping as an implicit regularization mechanism, and explains why finite-stopped optimization under the RFF approximation can play a role analogous to explicit ℓ_2 regularization.

C Further Analyses of the Didactic Example

In this section, we provide a more detailed analysis of kernels and their RFF approximations using the simple didactic regression example, $y = |x|$ for $x \in (-1, 1)$, introduced in the main text.

C.1 Detailed Baseline Algorithms



(a) ReLU-MLP (b) MLP+LN (c) FANS

Figure 9: **Network architectures of the baseline models.**

To clarify the differences between our approach and the baselines used in the didactic regression and offline RL experiments, we provide a detailed description of each algorithm. Figures 9 and 18 illustrate the network architectures of the baselines and our models, respectively.

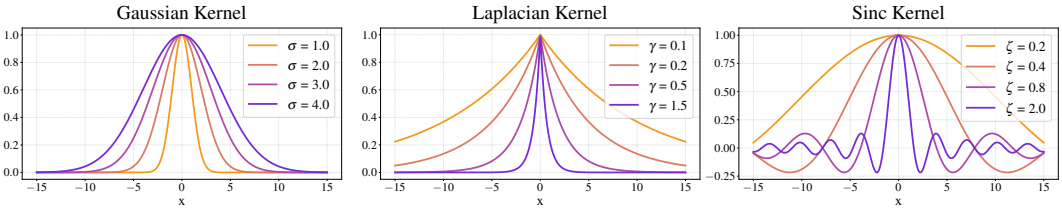


Figure 10: **Kernel functions across varying bandwidth values.** The plots illustrate the behavior of Gaussian, Laplacian, and Sinc kernels (from left to right) with respect to their bandwidth parameters, denoted by σ, $1/\gamma$, and $1/\zeta$, respectively.

Layer Normalization and Reward Scaling. RLPD [2] addresses overestimation arising from the linear extrapolation of ReLU-based MLPs by applying LN immediately before the final output layer. This modification prevents predictions from exploding by bounding the network output in terms of the final-layer weight norm. Specifically, under the assumption of no bias in the final layer and LN with no affine gain or shift, the prediction is bounded as

$$\begin{aligned} |f_{\theta,\mathbf{w}}(x)| &= \left|\mathrm{ReLU}(\mathrm{LN}(g_\theta(x)))^\top \mathbf{w}\right| \\ &\leq \|\mathrm{ReLU}(\mathrm{LN}(g_\theta(x)))\|_2 \, \|\mathbf{w}\|_2 \\ &\leq \|\mathrm{LN}(g_\theta(x))\|_2 \, \|\mathbf{w}\|_2 \\ &\leq \sqrt{h} \, \|\mathbf{w}\|_2, \end{aligned}$$

where h denotes the hidden dimension.

Building on this architecture, reward scaling technique scales the target values by a factor α, based on the observation that larger target scales encourage higher-resolution feature representations [13]. Specifically, the model is trained by minimizing the following mean squared error (MSE) loss:

$$\mathcal{L}(\theta; \{\mathbf{x}, \mathbf{y}\}) = \frac{1}{N} \sum_{i=1}^{N} \left(f_\theta(x_i) - \alpha y_i\right)^2,$$

where $\{\mathbf{x}, \mathbf{y}\}$ denotes the training dataset.

FANS. FANS [44] improves extrapolation behavior by replacing the standard ReLU activation with a Gaussian activation function:

$$\text{gauss_act}(x) = \exp(-x^2).$$

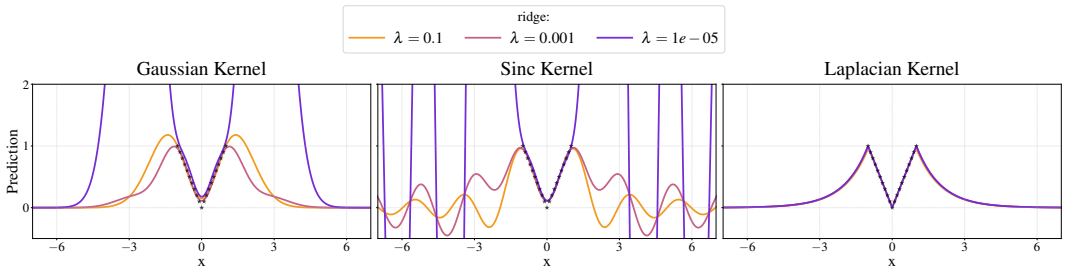
They argue that the Gaussian activation function progressively deactivates features whose values reside far from the origin, thereby effectively suppressing extrapolation error. This mechanism is particularly effective when the input data are normalized to have zero mean and unit variance.

C.2 Relationship between Probability Measures and Kernels

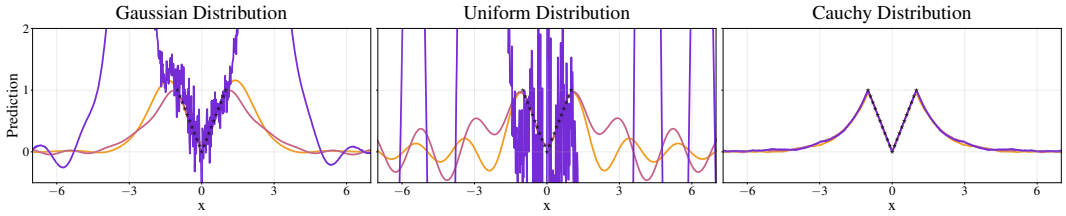
Table 3: **Kernels and corresponding spectral distributions via Bochner's Theorem.**

Relationship	**Kernel** $k(\Delta)$	**Distribution** $p(\omega)$
Gaussian $k(\Delta;\sigma) \leftrightarrow \mathcal{N}(\mathbf{0}, \sigma^{-2}\mathbf{I}_d)$	$\exp\left(-\frac{\|\Delta\|_2^2}{2\sigma^2}\right)$	$\left(\frac{\sigma^2}{2\pi}\right)^{\frac{d}{2}} \exp\left(-\frac{\sigma^2\|\omega\|_2^2}{2}\right)$
Laplacian $k(\Delta;\gamma) \leftrightarrow \prod_{j=1}^{d} \text{Cauchy}(0,\gamma)$	$\exp(-\gamma\|\Delta\|_1)$	$\prod_{j=1}^{d} \frac{\gamma}{\pi(\gamma^2+\omega_j^2)}$
Sinc $k(\Delta;\zeta) \leftrightarrow \mathcal{U}[-\zeta,\zeta]^d$	$\prod_{j=1}^{d} \frac{\sin(\zeta\Delta_j)}{\zeta\Delta_j}$	$\begin{cases}(2\zeta)^{-d}, & \|\omega\|_\infty \leq \zeta \\ 0, & \text{otherwise}\end{cases}$

In this section, we provide a more detailed discussion of the relationship between kernel selection and the probability measures used to sample Fourier frequencies, as these details are omitted from the



(a) KRR



(b) KRR approximated by RFF

Figure 11: **Ridge and stability.** Instability increases with decreasing ridge for most kernels, but the Laplacian kernel stays robust across all ridge settings.

main text. As discussed in Appendix B.1, Bochner's Theorem states that any shift-invariant positive definite kernel can be represented as the Fourier transform of a unique non-negative measure [33]. In this work, we consider Gaussian, Cauchy, and uniform spectral distributions, which correspond to the Gaussian, Laplacian, and Sinc kernels, respectively. Their analytical forms are summarized in Table 3. It is important to note how the parameters in each distribution relate to their corresponding kernels.

We use a bandwidth parameter to control the spatial scale of kernel correlations. For the Gaussian kernel, the bandwidth corresponds to the standard deviation σ. For the Laplacian and Sinc kernels, we define the bandwidth as the inverse scale parameters, $1/\gamma$ and $1/\zeta$, respectively. Figure 10 illustrates how these kernels vary with their parameters: as the bandwidth increases, the kernel becomes wider.

C.3 Influence of Ridge Regularization

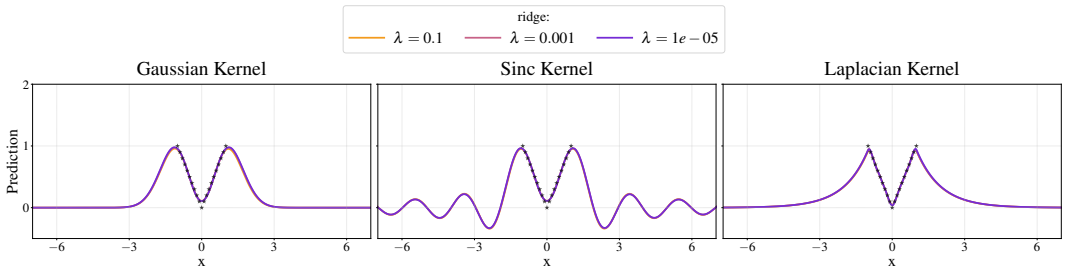
Lemma 1 and Theorem 1 provide bounds on the prediction magnitudes of KRR and RFF-KRR for any $\lambda > 0$. These upper bounds depend on the kernel spectrum and can increase as λ decreases. However, an increase in the upper bounds does not necessarily mean that the predictions themselves become larger or unstable. We therefore empirically examine how ridge regularization affects prediction behavior at a fixed dataset size.

We fix the number of data points at $n = 100$ and vary the ridge parameter λ. Figure 11 (a) shows the results for each kernel. The estimates for Gaussian and Sinc kernels become increasingly unstable as λ decreases. In contrast, the Laplacian kernel remains stable across ridge values. This stability is attributed to the slower eigenvalue decay of the Laplacian kernel, which enables stable learning even in the ridgeless setting [24]. Accordingly, KRR with the Laplacian kernel is described as exhibiting *tempered overfitting*, whereas the Gaussian kernel is associated with *catastrophic overfitting*, as its error bounds tend to diverge [24].

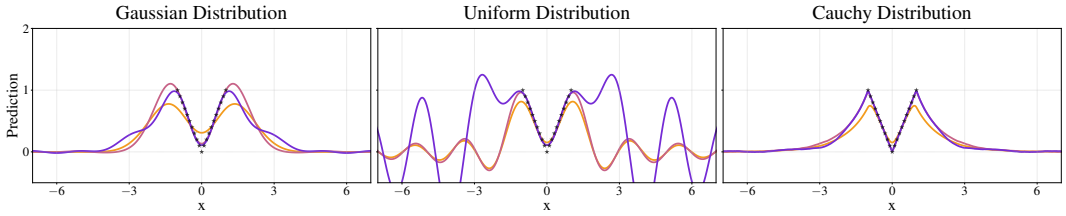
Figure 11 (b) illustrates KRR estimates using RFF approximations to these kernels. The RFF results closely mirror the behavior of the corresponding exact kernels. In particular, approximations based on the Cauchy distribution, which correspond to the Laplacian kernel, remain stable as the ridge parameter decreases, consistent with exact KRR.

C.4 Implicit Regularization via Finite-Step Gradient Descent

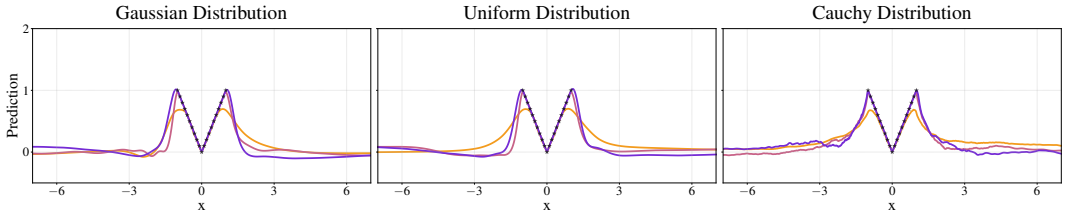
Section 3.2 discusses the challenges associated with explicit ridge regularization when adapting KRR to offline RL. As the explicit ridge decreases, the theoretical bounds become increasingly



(a) KRR learned by gradient descent



(b) KRR approximated by RFF and learned by gradient descent



(c) RFF-approximated KRR optimized via gradient descent within a neural network.

Figure 12: **Stability of KRR under gradient descent across varying ridge.** When trained via gradient descent, KRR exhibits more stable estimates across all ridges compared to the closed-form solutions.

loose (we examine this behavior empirically in the preceding subsection). Proposition 2 provides a complementary perspective on the role of finite-step optimization. It shows that finite-step gradient descent induces a spectral shrinkage effect analogous to ridge regularization, motivating its use as an implicit regularization mechanism in the ridgeless setting.

To examine this behavior empirically, we first consider finite-step optimization of the KRR coefficients $\boldsymbol{\alpha}$ under the predictor

$$f(x) = \sum_{i=1}^{N} \alpha_i \, k(x, x_i).$$

Rather than solving the corresponding KRR linear system in closed form, we obtain $\boldsymbol{\alpha}$ by minimizing

$$\mathcal{L}(\boldsymbol{\alpha}) = \|(\mathbf{K} + \lambda \mathbf{I})\, \boldsymbol{\alpha} - \mathbf{y}\|_2^2 .$$

As shown in Figure 12 (a), finite-step optimization yields stable estimates across the evaluated kernels and ridge values.

We next consider the RFF approximation of KRR, which takes the form of linear regression:

$$\hat{f}(x) = \phi(x)^\top \mathbf{w}.$$

We optimize the coefficient vector $\mathbf{w}$ using finite-step optimization by minimizing

$$\mathcal{L}(\mathbf{w}) = \|\Phi \mathbf{w} - \mathbf{y}\|_2^2 + \lambda \|\mathbf{w}\|_2^2.$$

As shown in Figure 12 (b), the RFF estimates exhibit more pronounced oscillations than their exact-kernel counterparts, particularly under uniform sampling. Nevertheless, finite-step optimization substantially reduces the instabilities observed in the corresponding closed-form solutions.

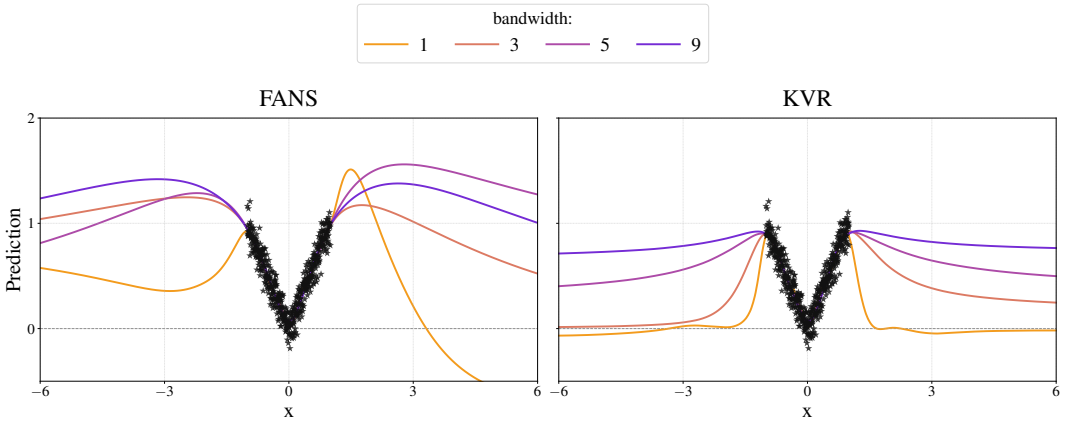


Figure 13: **Comparison of KVR and FANS across various bandwidths on the didactic example.**

We next extend the RFF model by applying the Fourier features to a learned neural representation. Given a neural network $g_\theta : \mathbb{R} \to \mathcal{H}$, where $\mathcal{H}$ denotes the latent feature space, we define

$$\bar{f}(x) = \phi(g_\theta(x))^\top \mathbf{w},$$

and jointly optimize θ and $\mathbf{w}$ by minimizing

$$\left\|\phi(g_\theta(\mathbf{x}))^\top \mathbf{w} - \mathbf{y}\right\|_2^2 + \lambda \|\mathbf{w}\|_2^2.$$

As shown in Figure 12(c), the neural model produces stable estimates with Gaussian and uniform sampling, including at small ridge values. This behavior is observed even with a feature width of 1024, compared with 10^5 features in the linear RFF setting. One notable difference from the preceding experiments is that Cauchy sampling now produces more pronounced fluctuations than Gaussian or uniform sampling. A plausible explanation is that the heavy-tailed Cauchy distribution can generate large frequencies, which can amplify gradients with respect to the learned representation and make optimization more sensitive. By contrast, uniform sampling bounds the frequency magnitude by construction, which may contribute to the more stable behavior observed in the neural setting.

Notably, both (b) and (c) of Figure 12 show that larger ridge values can shrink predictions below the target function even within the data support. This in-support shrinkage is consistent with the performance degradation observed under explicit ridge regularization in our offline RL experiments.

C.5 Effect of Bandwidth on Extrapolation

The kernel bandwidth determines how far the model extrapolates beyond the observed data: larger bandwidths allow broader extrapolation, whereas smaller bandwidths localize it more strongly around the observed data. This bandwidth-based control is conceptually related to FANS, as both methods regulate the range of functional influence through a scale parameter. In our method, the bandwidth determines the range of kernel influence, whereas in FANS, the standard deviation σ controls the range of the Gaussian activation:

$$\text{gauss_act}(\mathbf{x}; \sigma) = \exp\left(-\frac{\mathbf{x}^2}{\sigma^2}\right).$$

To compare these two mechanisms empirically, we conduct experiments on 100 noisy data points using a 3-layer MLP. For KVR, Fourier frequencies are sampled from a Gaussian distribution and kept fixed. We vary the Gaussian kernel bandwidth, with frequencies sampled according to its corresponding spectral distribution. For FANS, we vary the standard deviation of the Gaussian activation.

As shown in Figure 13, FANS exhibits inconsistent and unstable behavior outside the data support across bandwidth settings. By contrast, the RFF model exhibits stable and consistent extrapolation behavior across bandwidth settings, with larger bandwidths producing broader extrapolation and smaller bandwidths yielding more localized predictions around the data support.

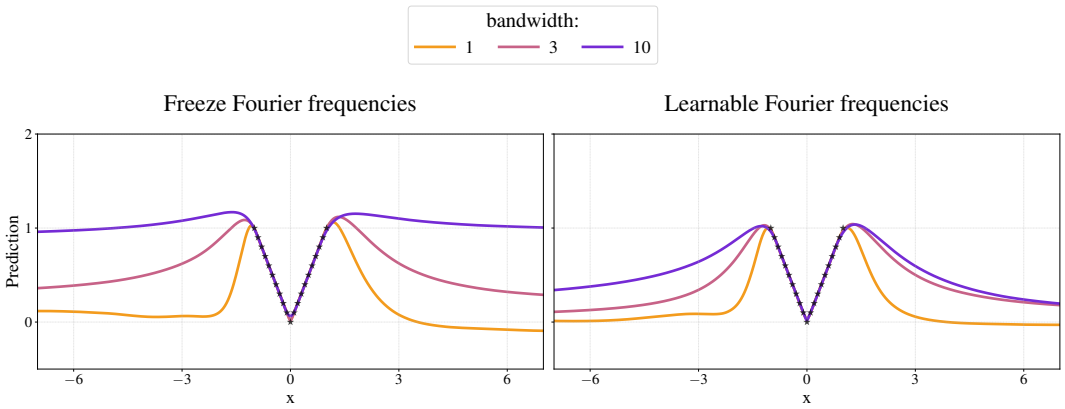


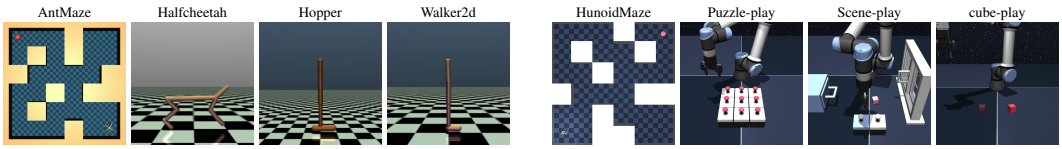
Figure 14: **KVR predictions for fixed and learnable Fourier frequencies across various bandwidths on the didactic example.**

C.6 Learnable Fourier frequencies

In the previous section, we analyzed extrapolation behavior across kernel bandwidths with fixed Fourier frequencies. In practice, however, we treat these frequencies as learnable parameters and optimize them with a small learning rate, reducing sensitivity to their initial random samples. The bandwidth therefore determines the initial frequency distribution, while the frequencies can subsequently adapt through optimization. This raises a natural question: how does extrapolation behavior change when the frequencies are learned? As shown in Figure 14, the model continues to suppress extrapolated predictions when the Fourier frequencies are learned. Although the differences across bandwidths become less pronounced after optimization, the desired extrapolation behavior remains visible over the evaluated range. These results suggest that allowing the Fourier frequencies to adapt does not eliminate the characteristic extrapolation behavior.

D Experiments

D.1 Benchmark Details



(a) D4RL (b) OGBench

Figure 15: D4RL and OGBench benchmarks.

D.1.1 D4RL

D4RL [4] was introduced in the early stage of offline RL and remains a standard benchmark for evaluating the general performance of offline RL algorithms. Although some tasks have become saturated over time, D4RL is still widely used due to its diverse task suite and standardized evaluation protocol. While the benchmark encompasses diverse environments including Maze2D, Adroit, FrankaKitchen, Flow, and Offline CARLA, this work focuses on AntMaze and Gym-MuJoCo tasks.

AntMaze. AntMaze features a quadrupedal ant robot tasked with navigating from a starting position to a designated goal state. The environments are categorized by maze size into *umaze*, *medium*, and *large*. The datasets also differ in their data collection strategies. In *umaze*, all trajectories share a fixed starting position, whereas *diverse* uses randomized start and goal locations to increase task complexity. The *play* datasets consist of trajectories collected between hand-selected start and goal locations.

Table 4: **Comparison between 5 and 24 random seeds on D4RL.** Values in parentheses indicate the relative change in performance when increasing from 5 to 24 learning seeds.

D4RL	5 random seeds	24 random seeds	
`halfcheetah-random`	30.3	29.3	(-3.3%)
`hopper-random`	32.3	32.3	(0.0%)
`walker2d-random`	21.9	21.4	(-2.3%)
`halfcheetah-medium`	64.7	64.8	(+0.2%)
`hopper-medium`	104.1	104.2	(+0.1%)
`walker2d-medium`	96.6	96.4	(-0.2%)
`halfcheetah-medium-replay`	55.2	55.1	(-0.2%)
`hopper-medium-replay`	102.5	103.0	(+0.5%)
`walker2d-medium-replay`	102.0	95.5	(-6.4%)
`halfcheetah-medium-expert`	106.2	105.4	(-0.8%)
`hopper-medium-expert`	113.0	112.9	(-0.1%)
`walker2d-medium-expert`	113.1	110.5	(-2.3%)
`antmaze-umaze`	99.0	99.1	(0.1%)
`antmaze-umaze-diverse`	98.4	95.2	(-3.3%)
`antmaze-medium-play`	94.0	93.3	(-0.7%)
`antmaze-medium-diverse`	94.8	93.8	(-1.1%)
`antmaze-large-play`	93.8	91.0	(-3.0%)
`antmaze-large-diverse`	92.4	91.8	(-0.6%)
`Average`	84.1	83.1	(-1.2%)

Gym-MuJoCo. Gym-MuJoCo datasets are constructed from the standard OpenAI Gym MuJoCo environments and are highly relevant to the core objective of offline RL, which is to derive an optimal policy from a static, suboptimal dataset. The datasets are categorized by the quality of the behavior policy used for data collection. The *random* dataset is collected from a randomly initialized policy, *medium* from an intermediate policy during training, and *expert* from a fully trained policy. Datasets denoted by **-replay* contain the replay buffer accumulated during training up to a specified performance level. Mixed datasets, such as *medium-expert* and *medium-random*, combine trajectories from policies of different quality levels to further diversify the data distribution.

Normalized Score. D4RL reports a normalized score to enable standardized comparison across tasks with different return scales. The score maps the performance of a random policy to 0 and that of an expert policy to 100, based on their average returns.

D.1.2 OGBench

OGBench [29] is a large-scale benchmark originally designed for offline goal-conditioned reinforcement learning (GCRL), comprising 8 environment types and 85 datasets. While it provides maze environments analogous to D4RL-AntMaze, OGBench introduces significantly greater challenges by incorporating humanoid agents with high-dimensional action spaces and *giant* mazes that exceed the scale of D4RL's largest environments. To evaluate diverse algorithmic capabilities, the benchmark provides a range of dataset variants, including noisy datasets, configurations tailored for evaluating *stitching* efficiency, and *explore* datasets composed of suboptimal trajectories. Beyond navigation, OGBench encompasses manipulation tasks involving robotic arms engaged in diverse play-style activities, such as *puzzle*, *cube*, and *scene*. Although OGBench is primarily intended for GCRL, it also supports standard offline RL evaluation through *singletask* settings, which consist of five subtasks with fixed start-goal pairs and reward functions reconfigured for offline learning.

D.2 5 Learning Seeds vs. 24 Learning Seeds

As discussed in the main text, we aim to reduce the effect of the distractor policy phenomenon [11], where a model can achieve high peak scores despite low average performance due to large stochastic variance. To ensure a fair and robust evaluation, we expanded the number of random seeds beyond the conventional five, utilizing 24 seeds for D4RL and 12 seeds for OGBench. We tune hyperparameters using an initial set of 5 seeds, then train the model with 19 additional seeds and report the final

average over all 24 seeds. Table 4 shows the performance discrepancy between the initial 5-seed results and the full 24-seed results. As shown, the reported scores decrease on most tasks when the number of seeds is increased. Notably, on *walker2d-medium-replay*, the performance drops by more than 5%. These results suggest that the common practice of using only five random seeds may be insufficient for reliable evaluation, as it can fail to capture the inherent stochasticity of offline RL experiments.

D.3 Experimental Setup for Section 4.3

In Section 4.3, we empirically evaluate the extrapolation behavior of KVR-MLP against existing critic-regularization baselines. To ensure a fair comparison, we standardize all technical components, including the TD3 base architecture, network width, and ensemble size, and vary only the critic structure across methods. For each baseline, we use the critic-specific hyperparameters from its original implementation. Since the large batch size and BC penalty are core components of ReBRAC [41], we allow these hyperparameters to deviate from the standardized configuration used for the other methods.

D.4 Sensitivity to Bandwidth

Table 5: **Ablation study on the bandwidth hyperparameter (mean $\pm$ std, 5 seeds).** Green and orange bold text denote the best and second-best results for each task, respectively.

	bandwidth					
D4RL	1	2	4	8	16	32
halfcheetah-medium	20.9 $\pm$ 17.3	63.3 $\pm$ 1.7	**64.7** $\pm$ 1.2	**63.6** $\pm$ 0.8	62.8 $\pm$ 1.0	**63.6** $\pm$ 0.8
hopper-medium	101.9 $\pm$ 1.3	100.6 $\pm$ 6.0	**104.0** $\pm$ 0.4	103.4 $\pm$ 0.5	103.4 $\pm$ 1.8	**103.7** $\pm$ 1.0
walker2d-medium	88.4 $\pm$ 1.2	91.3 $\pm$ 2.0	92.6 $\pm$ 0.6	**96.5** $\pm$ 1.7	95.0 $\pm$ 3.4	**100.1** $\pm$ 1.4
antmaze-umaze-diverse	96.8 $\pm$ 0.7	96.6 $\pm$ 1.0	**97.6** $\pm$ 1.7	**97.4** $\pm$ 1.4	95.2 $\pm$ 2.5	96.6 $\pm$ 1.9
antmaze-medium-diverse	86.4 $\pm$ 5.3	92.2 $\pm$ 2.9	**93.6** $\pm$ 4.1	93.4 $\pm$ 4.8	**94.0** $\pm$ 2.4	81.8 $\pm$ 3.5
antmaze-large-diverse	87.4 $\pm$ 7.7	88.6 $\pm$ 3.9	**94.0** $\pm$ 3.5	89.4 $\pm$ 5.2	88.4 $\pm$ 6.6	**89.6** $\pm$ 2.8
Average	80.3	88.8	**91.1**	**90.6**	89.8	89.2

The RFF layer introduces several hyperparameters, including the number of sampled Fourier frequencies, their learning rate, the sampling measure, and the bandwidth. We fix the number of sampled Fourier frequencies at 1024 and use a uniform sampling measure across all tasks. For KVR-LN, the frequency learning rate is set to 0 for D4RL Gym-MuJoCo and to 3×10^{-5} for all other environments. Thus, bandwidth remains the main RFF-specific hyperparameter tuned across tasks, as it determines the initial frequency distribution even when the frequencies are learnable. We evaluate bandwidths from the set $S = \{1, 2, 4, 8, 16, 32\}$. For each task, we rank these choices by performance and then aggregate the scores at each rank across tasks. As shown in Figure 16 and Table 5, performance is relatively stable across a broad range of bandwidth settings, with most tasks exhibiting noticeable degradation only for a small subset of settings. Overall, these results indicate that KVR-LN is relatively insensitive to bandwidth over the evaluated range.

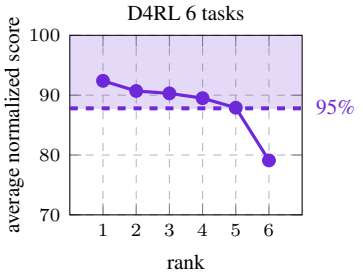


Figure 16: **Near-optimal performance can be achieved with small bandwidth tuning.** Scores are averaged via within-task ranking across MuJoCo medium and AntMaze diverse tasks.

D.5 Effect of Spectral Measures

In our didactic analysis, we examined how the probability measure used to sample Fourier frequencies affects estimation behavior, with each measure corresponding to a specific kernel in KRR. While the theoretical results suggest that the Laplacian kernel is the most robust in simplified settings [24], different results appear when RFFs are integrated with MLPs in complex offline RL tasks. Table 6

Table 6: **Experimental results across various spectral measures (mean ± std, 5 seeds).** Green and orange text denote the best and second-best results for each task, respectively.

Task	Gaussian	Laplacian	Orthogonal	Uniform
halfcheetah-medium	**64.9** ± 1.9	64.2 ± 1.5	**65.2** ± 0.3	64.7 ± 1.1
hopper-medium	**104.1** ± 0.4	**104.2** ± 0.5	103.5 ± 0.3	**104.1** ± 0.1
walker2d-medium	96.4 ± 1.6	94.4 ± 1.0	**98.0** ± 1.7	**96.6** ± 1.3
antmaze-umaze-diverse	**97.2** ± 0.7	96.0 ± 1.4	97.0 ± 1.7	**98.4** ± 2.4
antmaze-medium-diverse	**96.2** ± 1.5	90.4 ± 2.6	**95.0** ± 2.3	94.8 ± 3.1
antmaze-large-diverse	91.2 ± 2.2	90.4 ± 2.6	**91.4** ± 2.5	**92.4** ± 2.5
Average	91.6	89.9	**91.7**	**91.8**

reports D4RL benchmark scores for different spectral measures. Cauchy sampling, corresponding to the Laplacian kernel, yields the lowest average performance. We hypothesize that its heavy-tailed distribution produces large-magnitude frequency samples more frequently, which may make optimization more difficult. In contrast, Gaussian sampling, orthogonal sampling [47] (a method designed to enhance stability by enforcing orthogonality on Gaussian-based samples), and uniform sampling exhibit comparable performance. Although uniform sampling corresponds to the oscillatory Sinc kernel, it performs consistently well when combined with the neural critic and achieves the highest average score among the evaluated spectral measures. We therefore adopt uniform sampling as the default spectral measure in KVR.

D.6 Q-Values with Learnable Fourier Frequencies

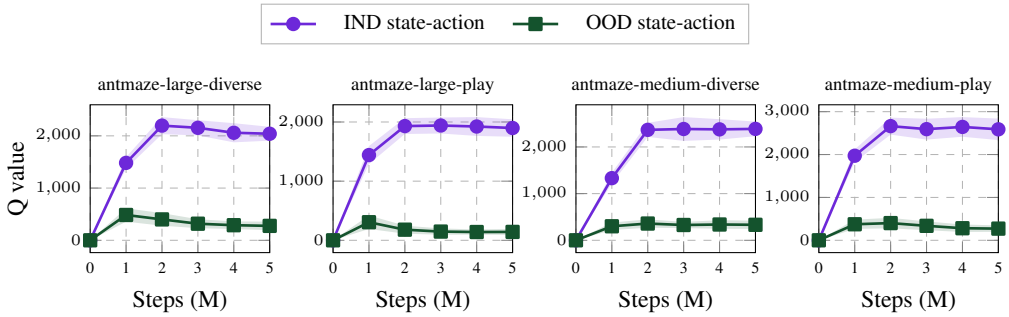


Figure 17: **IND and OOD Q-values with learnable Fourier frequencies across four D4RL AntMaze tasks.**

We examine the Q-value estimates of KVR with learnable Fourier frequencies on D4RL AntMaze medium and large tasks. Following the evaluation procedure in Section 4.3, we compare critic estimates for in-distribution state-action pairs and infeasible actions sampled outside the valid action space. As shown in Figure 17, Q-values for infeasible actions remain substantially lower than those for in-distribution pairs throughout training. These results support that KVR effectively suppresses Q-values for unsupported actions even when the Fourier frequencies are learned.

D.7 Adaptation of Learnable Fourier Frequencies

KVR treats the Fourier frequencies as learnable parameters to reduce sensitivity to their initial random sampling. To allow adaptation while limiting departures from the initial spectral samples, we set the frequency learning rate to at most $0.1\times$ the base learning rate. We first examine frequency adaptation on the D4RL AntMaze-diverse tasks. For Table 7, frequencies are initialized from the Gaussian spectral distribution corresponding to a Gaussian kernel with bandwidth 5. With a fixed dimension and bandwidth, Gaussian sampling produces frequency vectors with similar norms, making it easier to compare their magnitudes before and after training. We also report the mean absolute displacement per coordinate, $\Delta_{\text{freq}} = d^{-1}\|\boldsymbol{\xi}_{\text{learned}} - \boldsymbol{\xi}_{\text{init}}\|_1$, where d is the frequency vector dimension. The

Table 7: **Changes in learnable Fourier frequencies after training (mean $\pm$ std, 5 seeds).** We report frequency norms and the mean absolute displacement, $\Delta_{\text{freq}} = d^{-1}\|\boldsymbol{\xi}_{\text{learned}} - \boldsymbol{\xi}_{\text{init}}\|_1$, on D4RL AntMaze-diverse tasks. For this analysis, frequencies are sampled from the Gaussian spectral distribution corresponding to a Gaussian kernel with bandwidth 5.

D4RL-AntMaze-diverse	$\|\boldsymbol{\xi}_{\text{init}}\|$	$\|\boldsymbol{\xi}_{\text{learned}}\|$	Δ_{freq}
`umaze-diverse`	6.40 $\pm$ 0.0	6.43 $\pm$ 0.3	0.07 $\pm$ 0.0
`medium-diverse`	6.39 $\pm$ 0.0	6.59 $\pm$ 0.3	0.07 $\pm$ 0.0
`large-diverse`	6.39 $\pm$ 0.0	6.65 $\pm$ 0.1	0.06 $\pm$ 0.0
`Average`	6.40	6.56	0.07

Table 8: **Performance results under different Fourier frequency learning rates (mean $\pm$ std, 5 seeds).** Green and orange text denote the best and second-best results for each task, respectively.

Learning rate	0	3×10^{-5}	3×10^{-4}	3×10^{-3}
`umaze-diverse`	94.6 $\pm$ 4.4	**98.4** $\pm$ 2.4	**96.5** $\pm$ 1.1	0.0 $\pm$ 0.0
`medium-diverse`	**90.3** $\pm$ 4.1	**94.8** $\pm$ 3.1	0.0 $\pm$ 0.0	0.0 $\pm$ 0.0
`large-diverse`	**79.3** $\pm$ 6.4	**92.4** $\pm$ 2.5	0.0 $\pm$ 0.0	0.0 $\pm$ 0.0
`Average`	**88.1**	**95.2**	32.2	0.0

reported norms change by less than 5%, and Δ_{freq} remains below 0.1, indicating modest deviations from initialization.

We separately evaluate the frequency learning rate on the AntMaze-diverse tasks in Table 8. A learning rate of $0.1\times$ the base learning rate achieves the highest mean score on each task, outperforming both frozen frequencies and larger learning rates. Together, these results support using a small learning rate to enable frequency adaptation while limiting departures from initialization.

D.8 KVR with Different Base Algorithms

Table 9: **Performance of KVR with different base algorithms (mean $\pm$ std, 5 seeds).** Green and orange text denote the best and second-best results for each task, respectively.

D4RL	SAC+BC	IQL	TD3+BC (default)
`halfcheetah-medium`	**62.6** $\pm$ 0.4	46.7 $\pm$ 0.2	**64.7** $\pm$ 1.1
`hopper-medium`	**102.8** $\pm$ 1.7	64.2 $\pm$ 3.8	**104.1** $\pm$ 0.1
`walker2d-medium`	**93.4** $\pm$ 0.8	80.9 $\pm$ 2.5	**96.6** $\pm$ 1.3
`antmaze-umaze-diverse`	**98.0** $\pm$ 1.1	62.4 $\pm$ 10.3	**98.4** $\pm$ 2.4
`antmaze-medium-diverse`	**96.8** $\pm$ 0.8	52.4 $\pm$ 7.7	**94.8** $\pm$ 3.1
`antmaze-large-diverse`	**92.4** $\pm$ 3.6	30.0 $\pm$ 6.7	**92.4** $\pm$ 2.5
`Average`	**91.0**	56.1	**91.8**

In the main experiments, we instantiate KVR on top of TD3+BC. To examine whether its effectiveness depends on this particular base algorithm, we additionally integrate the RFF-KRR critic into SAC+BC and IQL and evaluate the resulting methods on D4RL MuJoCo-medium and AntMaze tasks. As shown in Table 9, KVR with SAC+BC achieves performance comparable to its TD3+BC counterpart, indicating that the RFF-based critic is not specific to TD3+BC. In contrast, the IQL-based variant yields lower performance and remains closer to the original IQL. This difference may arise from the value-learning structure of IQL. Unlike TD3+BC and SAC+BC, IQL learns the state value through expectile regression over dataset actions and uses this value for Bellman backups, without evaluating next actions sampled by the learning policy. Consequently, shaping the extrapolation of Q-values for unsupported actions has a more limited effect on IQL's value updates, potentially reducing the benefit of the RFF-based critic.

Table 10: **OGBench results under matched settings (mean ± std, 12 seeds).** All results use matched experimental settings. KVR is trained for 1M steps without critic ensembles. Green and orange boldface indicate the best and second-best scores for each task, respectively.

OGBench Single-Task	IQL	ReBRAC	IFQL	FQL	GFP	KVR 1M Steps w/o Ens.
antmaze-large-navigate						
task1	54.1 ± 8.1	**93.8** ± 12.4	46.4 ± 14.3	91.4 ± 1.8	89.6 ± 2.2	**94.5** ± 1.8
task2	**49.9** ± 18.7	34.5 ± 43.7	32.6 ± 13.3	45.6 ± 41.9	34.4 ± 31.4	**87.3** ± 4.0
task3	74.8 ± 28.8	**92.8** ± 6.8	33.0 ± 43.7	84.3 ± 17.7	90.3 ± 2.3	**93.0** ± 2.6
task4	**63.3** ± 7.0	46.6 ± 39.8	10.7 ± 9.1	6.6 ± 8.7	19.1 ± 19.1	**90.6** ± 3.5
task5	79.6 ± 3.2	**96.0** ± 2.1	56.5 ± 23.2	69.6 ± 40.2	78.0 ± 29.5	**93.0** ± 1.6
antmaze-giant-navigate						
task1	0.1 ± 0.4	**9.4** ± 21.1	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	**59.5** ± 14.5
task2	1.2 ± 2.6	**14.9** ± 20.0	0.9 ± 2.3	0.1 ± 0.3	0.0 ± 0.0	**67.0** ± 23.7
task3	0.1 ± 0.4	0.0 ± 0.0	**2.9** ± 2.9	0.0 ± 0.0	0.0 ± 0.0	**72.0** ± 6.8
task4	0.6 ± 1.1	**0.9** ± 1.6	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	**82.6** ± 14.1
task5	33.1 ± 8.4	**53.5** ± 19.5	8.2 ± 6.8	2.5 ± 1.8	0.0 ± 0.0	**68.3** ± 10.7
humanoidmaze-medium-navigate						
task1	27.7 ± 7.8	18.5 ± 8.4	**81.6** ± 3.6	1.0 ± 0.7	**82.3** ± 17.0	33.9 ± 15.3
task2	34.2 ± 5.1	58.1 ± 22.6	**91.8** ± 3.0	7.1 ± 3.1	**97.5** ± 2.1	77.5 ± 6.8
task3	27.7 ± 8.7	43.7 ± 7.0	36.3 ± 42.1	8.5 ± 2.5	**71.1** ± 35.2	**55.1** ± 13.2
task4	0.4 ± 0.5	**25.6** ± 17.1	1.4 ± 1.1	2.4 ± 1.5	0.0 ± 0.0	**60.8** ± 7.6
task5	64.0 ± 5.5	74.1 ± 15.2	**97.9** ± 1.0	6.6 ± 3.2	**98.5** ± 1.5	68.8 ± 14.5
humanoidmaze-large-navigate						
task1	1.6 ± 1.0	3.4 ± 3.0	1.8 ± 0.7	0.0 ± 0.0	**17.8** ± 28.6	**22.0** ± 23.9
task2	0.0 ± 0.0	0.1 ± 0.4	**0.4** ± 0.7	0.0 ± 0.0	0.0 ± 0.0	**1.4** ± 2.3
task3	6.6 ± 3.0	**38.5** ± 11.6	**58.4** ± 10.7	0.6 ± 0.7	11.5 ± 3.9	38.4 ± 21.0
task4	1.5 ± 1.0	**7.5** ± 4.7	0.3 ± 0.7	0.9 ± 0.6	0.0 ± 0.0	**10.0** ± 12.9
task5	1.8 ± 2.1	**5.3** ± 4.1	0.0 ± 0.0	0.1 ± 0.4	**11.4** ± 19.9	0.3 ± 0.7
scene-play						
task1	40.5 ± 41.2	84.5 ± 7.0	97.2 ± 1.4	**99.9** ± 0.4	**97.6** ± 6.3	88.9 ± 13.8
task2	10.7 ± 12.6	27.2 ± 6.2	**90.3** ± 4.5	**94.9** ± 4.9	85.0 ± 14.1	80.4 ± 10.9
task3	13.0 ± 13.8	30.4 ± 20.5	35.5 ± 10.9	**94.3** ± 2.1	**89.4** ± 6.1	76.8 ± 10.6
task4	9.9 ± 8.4	58.9 ± 22.4	**75.4** ± 9.2	**62.2** ± 28.7	22.4 ± 30.5	48.4 ± 25.2
task5	14.9 ± 15.8	11.4 ± 9.6	**19.9** ± 11.6	**46.7** ± 26.3	12.4 ± 29.4	16.6 ± 11.3
puzzle-3x3-play						
task1	7.4 ± 5.1	52.8 ± 15.2	96.5 ± 2.8	**99.9** ± 0.4	97.1 ± 3.6	**98.1** ± 1.5
task2	1.8 ± 1.4	16.5 ± 29.1	**95.5** ± 4.0	74.1 ± 33.0	94.9 ± 5.0	**98.1** ± 2.2
task3	1.3 ± 0.8	25.5 ± 36.8	**95.3** ± 3.2	84.8 ± 23.0	**89.6** ± 12.4	**95.3** ± 3.6
task4	1.0 ± 0.7	6.6 ± 6.1	**92.1** ± 3.2	64.8 ± 23.9	90.9 ± 9.7	**93.5** ± 5.0
task5	1.6 ± 1.7	10.9 ± 23.2	87.0 ± 23.9	80.6 ± 31.1	**87.9** ± 22.1	**96.8** ± 2.2
Average	20.8	34.7	44.9	37.7	**49.0**	**65.6**

D.9 Comparison with Baselines under Matched Experimental Settings on OGBench

To provide a more consistent comparison on OGBench, we evaluate KVR and the baselines under aligned experimental settings. Because each method relies on different method-specific components and hyperparameters, making every aspect of the experimental setup identical is neither feasible nor desirable. Instead, we align the key factors associated with KVR's implementation. Specifically, we use the following settings:

1. **Reward setting:** A reward scale of 10000 is applied to all OGBench tasks.
2. **Training budget:** All methods are trained for 1M gradient steps.
3. **Critic ensemble:** No additional critic ensemble is used beyond standard double Q-learning.
4. **BC regularization:** BC regularization is applied to all applicable baselines except IQL.

5. **Network architecture:** Both the actor and critic use four hidden layers of width 512.
6. **Implementation:** All methods are implemented and evaluated in the same codebase based on the FQL implementation.

These settings reduce differences in auxiliary components of KVR, allowing us to evaluate the contribution of the RFF-based critic more directly.

Since the reported baselines were tuned for the standard OGBench setting, their original hyperparameters may be suboptimal under our matched setting. We therefore retune each baseline following the guidelines from its original implementation. For each method, we select the hyperparameters most influential to its performance and search over the following candidate values:

IQL
- Inverse temperature α: $\{0.3, 1, 3, 10\}$

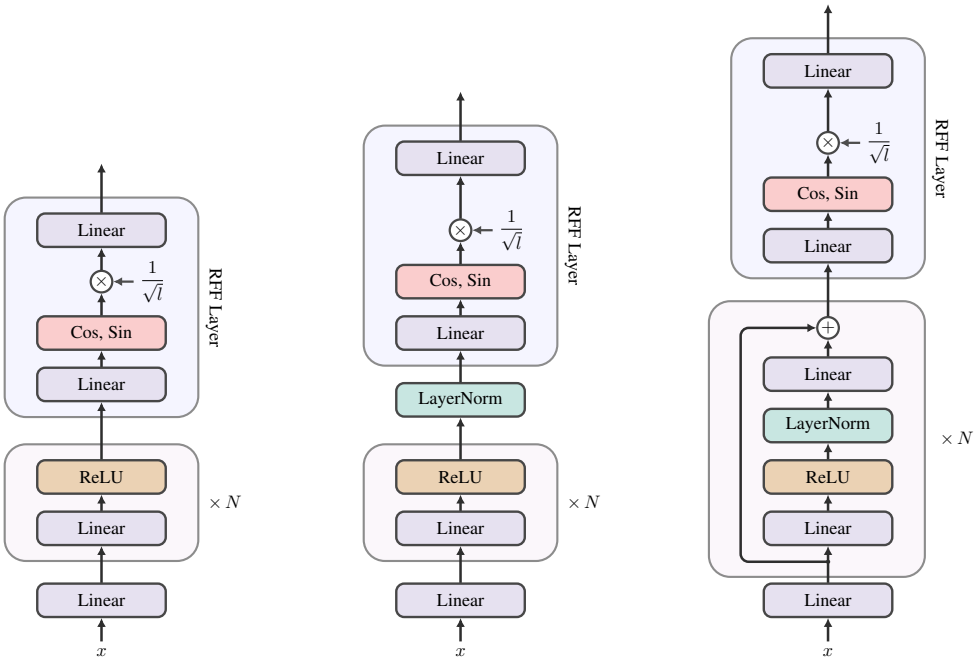
ReBRAC
- Actor BC coefficient β_1: $\{0.001, 0.01, 0.1, 1\}$
- Critic BC coefficient β_2: $\{0.001, 0.01, 0.1, 1\}$

IFQL
- Number of candidate actions N: $\{32, 64, 128\}$

FQL
- Actor distillation coefficient α: $\{10, 30, 300, 1000, 10000\}$

GFP
- Actor distillation coefficient α: $\{0.01, 0.03, 0.1, 0.3, 1, 3\}$
- Guidance temperature η: $\{10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}\}$

As shown in Table 10, the matched settings affect the task domains differently: the retuned baselines generally improve on the manipulation tasks but perform worse on the maze-navigation tasks. Nevertheless, KVR achieves the highest average performance under these settings, even without an additional critic ensemble.

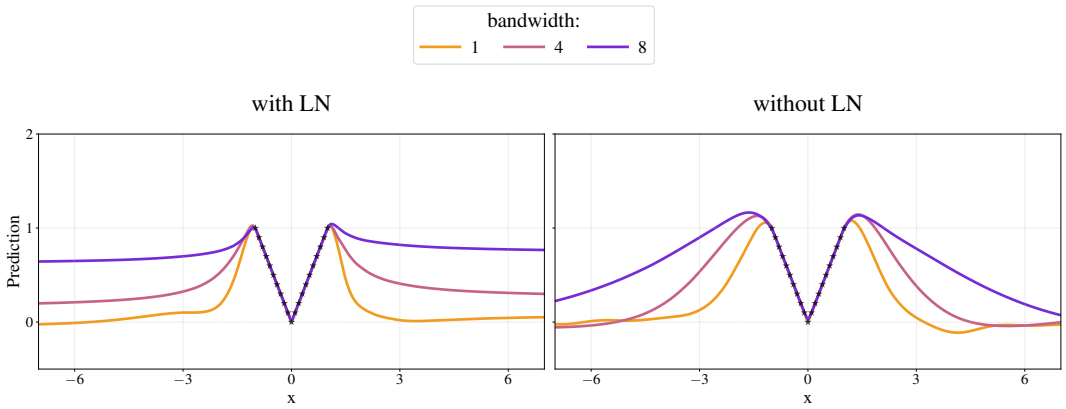


(a) KVR-MLP (b) KVR-LN (c) KVR-RES

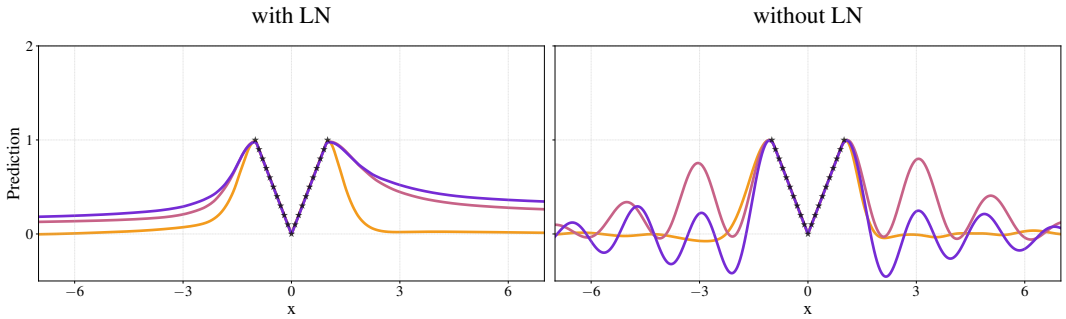
Figure 18: **Critic architectures of KVR.**

D.10 Critic Architectures for KVR

As discussed in the main text, KVR considers three critic architectures: a vanilla ReLU-based MLP (KVR-MLP), its variant with LN applied immediately before the RFF layer (KVR-LN), and a residual network with LN (KVR-RES) (Figure 18). In this section, we investigate the properties of these architectures.



(a) Fixed Fourier frequencies



(b) Learnable Fourier frequencies

Figure 19: **Comparison of extrapolation with and without RFF input normalization in the 1D didactic example.** With learnable frequencies, LN suppresses extrapolation oscillations and yields smooth saturation.

D.10.1 Effect of Normalization Before the RFF Layer

Corollary 1 establishes that preserving the vanishing property in the input space requires distant inputs to remain far from the training representations in the latent space. Under suitable conditions, ReLU-based MLPs satisfy this requirement through their asymptotically affine behavior, while residual networks do so through bounds on latent-space distances. However, applying LN before the RFF layer can disrupt this distance relationship by normalizing latent representations, thereby removing the theoretical guarantee of the vanishing property in the input space. Nevertheless, LN can still improve extrapolation stability by inducing asymptotically flat predictions and reducing the sensitivity of kernel correlations to latent feature scales, while retaining bandwidth-dependent extrapolation behavior.

Asymptotic saturation under LN. We first analyze how LN affects the extrapolation behavior of the RFF-based critic. Consider a fixed input direction $\mathbf{x}(t) = \mathbf{x}_0 + t\mathbf{v}$, where $\mathbf{v} \neq 0$ and $t \to \infty$. Since ReLU-based MLPs are piecewise affine, their representations eventually take the form

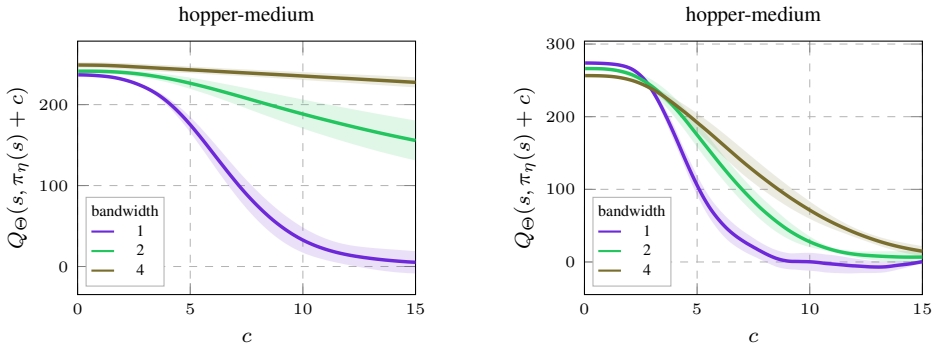
$$g_\theta(\mathbf{x}(t)) = t\mathbf{a} + \mathbf{b}$$

for some fixed vectors $\mathbf{a}$ and $\mathbf{b}$. Applying LN to this representation yields a finite limit as $t \to \infty$, leading to the following result.

Proposition 3 (Asymptotic saturation under LN)**.** *Let $g_\theta : \mathbb{R}^d \to \mathbb{R}^h$ be a fixed finite ReLU-based MLP, and define*

$$f(\mathbf{x}) = \mathbf{w}^\top \phi(\mathrm{LN}(g_\theta(\mathbf{x}))),$$

where $\mathbf{w}$ and the Fourier frequencies are fixed and finite, and LN has fixed finite affine parameters with $\epsilon > 0$.



(a) KVR-LN (b) KVR-MLP

Figure 20: **Effect of LN on KVR's extrapolation behavior across bandwidths (mean ± std).**

Then, along any input ray $\mathbf{x}(t) = \mathbf{x}_0 + t\mathbf{v}$ *with* $\mathbf{v} \neq 0$*, there exists a finite constant* f_∞ *such that*

$$f(\mathbf{x}(t)) = f_\infty + O(t^{-1}), \qquad \frac{d}{dt} f(\mathbf{x}(t)) = O(t^{-2})$$

as $t \to \infty$.

Proof. Since g_θ is a finite ReLU-based MLP, it is eventually affine along any fixed input ray:

$$g_\theta(\mathbf{x}(t)) = t\mathbf{a} + \mathbf{b}.$$

Let $P = I - \frac{1}{h}\mathbf{1}\mathbf{1}^\top$ be the centering matrix. Then

$$\mathrm{LN}(t\mathbf{a} + \mathbf{b}) = \Gamma \frac{\sqrt{h}(tP\mathbf{a} + P\mathbf{b})}{\sqrt{\|tP\mathbf{a} + P\mathbf{b}\|_2^2 + h\epsilon}} + \boldsymbol{\beta},$$

where Γ and $\boldsymbol{\beta}$ are the affine parameters of LN.

If $P\mathbf{a} \neq 0$, the normalized representation converges to a finite limit $\mathbf{z}_\infty$, satisfying

$$\mathrm{LN}(t\mathbf{a} + \mathbf{b}) = \mathbf{z}_\infty + O(t^{-1}), \qquad \frac{d}{dt}\mathrm{LN}(t\mathbf{a} + \mathbf{b}) = O(t^{-2}).$$

If $P\mathbf{a} = 0$, the normalized representation is eventually constant, so the same bounds hold. Since the RFF mapping is smooth with bounded derivatives for fixed finite frequencies, the result follows from the mean value inequality and chain rule. □

Proposition 3 shows that applying LN before the RFF layer induces asymptotically flat extrapolation along distant input directions. Unlike the vanishing property, which drives predictions toward zero, this behavior allows predictions to converge to finite, potentially nonzero values. Nevertheless, the diminishing rate of change provides a mechanism for suppressing persistent oscillations in extrapolated predictions.

Effect on kernel geometry and bandwidth. LN also alters kernel geometry by reducing sensitivity to the scale of latent representations. In our implementation, the uniform spectral distribution corresponds to the product Sinc kernel

$$k_\beta(\mathbf{z}, \mathbf{z}') = \prod_{j=1}^{h} \mathrm{sinc}\left(\frac{z_j - z'_j}{\beta}\right).$$

Without normalization, scaling latent representations by $\alpha > 0$ is equivalent to rescaling the kernel bandwidth:

$$k_\beta(\alpha\mathbf{z}, \alpha\mathbf{z}') = k_{\beta/\alpha}(\mathbf{z}, \mathbf{z}').$$

In contrast, LN is approximately invariant to positive rescaling when the feature variance dominates ϵ, reducing the dependence of kernel correlations on latent feature scales while retaining bandwidth-dependent similarity. Furthermore, Proposition 3 implies that kernel similarities along distant input directions converge to finite, potentially nonzero limits that depend on the bandwidth. This is consistent with the bandwidth-dependent extrapolation behavior observed in KVR despite the additional normalization.

Table 11: **Benchmark results across critic architectures.** We compare KVR-LN, KVR-MLP, and KVR-RES. Results are reported as mean ± standard deviation over 5 and 12 learning seeds for D4RL and OGBench, respectively. Green and orange bold text indicate the best and second-best scores for each task, respectively.

	KVR-LN	KVR-MLP	KVR-RES
D4RL-MuJoCo			
halfcheetah-random	**30.3** ± 0.9	29.5 ± 1.0	**29.6** ± 1.2
hopper-random	**32.3** ± 0.2	**32.2** ± 0.1	31.7 ± 0.3
walker2d-random	**21.9** ± 0.9	**21.9** ± 0.5	21.8 ± 0.3
halfcheetah-medium	**64.7** ± 1.1	63.7 ± 1.1	**66.7** ± 0.4
hopper-medium	104.1 ± 0.1	**104.3** ± 0.4	**104.9** ± 0.4
walker2d-medium	**96.6** ± 1.3	**93.6** ± 0.2	93.2 ± 0.5
halfcheetah-medium-replay	55.2 ± 0.7	**55.3** ± 0.4	**56.7** ± 0.3
hopper-medium-replay	**102.5** ± 1.5	101.7 ± 0.4	**102.4** ± 0.1
walker2d-medium-replay	**102.0** ± 2.1	100.7 ± 1.4	**100.8** ± 3.1
halfcheetah-medium-expert	**106.2** ± 0.8	**103.1** ± 2.8	102.9 ± 2.2
hopper-medium-expert	**113.0** ± 0.1	112.4 ± 0.1	**112.6** ± 0.4
walker2d-medium-expert	**113.1** ± 0.9	111.7 ± 0.3	**113.1** ± 0.3
D4RL-AntMaze			
umaze	**99.0** ± 0.7	**98.4** ± 0.5	**98.4** ± 0.7
umaze-diverse	**98.4** ± 2.4	**97.0** ± 3.0	96.0 ± 1.7
medium-play	94.0 ± 1.7	**96.4** ± 1.6	**94.8** ± 0.4
medium-diverse	**94.8** ± 3.1	**90.0** ± 7.1	89.4 ± 2.7
large-play	**93.8** ± 2.2	90.5 ± 2.5	**93.6** ± 1.2
large-diverse	**92.4** ± 2.5	**90.8** ± 3.7	89.4 ± 4.9
OGBench-HumanoidMaze			
medium-navigate-task1	75.4 ± 5.9	**86.0** ± 11.8	1.5 ± 2.3
medium-navigate-task2	86.3 ± 12.0	**97.8** ± 1.5	0.1 ± 0.3
medium-navigate-task3	73.8 ± 11.5	**89.6** ± 4.1	7.3 ± 5.0
medium-navigate-task4	39.1 ± 12.3	**42.3** ± 31.8	10.1 ± 10.1
medium-navigate-task5	**75.5** ± 26.8	34.8 ± 44.4	8.5 ± 12.5
large-navigate-task1	13.5 ± 3.8	**82.1** ± 15.2	0.0 ± 0.0
large-navigate-task2	2.0 ± 2.7	**27.5** ± 35.6	0.0 ± 0.0
large-navigate-task3	25.2 ± 2.4	**93.9** ± 2.2	0.6 ± 1.0
large-navigate-task4	15.0 ± 15.6	**65.0** ± 37.8	3.0 ± 1.7
large-navigate-task5	11.2 ± 8.9	**51.9** ± 40.3	0.5 ± 1.0
Average	**69.0**	**77.3**	54.6

Validation experiments. To empirically validate the preceding analysis, we investigate the effect of normalizing RFF inputs on the didactic regression example and the D4RL hopper-medium task.

For the didactic example, we use a uniform spectral distribution and examine extrapolation across different bandwidths, considering both fixed and learnable Fourier frequencies. As shown in Figure 19, with fixed frequencies, the model with LN exhibits asymptotically flat extrapolation rather than vanishing toward zero, with smaller bandwidths yielding lower saturation levels. In contrast, the model without LN exhibits predictions that decay toward zero outside the data support, while retaining bandwidth-dependent extrapolation. When the Fourier frequencies are learned, the model with LN saturates at lower values than in the fixed-frequency setting. Notably, the model without LN exhibits unstable oscillations outside the data support, whereas LN produces smooth and stable saturation. These observations are consistent with the theoretical analysis of LN-induced asymptotic saturation and its stabilizing effect on extrapolation.

We further investigate whether these properties persist in complex offline RL by comparing KVR-LN and KVR-MLP across different bandwidths on the D4RL hopper-medium task. Following the experimental setup in Section 4.3, we sample states $\mathbf{s}$ from the dataset and evaluate critic predictions for actions obtained by perturbing the learned policy output $\pi_\eta(\mathbf{s})$ with $c \cdot \mathbb{1}$, where c controls the magnitude of the perturbation. As shown in Figure 20, KVR-MLP exhibits consistent bandwidth-

dependent extrapolation, with Q-values approaching zero as the perturbation increases. In contrast, KVR-LN maintains nonzero Q-values in distant regions, exhibiting saturation rather than vanishing behavior. Nevertheless, smaller bandwidths continue to yield lower extrapolated Q-values, indicating that bandwidth-dependent regulation persists despite normalization.

Together, these results support the theoretical analysis, showing that LN induces asymptotically flat extrapolation while improving the stability of predictions in unsupported regions.

D.10.2 Benchmark Results

Table 11 compares the benchmark performance of different critic architectures. On D4RL MuJoCo, KVR-LN achieves the highest average score, closely followed by KVR-RES and KVR-MLP. KVR-LN also achieves the highest average performance on D4RL AntMaze. On OGBench, KVR-RES performs substantially worse than the other two variants, whereas KVR-MLP achieves the highest average score, with particularly large gains on humanoidmaze-large.

D.11 Comparison with PARS

Table 12: **Comparison between KVR and PARS on D4RL AntMaze under an identical single hyperparameter setting.** Results for KVR are averaged over 24 training seeds.

D4RL-AntMaze	PARS	KVR
`umaze`	93.3	**99.1**
`umaze-diverse`	84.7	**95.2**
`medium-play`	89.4	**93.3**
`medium-diverse`	86.2	**93.8**
`large-play`	77.5	**91.0**
`large-diverse`	83.5	**91.8**
`Average`	85.8	**94.0**

KVR and PARS [13] share TD3+BC as the base algorithm and use several common components, including critic ensembles and reward scaling. We adopt most general hyperparameters from the original PARS implementation. For task-specific configurations on D4RL, including Gym-MuJoCo and AntMaze, we largely use the settings reported in PARS. We tune hyperparameters for tasks where the default settings do not yield stable convergence. Notably, for AntMaze, we use a unified hyperparameter configuration across all tasks, strictly following the single-hyperparameter setup established in PARS. Table 12 compares KVR and PARS under the same setup: critic ensemble size 4, reward scale 10000, and BC coefficient 0.01. Even with these synchronized parameters, KVR significantly outperforms PARS across all AntMaze tasks. This performance gap highlights the effectiveness of the RFF layer compared with the penalizing infeasible actions component.

D.12 Ablation on ridge regularization

Table 13: **Ablation study on ridge regularization (mean $\pm$ std, 5 seeds).** Green and orange text denote the best and second-best results for each task, respectively.

	ridge λ				
D4RL	`1`	`0.01`	`0.0001`	`0.000001`	`0`
`halfcheetah-medium`	-1.1 $\pm$ 2.1	0.2 $\pm$ 2.6	46.1 $\pm$ 0.9	**59.7** $\pm$ 1.1	**64.7** $\pm$ 1.1
`hopper-medium`	0.9 $\pm$ 0.7	12.8 $\pm$ 10.8	17.9 $\pm$ 25.2	**102.5** $\pm$ 0.4	**104.1** $\pm$ 0.1
`walker2d-medium`	0.3 $\pm$ 1.5	0.3 $\pm$ 1.3	24.9 $\pm$ 14.8	**87.9** $\pm$ 19.6	**96.6** $\pm$ 1.3
`antmaze-umaze-diverse`	0.0 $\pm$ 0.0	6.4 $\pm$ 6.9	95.0 $\pm$ 2.6	**95.8** $\pm$ 2.3	**98.4** $\pm$ 2.4
`antmaze-medium-diverse`	34.8 $\pm$ 6.8	57.8 $\pm$ 24.8	**94.4** $\pm$ 1.7	92.7 $\pm$ 4.3	**94.8** $\pm$ 3.1
`antmaze-large-diverse`	24.4 $\pm$ 12.1	85.6 $\pm$ 2.9	89.2 $\pm$ 3.5	**90.0** $\pm$ 5.4	**92.4** $\pm$ 2.5
`Average`	9.9	27.2	61.3	**88.1**	**91.8**

Table 13 presents the performance results for different ridge regularization values. For most tasks, performance improves as the ridge parameter decreases, with the ridgeless setting achieving the best performance on all evaluated tasks.

D.13 Computational Cost

We compare KVR and the baselines in terms of per-step training time and GPU memory usage. All methods are implemented in JAX [3] and evaluated on the D4RL hopper-medium task using a single NVIDIA GeForce RTX 3090 GPU. We report the average per-step training time over 1,000 gradient steps, with the results summarized in Table 14. Because all methods share the same TD3 base algorithm, the differences in speed and memory usage are moderate. In terms of GPU memory usage, FANS requires the largest memory due to its relatively large network architecture. For per-step training time, ReBRAC and FANS are slower than the other methods, as they employ deeper networks. KVR is slower than PARS due to the additional computation introduced by the RFF layer.

Table 14: **Computational cost comparison on hopper-medium task.**

Method	Time / step (ms)	GPU memory (MB)
KVR	1.973	532.0
ReBRAC	2.195	412.0
FANS	2.124	604.0
PARS	1.750	420.0

D.14 Offline-to-Online Fine-Tuning.

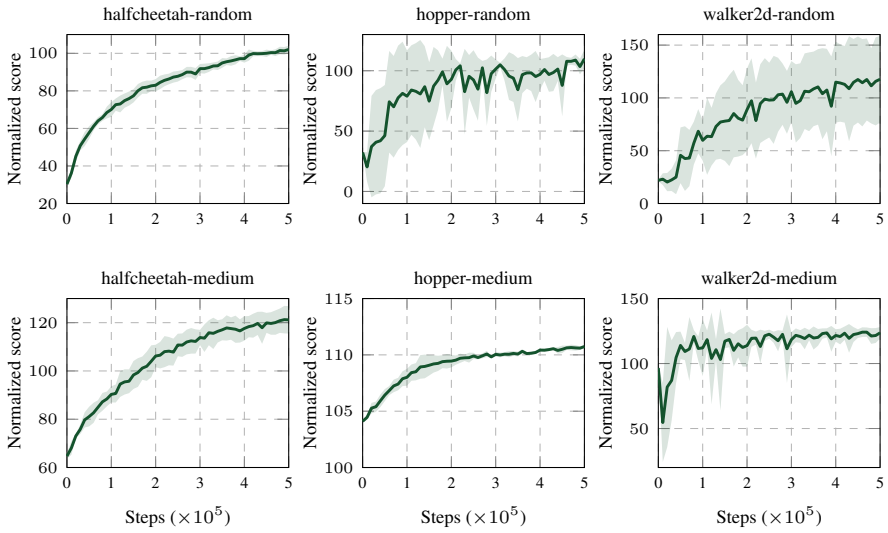


Figure 21: **Learning curves of online fine-tuning across D4RL random and medium tasks (mean $\pm$ std, 5 seeds).**

Offline-to-online (O2O) RL aims to further improve an offline pre-trained policy through additional interaction with the environment. Although KVR is primarily designed for the offline setting, we additionally examine whether it can serve as an effective initialization for online fine-tuning. We evaluate KVR on the random and medium datasets of D4RL and continue training for 500000 gradient steps. The resulting learning curves are shown in Figure 21. Despite the concern that KVR's tendency to suppress values for unsupported actions could hinder exploration beyond the offline data support, KVR can be successfully fine-tuned across the evaluated tasks. One possible explanation is related to how KVR treats Q-values on the offline data support. As discussed by Nakamoto et al. [28], excessive underestimation of Q-values on the offline data support can impede early-stage online fine-tuning. KVR attenuates extrapolated values without explicitly penalizing Q-values on the data support, thereby avoiding direct suppression of Bellman targets on observed state-action pairs. As new transitions are collected from previously unseen regions, those regions become part of the observed data and are no longer governed by KVR's extrapolation bias. The critic can then fit these newly

observed samples directly, without suppressing their values, allowing the policy to gradually explore beyond the original offline support.

D.15 Full Results

D.15.1 Full Benchmark Results.

Tables 15 and 16 report the full results on D4RL and OGBench, respectively.

Table 15: **The full results on D4RL tasks (mean ± std, 24 seeds).** Green and orange bold text indicate the best and second-best scores for each task, respectively.

GYM-MuJoCo	IQL	EDAC	TD3BC	ReBRAC	PARS	FANS	KVR-LN	KVR-MLP
halfcheetah-random	13.1	28.4	11.0	**29.5**	**30.4**	_	29.3 ± 1.9	29.1 ± 2.0
hopper-random	7.9	25.3	8.5	8.1	25.4	_	**32.3** ± 0.3	**32.3** ± 0.1
walker2d-random	5.4	16.6	1.6	18.4	**21.8**	_	21.4 ± 1.6	**22.1** ± 0.7
halfcheetah-medium	47.4	**65.9**	48.3	65.6	64.2	**66.6**	64.8 ± 1.3	63.2 ± 1.6
hopper-medium	66.3	101.6	59.3	102.0	104.1	**104.6**	**104.2** ± 0.4	104.1 ± 0.3
walker2d-medium	78.3	92.5	83.7	82.5	**97.3**	**101.0**	96.4 ± 1.4	92.5 ± 0.7
halfcheetah-medium-replay	44.2	**61.3**	44.6	51.0	**57.0**	55.9	55.1 ± 0.9	55.1 ± 0.7
hopper-medium-replay	94.7	101.0	60.9	98.1	**103.1**	**103.2**	103.0 ± 1.2	101.1 ± 1.1
walker2d-medium-replay	73.9	87.1	81.8	77.3	95.8	**98.3**	95.5 ± 10.7	**97.8** ± 3.7
halfcheetah-medium-expert	86.7	**106.3**	90.7	101.1	103.0	102.8	**105.4** ± 3.4	104.1 ± 2.5
hopper-medium-expert	91.5	110.7	98.0	107.0	**113.1**	**113.3**	112.9 ± 0.3	112.5 ± 0.2
walker2d-medium-expert	109.6	**114.7**	110.1	111.6	111.8	**118.1**	110.5 ± 5.8	111.4 ± 0.4
Average	59.9	76.0	58.2	71.0	**77.3**	_	**77.6**	77.1

AntMaze	IQL	EDAC	TD3BC	ReBRAC	PARS	FANS	KVR-LN	KVR-MLP
umaze	87.5	0.0	78.6	97.8	93.8	87.8	**99.1** ± 1.1	**98.3** ± 1.7
umaze-diverse	62.2	0.0	71.4	88.3	89.9	72.3	**95.2** ± 2.4	**97.5** ± 2.3
medium-play	71.2	0.0	10.6	84.0	91.2	3.3	**93.3** ± 2.9	**95.2** ± 2.7
medium-diverse	70.0	0.0	3.0	76.3	**92.0**	5.8	**93.8** ± 2.7	89.0 ± 6.2
large-play	39.6	0.0	0.2	60.4	84.8	0.0	**91.0** ± 3.0	**89.8** ± 4.3
large-diverse	47.5	0.0	0.0	54.4	83.2	0.0	**91.8** ± 3.1	**89.4** ± 4.1
Average	63.0	0.0	27.3	76.9	89.2	28.2	**94.0**	**93.2**

Table 16: **The full results on OGBench tasks (mean ± std, 12 seeds).** Green and orange bold text indicate the best and second-best scores for each task, respectively.

OGBench Singletask	IQL	ReBRAC	IFQL	FQL	GFP	KVR-LN	KVR-MLP
antmaze-large-navigate							
task1	48	91	24	80	95.4	**97.8** ± 1.0	**97.6** ± 1.9
task2	42	88	8	57	92.2	**92.6** ± 3.1	**92.9** ± 2.1
task3	72	51	52	93	95.6	**99.4** ± 0.7	**97.8** ± 1.8
task4	51	84	18	80	**90.6**	**95.6** ± 2.0	77.7 ± 30.5
task5	54	90	38	83	95.0	**97.2** ± 1.3	**98.0** ± 1.3
antmaze-giant-navigate							
task1	0	27	0	4	12.6	**69.0** ± 7.8	**75.9** ± 4.4
task2	1	16	0	9	52.2	**92.5** ± 3.3	**57.3** ± 31.4
task3	0	**34**	0	0	13.7	**87.5** ± 2.8	16.9 ± 29.4
task4	0	5	0	14	17.8	**91.4** ± 3.2	**87.6** ± 5.8
task5	19	49	13	16	43.2	**90.9** ± 2.8	**74.4** ± 32.3
humanoidmaze-medium-navigate							
task1	32	16	69	19	**83.5**	75.4 ± 5.9	**86.0** ± 11.8
task2	41	18	85	**94**	91.2	86.3 ± 12.0	**97.8** ± 1.5
task3	25	36	49	74	**86.3**	73.8 ± 11.5	**89.6** ± 4.1
task4	0	15	1	3	3.0	**39.1** ± 12.3	**42.3** ± 31.8
task5	66	24	**98**	**97**	95.8	75.5 ± 26.8	34.8 ± 44.4
humanoidmaze-large-navigate							
task1	3	2	6	7	**57.2**	13.5 ± 3.8	**82.1** ± 15.2
task2	0	0	0	0	0.1	**2.0** ± 2.7	**27.5** ± 35.6
task3	7	8	**48**	11	14.6	25.2 ± 2.4	**93.9** ± 2.2
task4	1	1	1	2	3.7	**15.0** ± 15.6	**65.0** ± 37.8
task5	1	2	0	1	**13.1**	11.2 ± 8.9	**51.9** ± 40.3
scene-play							
task1	94	95	98	**100**	**99.8**	93.9 ± 3.9	92.6 ± 5.6
task2	12	50	0	76	**89.0**	75.6 ± 8.0	**87.1** ± 6.2
task3	32	55	54	**98**	**78.0**	62.3 ± 7.7	73.4 ± 7.7
task4	0	3	0	5	0.6	**70.2** ± 17.7	**86.5** ± 10.8
task5	0	0	0	0	0.0	**75.8** ± 7.4	**66.1** ± 11.3
puzzle-3x3-play							
task1	33	97	94	90	94.8	**99.0** ± 1.5	**99.1** ± 0.7
task2	4	1	1	16	0.3	**98.5** ± 2.2	**98.9** ± 1.0
task3	3	3	0	10	0.9	**93.6** ± 3.7	**96.3** ± 2.2
task4	2	2	0	16	5.4	**90.0** ± 5.3	**96.4** ± 1.5
task5	3	5	0	16	14.1	**96.0** ± 3.0	**99.1** ± 0.4
Average	21.5	32.3	25.2	39.0	48.0	**72.9**	**78.1**

D.15.2 Learning Curves of KVR.

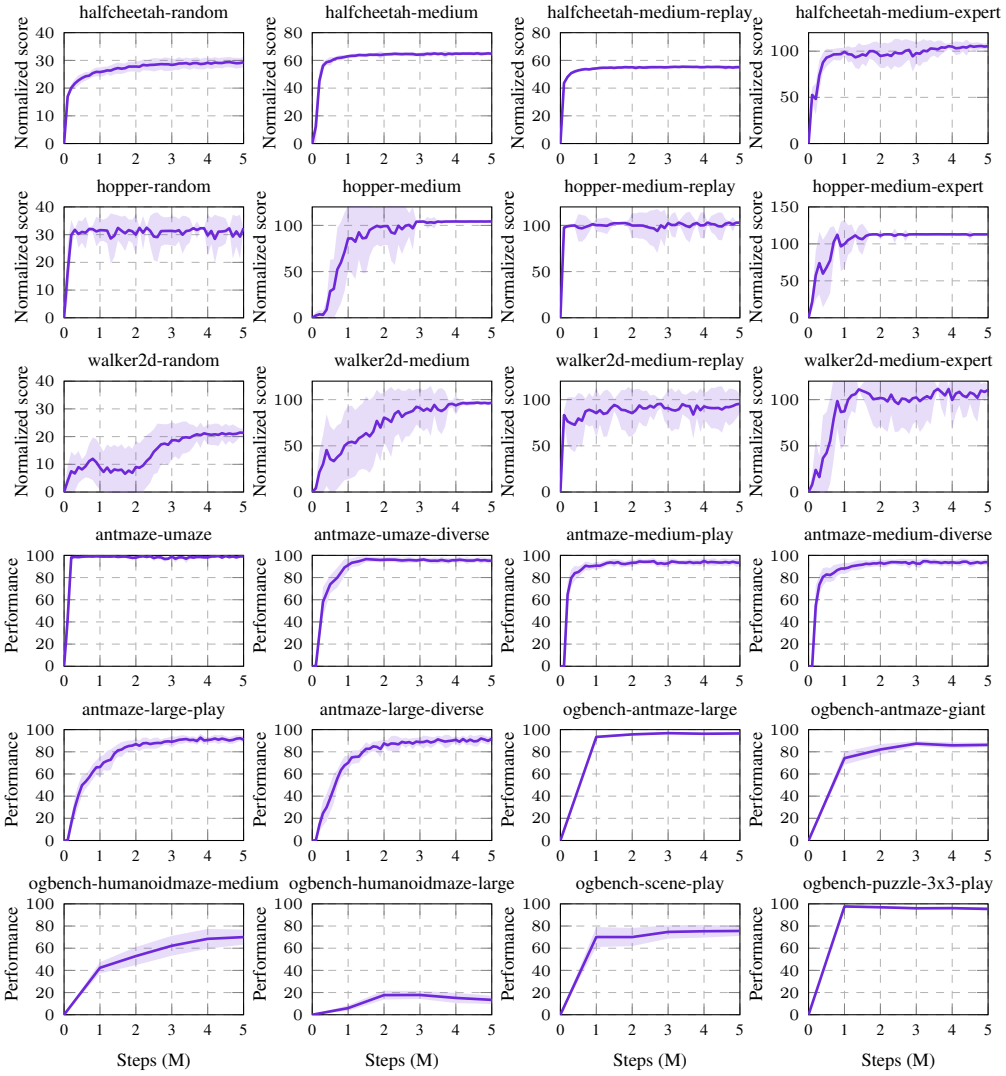


Figure 22: **Learning curves across D4RL and OGBench benchmarks.** For D4RL, each curve shows the mean and standard deviation over 24 random seeds. For OGBench, each curve reports the mean across five subtasks $\pm$ estimated SD, with 12 random seeds per task.

D.16 Full Ablation Results

Ablation of the RFF layer. We conduct an ablation study to assess the contribution of the RFF layer, a key component of KVR. We use KVR-MLP to isolate the effect of the RFF layer from that of LN, which can independently help mitigate overestimation. The RFF layer consists of two linear layers, sinusoidal activations, and a scaling factor. For the ablated model, we retain the linear layers, replace the sinusoidal activations with ReLU for D4RL and GeLU [10] for OGBench, and remove the scaling factor. This design keeps the overall architecture and parameter scale closely aligned with those of the original model, isolating the effect of the RFF transformation. The full results are reported in Tables 17 and 18. Removing the RFF layer leads to a substantial performance degradation across the evaluated tasks. On tasks exhibiting severe performance degradation, OOD Q-values exceed IND Q-values and grow to abnormally large magnitudes, indicating severe value overestimation (Figure 23). Specifically, the ablated model exhibits abnormally large Q-values suggestive of overestimation on 22 out of 48 evaluated tasks, whereas KVR-MLP maintains stable critic estimates across all tasks. These results

Table 17: **Full ablation results for the RFF layer on D4RL tasks (mean ± std, 5 seeds).**

GYM-MuJoCo	KVR-MLP	KVR-MLP (w/o RFF layer)
halfcheetah-random	29.5 ± 1.0	29.0 ± 2.0
hopper-random	32.2 ± 0.1	14.3 ± 9.1
walker2d-random	21.9 ± 0.5	15.1 ± 5.7
halfcheetah-medium	63.7 ± 1.1	25.0 ± 34.4
hopper-medium	104.3 ± 0.4	0.7 ± 0.0
walker2d-medium	93.6 ± 0.2	96.1 ± 0.5
halfcheetah-medium-replay	55.3 ± 0.4	54.6 ± 2.0
hopper-medium-replay	101.7 ± 0.4	103.0 ± 0.7
walker2d-medium-replay	100.7 ± 1.4	95.4 ± 6.7
halfcheetah-medium-expert	103.1 ± 2.8	101.1 ± 8.3
hopper-medium-expert	112.4 ± 0.1	0.6 ± 0.0
walker2d-medium-expert	111.7 ± 0.3	117.5 ± 0.6
Average	77.5	54.4
AntMaze	KVR-MLP	KVR-MLP (w/o RFF layer)
umaze	98.4 ± 0.5	88.6 ± 3.8
umaze-diverse	97.0 ± 3.0	90.6 ± 2.6
medium-play	96.4 ± 1.6	92.4 ± 2.4
medium-diverse	90.0 ± 7.1	73.8 ± 37.0
large-play	90.5 ± 2.5	51.0 ± 12.3
large-diverse	90.8 ± 3.7	35.4 ± 20.2
Average	93.9	72.0

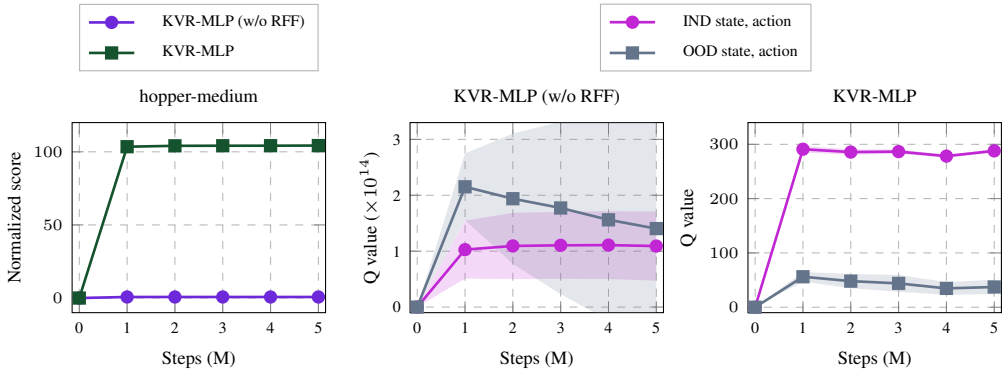


Figure 23: **Ablation of the RFF layer on Hopper-medium (mean ± std, 5 seeds).**

indicate the importance of the RFF layer in mitigating value overestimation and achieving strong performance.

Ablation of auxiliary components. We further investigate the contributions of auxiliary components in KVR, including the critic ensemble, reward scaling, and BC regularization. We conduct these ablation experiments on D4RL MuJoCo-medium and AntMaze-diverse tasks, with the results reported in Table 19.

D.17 Implementation Details

This section describes the implementation details of KVR.

Base algorithm As described in Section 3.4, we adopt TD3+BC [5] as our base algorithm. We replace its double critics with an ensemble of N critics, each equipped with an RFF layer, and follow

Table 18: **Full ablation results for the RFF layer on OGBench tasks (mean ± std, 12 seeds).**

OGBench Singletask	KVR-MLP	KVR-MLP (w/o RFF layer)
antmaze-large-navigate		
task1	97.6 ± 1.9	98.1 ± 2.2
task2	92.9 ± 2.1	59.1 ± 45.7
task3	97.8 ± 1.8	98.0 ± 0.5
task4	77.7 ± 30.5	95.9 ± 0.8
task5	98.0 ± 1.3	98.0 ± 1.5
antmaze-giant-navigate		
task1	75.9 ± 4.4	0.0 ± 0.0
task2	57.3 ± 31.4	0.0 ± 0.0
task3	16.9 ± 29.4	0.0 ± 0.0
task4	87.6 ± 5.8	0.0 ± 0.0
task5	74.4 ± 32.3	0.0 ± 0.0
humanoidmaze-medium-navigate		
task1	86.0 ± 11.8	0.0 ± 0.0
task2	97.8 ± 1.5	0.0 ± 0.0
task3	89.6 ± 4.1	0.0 ± 0.0
task4	42.3 ± 31.8	0.0 ± 0.0
task5	34.8 ± 44.4	0.0 ± 0.0
humanoidmaze-large-navigate		
task1	82.1 ± 15.2	0.0 ± 0.0
task2	27.5 ± 35.6	0.0 ± 0.0
task3	93.9 ± 2.2	0.0 ± 0.0
task4	65.0 ± 37.8	0.0 ± 0.0
task5	51.9 ± 40.3	0.0 ± 0.0
scene-play		
task1	92.6 ± 5.6	3.4 ± 4.2
task2	87.1 ± 6.2	7.1 ± 15.1
task3	73.4 ± 7.7	6.3 ± 12.3
task4	86.5 ± 10.8	69.9 ± 21.2
task5	66.1 ± 11.3	21.8 ± 25.7
puzzle-3x3-play		
task1	99.1 ± 0.7	73.8 ± 40.0
task2	98.9 ± 1.0	60.7 ± 47.1
task3	96.3 ± 2.2	98.8 ± 1.4
task4	96.4 ± 1.5	85.6 ± 32.4
task5	99.1 ± 0.4	60.6 ± 47.1
Average	78.1	31.2

the ensemble aggregation procedure of Ball et al. [2], Kim et al. [13]. Experiments using IQL and SAC+BC as alternative base algorithms are presented in Appendix D.8.

Network architectures. The actor's network is a vanilla MLP. For the critic's base network, we consider three architectures: a vanilla MLP (KVR-MLP), an MLP with LN (KVR-LN), and a residual network with LN (KVR-RES), as illustrated in Figure 18. Both the actor and the critic's base network use a depth of 2 and a width of 256 for D4RL, and a depth of 4 and a width of 512 for OGBench, following Kim et al. [13] and Park et al. [30], respectively. Both networks use ReLU activations for D4RL and GELU [10] activations for OGBench. Each critic is equipped with an RFF layer using 1024 sampled Fourier frequencies across all tasks.

Training and evaluation. We train for 5M gradient steps using Adam [14] with a batch size of 256. The actor is updated once every two critic updates. We train with 24 random seeds for D4RL and 12 for OGBench. Each evaluation uses 50 episodes for OGBench, 10 for D4RL MuJoCo, and 100 for D4RL AntMaze. Reported scores are average success rates over the final evaluation epochs for OGBench and D4RL AntMaze, and the average normalized score at the final evaluation epoch for MuJoCo. For offline-to-online experiments, we follow the training procedure of Kim et al. [13].

Table 19: **Ablation study of KVR components (mean ± std, 5 seeds).**

Task	w/o ensemble	w/o reward scale	w/o BC penalty	full
halfcheetah-medium	64.7 ± 1.1	64.7 ± 1.1	64.7 ± 1.1	64.7 ± 1.1
hopper-medium	102.3 ± 0.2	104.1 ± 0.1	104.1 ± 0.1	104.1 ± 0.1
walker2d-medium	85.4 ± 4.4	96.6 ± 1.3	96.6 ± 1.3	96.6 ± 1.3
antmaze-umaze-diverse	95.8 ± 1.9	96.0 ± 3.8	13.0 ± 20.8	98.4 ± 2.4
antmaze-medium-diverse	93.8 ± 2.6	67.2 ± 34.0	21.4 ± 13.3	94.8 ± 3.1
antmaze-large-diverse	92.4 ± 6.0	82.0 ± 3.5	0.0 ± 0.0	92.4 ± 2.5
Average	89.1	85.1	50.0	91.8

Hyperparameter tuning. All parameters except the Fourier frequencies use a fixed learning rate of 3×10^{-4}. The Fourier frequencies are fixed for D4RL MuJoCo. For D4RL AntMaze, their learning rates are fixed at 3×10^{-5} for KVR-LN and 1.5×10^{-6} for KVR-MLP. For OGBench, we fix this learning rate at 3×10^{-5} for KVR-LN and tune it for KVR-MLP.

We tune the policy noise, critic ensemble size, and RFF bandwidth for D4RL MuJoCo, and the discount factor, BC coefficient, and RFF bandwidth for OGBench. For D4RL AntMaze, we largely follow the hyperparameter settings of PARS [13] (see Appendix D.11). The final configurations for KVR-LN and KVR-MLP are provided in Tables 20 and 21.

For D4RL, we select hyperparameters using five seeds, then train the model with 19 additional seeds and report the final average over all 24 seeds. For OGBench, we select hyperparameters on the default task using two seeds and apply the selected configuration to all five tasks, each trained from scratch using 12 separate seeds.

Table 20: **General hyperparameter setting.**

Hyperparameter	Value
Learning rate	0.0003
Learning steps	5e6
Optimizer	Adam [14]
Target network smoothing coefficient	0.005
Discount factor	0.99 (default), 0.995 (antmaze), 0.999 (antmaze-giant, humanoidmaze)
Number of frequencies	1024
Critic network dimensions	[256, 256] (D4RL), [512,512,512,512] (OGBench)
Actor network dimensions	[256, 256] (D4RL), [512,512,512,512] (OGBench)
Nonlinearity	ReLU (D4RL), GeLU (OGBench), cos & sin (RFF layer)
Minibatch size	256
Policy frequency	2
Policy noise	0.2 (default), 0 (halfcheetah-medium, medium-replay, walker2d-medium)
Action noise clip	0.5
Reward scaling	1 (D4RL-MuJoCo), 10000 (D4RL-AntMaze, OGBench)

Table 21: **Task-specific hyperparameter settings for KVR-LN and KVR-MLP.**

Task	KVR-LN				KVR-MLP			
	Ensemble size	BC coefficient	Bandwidth	Fourier frequencies learning rate	Ensemble size	BC coefficient	Bandwidth	Fourier frequencies learning rate
OGBench								
antmaze-large-navigate	4	0.05	32	3×10^{-5}	4	0.05	32	3×10^{-7}
antmaze-giant-navigate	4	0.01	16	3×10^{-5}	4	0.01	256	3×10^{-5}
humanoidmaze-medium-navigate	4	0.05	16	3×10^{-5}	4	0.03	256	3×10^{-5}
humanoidmaze-large-navigate	4	0.05	32	3×10^{-5}	4	0.01	256	3×10^{-5}
scene-play	4	0.2	8	3×10^{-5}	4	0.1	64	3×10^{-5}
puzzle-3x3-play	4	0.5	8	3×10^{-5}	4	0.5	128	3×10^{-5}
D4RL-AntMaze								
AntMaze-umaze-v2	4	0.01	3	3×10^{-5}	4	0.01	32	1.5×10^{-6}
AntMaze-umaze-diverse-v2	4	0.01	3	3×10^{-5}	4	0.01	32	1.5×10^{-6}
AntMaze-medium-diverse-v2	4	0.01	3	3×10^{-5}	4	0.01	32	1.5×10^{-6}
AntMaze-medium-play-v2	4	0.01	3	3×10^{-5}	4	0.01	32	1.5×10^{-6}
AntMaze-large-diverse-v2	4	0.01	3	3×10^{-5}	4	0.01	32	1.5×10^{-6}
AntMaze-large-play-v2	4	0.01	3	3×10^{-5}	4	0.01	32	1.5×10^{-6}
D4RL-MuJoCo								
halfcheetah-random	2	0	1	0	2	0	32	0
halfcheetah-medium	2	0	3	0	2	0	4	0
halfcheetah-medium-replay	4	0	3	0	4	0	16	0
halfcheetah-medium-expert	10	0	30	0	8	0	16	0
hopper-random	6	0	1	0	6	0	4	0
hopper-medium	6	0	4	0	6	0	4	0
hopper-medium-replay	4	0	10	0	6	0	16	0
hopper-medium-expert	8	0	2	0	8	0	8	0
walker2d-random	10	0	4	0	10	0	1	0
walker2d-medium	10	0	10	0	10	0	64	0
walker2d-medium-replay	4	0	2	0	4	0	8	0
walker2d-medium-expert	6	0	9	0	8	0	32	0

NeurIPS Paper Checklist

1. **Claims**

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

   Answer: [Yes]

   Justification: The abstract and introduction clearly describe the scope of the paper, the main challenges addressed, and our contributions.

   Guidelines:

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

2. **Limitations**

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

   Answer: [Yes]

   Justification: We discuss the limitations of our method in Section 5.

   Guidelines:

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

3. **Theory assumptions and proofs**

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

   Answer: [Yes]

Justification: Appendix B provides the full assumptions and complete proofs for all theoretical results presented in the main text.

Guidelines:

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

4. **Experimental result reproducibility**

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

   Answer: [Yes]

   Justification: We describe our algorithm in detail in Section 3.4 and provide all hyperparameter settings used in our experiments in Appendix D.17.

   Guidelines:

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

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

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

Answer: [Yes]

Justification: We use publicly available benchmark datasets and provide the experimental code, along with instructions for reproducing the main results, in the supplemental material. We also plan to make the code openly accessible upon publication of the final paper.

Guidelines:

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

6. **Experimental setting/details**

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

   Answer: [Yes]

   Justification: We provide all experimental details and hyperparameter settings used in our experiments in Appendix D.

   Guidelines:

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

7. **Experiment statistical significance**

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

   Answer: [Yes]

   Justification: We report statistical information for all experiments, including the number of seeds, mean, and standard deviation.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.

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

8. **Experiments compute resources**

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

   Answer: [Yes]

   Justification: Appendix D.13 reports the hardware setup and computational cost, including per-step training time and GPU memory usage.

   Guidelines:

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

9. **Code of ethics**

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

   Answer: [Yes]

   Justification: We fully adhere to the NeurIPS Code of Ethics.

   Guidelines:

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

10. **Broader impacts**

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

    Answer: [N/A]

    Justification: This work focuses on a technical contribution to offline RL, and we do not anticipate societal impacts beyond those of general offline RL research.

    Guidelines:

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

11. **Safeguards**

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

    Answer: [N/A]

    Justification: Our work does not involve the release of high-risk models or datasets.

    Guidelines:

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

12. **Licenses for existing assets**

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

    Answer: [Yes]

    Justification: All assets used in this work are properly credited, and we follow the corresponding licenses and terms of use.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.

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

13. **New assets**

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

    Answer: [Yes]

    Justification: Our code is documented in the accompanying README file, which will be released together with the code after publication of the final paper.

    Guidelines:

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

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

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

    Answer: [N/A]

    Justification: This work does not involve crowdsourcing experiments or research with human subjects.

    Guidelines:

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

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

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

    Answer: [N/A]

    Justification: This work does not involve crowdsourcing experiments or research with human subjects.

    Guidelines:

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

16. **Declaration of LLM usage**

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

    Answer: [N/A]

    Justification: We use LLMs only as an editing aid.

    Guidelines:

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