Decoupling is the Key: Scaling Deep Value Networks in Reinforcement Learning

Yunsheng Xue **Ziyi Zhang** **Youfang Lin** **Zhihao Wu***

Beijing Jiaotong University, Beijing, China
Beijing Key Laboratory of Traffic Data Mining and Embodied Intelligence
{xueyunsheng,23125301,yflin,zhwu}@bjtu.edu.cn

Abstract

Scaling laws have driven remarkable performance breakthroughs in Computer Vision (CV) and Natural Language Processing (NLP) by increasing model depth. Compared to increasing width, deeper networks provide higher parameter efficiency and more expressive representations. However, increasing network depth in reinforcement learning (RL) still yields limited performance gains. In this paper, through theoretical and empirical analysis, we reveal that this performance degradation is primarily attributed to implicit low-rank bias and overfitting. While these two issues also exist in shallow networks, the detrimental effects caused by component coupling are significantly amplified as the network depth increases, leading to severe performance degradation. For the first issue, we theoretically prove the existence of bootstrapped spectral coupling, which causes high-frequency spectral components to couple with the low-frequency spectral components of the value network, thereby driving representations of deep value networks into severe rank collapse. For the second issue, we reveal that deep value networks severely overfit the noise induced by TD target couplings, which means the construction of TD targets is influenced by other components, such as the boundary of value distribution or current policy. Therefore, our insight is that decoupling is the key to successfully scaling deep value networks in RL. Motivated by this insight, we propose two simple yet effective solutions respectively: policy-independent spectral decoupling and boundary-target decoupling. Moreover, decoupling policy and value learning (as in AWR) is also a necessary target decoupling for deep value network training in offline RL. By integrating these decoupling components, we propose **S**pectrally D**e**coupl**e**d **D**istributional Learning (**SEED**). Empirically, to the best of our knowledge, SEED is the first approach to successfully unlock the potential of depth in both online and offline RL setting, enabling value networks to scale effectively from 3 to 18 layers with consistent performance gains. Code is available at: https://github.com/xueyunsheng/SEED

1 Introduction

In deep learning, Scaling Laws have become a fundamental principle governing the capabilities of neural networks [1–3]. Under supervised and self-supervised learning paradigms, models exhibit substantial performance improvements and enhanced generalization as the number of parameters [4, 5] and network depth increase [6, 7]. For instance, in NLP, Transformer-based pre-trained models [8, 9] have achieved remarkable language understanding and generation capabilities through deep stacking of self-attention layers, further improving the generation quality of LLMs like ChatGPT. These breakthroughs indicate that scaling network depth is a core strategy for advancing performance.

*Corresponding author.

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

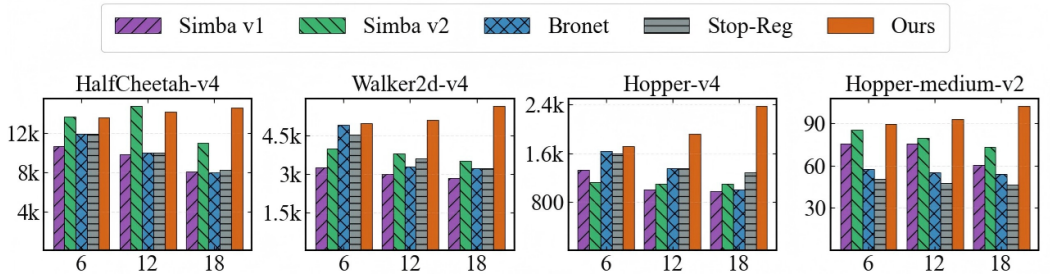

Figure 1: **Performance comparison across different network depths in MuJoCo environments.** The x-axes represent the number of layers (depth) of the value network. The y-axes report the episodic returns for online environments and the normalized D4RL scores for offline tasks (Hopper-medium-v2). Notably, our method maintains robust performance without degradation as depth increases, significantly outperforming baseline approaches at deeper layers.

Several successful paradigms have emerged for effectively scaling value network capacity in online RL [10–13]. Recent work demonstrates that incorporating residual connections, appropriate normalization techniques (e.g., LayerNorm [14]), and distributional RL [13, 15] can maintain network plasticity [16, 17] during scaling, thus mitigating performance degradation from overfitting. Despite the remarkable progress in deep reinforcement learning, scaling value networks to greater depths remains a persistent and poorly understood challenge, as shown in Figure 1. In this paper, our main contribution to address this gap is that we reveal three main coupling mechanisms that fundamentally limit the depth scalability of value networks.

First, we identify bootstrapped spectral coupling, which is a novel degradation mechanism where bootstrapping entangles the value function's high-frequency components with dominant low-frequency components, causing severe rank collapse that reduces even deep networks to near-linear functions with drastically diminished expressive power. While prior works have identified that bootstrapping induces feature rank collapse [18] through implicit feature co-adaptation [19] or harmful TD cross-covariance [20], they did not establish a clear relationship between this collapse and network depth. We further reveal that this spectral imbalance becomes increasingly severe as depth increases in this paper. Second, distributional RL [15, 13] suffers from critical TD target coupling: projecting TD targets onto fixed supports provides rich gradients for deeper training but introduces quantization and truncation errors that will be amplified with deep value network. Additionally, coupled policy-value learning dynamics in offline RL exacerbate TD target noise contamination by generating out-of-distribution (OOD) actions. Notably, while these issues exist in shallow networks, they grow progressively worse with depth, becoming the fundamental bottlenecks limiting value network scalability.

Therefore, our core insight is that decoupling is the key to scaling deep value networks in RL. First, to address the bootstrapped spectral coupling, we introduce policy-independent spectral decoupling, which employs a policy-independent linear term to explicitly decouple the deep value network and thus prevent rank collapse. Second, to resolve the distributional TD target coupling, we propose an adaptive dynamic boundary value distribution learning method that dynamically adjusts support boundaries based on distributional coverage confidence. Moreover, in offline setting, we find that decoupled policy-value learning paradigms, such as Advantage Weighted Regression (AWR) [21], are critically indispensable.

By integrating these decoupling components, we propose **S**pectrally D**e**coupl**e**d **D**istributional Learning (**SEED**) framework. Extensive experiments on the MuJoCo and AntMaze benchmarks demonstrate that SEED effectively unlocks the potential of deep architectures. By integrating SEED into the Soft Actor-Critic (SAC) [22] for online and FQL [23] for offline RL, and scaling the network architecture from 3 layers to 18 layers, we achieve an average 37.6% and 13.12% performance improvement across multiple standard benchmark environments using a single unified set of hyperparameters, without the need for tuning per-environment or per-depth.

2 Preliminaries

In this section, we briefly formalize the standard reinforcement learning setup [24], including both online and offline settings, followed by a review of the value distribution learning framework [15].

2.1 Reinforcement Learning

We consider the standard Markov Decision Process (MDP) setting defined by the tuple $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma)$, where $\mathcal{S}$ and $\mathcal{A}$ denote the state and action spaces, respectively. $\mathcal{P}(s'|s,a)$ represents the state transition dynamics, $\mathcal{R}(s,a)$ is the reward function, and $\gamma \in [0,1)$ is the discount factor. The goal of reinforcement learning is to learn a policy $\pi(a|s)$ that maximizes the expected cumulative discounted return :

$$J(\pi) = \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t r_t \right]. \tag{1}$$

2.2 Value Distribution Learning

In contrast to standard RL which estimates the scalar expected return, value distribution learning [15, 13] aims to learn the entire value distribution of the random return variable $Z^\pi(s,a)$ whose expectation is the value Q. This distribution is updated via the distributional Bellman operator $\mathcal{T}^\pi$:

$$Z^\pi(s,a) \overset{D}{=} R(s,a) + \gamma Z^\pi(s',a'), \quad a' \sim \pi(\cdot|s') \tag{2}$$

where $\overset{D}{=}$ denotes equality in distribution. In this paper, we adopt the C51 framework to approximate $Z^\pi(s,a)$ using a parametric value network $Z_\theta(s,a)$. Specifically, $Z_\theta(s,a)$ is modeled as a categorical distribution $p(s,a)$ over a fixed support set $\{z_1, \ldots, z_N\}$, where the bins z_i uniformly discretize the support range $[V_{\min}, V_{\max}]$. The value network is updated by minimizing the KL divergence between the projected distributional Bellman target and Z_θ:

$$\mathcal{L}(\theta) = \mathbb{E}_{(s,a,r,s')\sim\mathcal{D}} \left[D_{KL} \left(\Phi \mathcal{T} Z_{\theta'}(s,a) \parallel Z_\theta(s,a) \right) \right], \tag{3}$$

where Φ is the projection operator that maps the target update onto the fixed support set. In this value distribution learning framework, the support range $[V_{\min}, V_{\max}]$ is treated as fixed.

3 Spectrally Decoupled Distributional Learning

In this section, we first theoretically and empirically analyze the fundamental coupling factors that constrain the depth scaling of value networks. Motivated by this analysis, we introduce a suite of simple yet effective decoupling solutions. By integrating these components, we propose the SEED method to fundamentally address the scaling challenges of deep value networks.

3.1 Motivation

Why does scaling value network depth lead to severe performance degradation in RL? To answer this question, we first theoretically analyze how bootstrapped spectral coupling causes representation towards rank collapse and detail how TD-target coupling leads deep value networks to overfit noise. Appendix C provides formal definitions and detailed proofs for our theoretical claims.

Definition 3.1 (Spectral Decomposition of Value Function)**.** Let $\mathcal{K}(x,x') = \langle \nabla_\theta Q(x), \nabla_\theta Q(x') \rangle$ be the positive definite NTK [25] defined on the state-action space $\mathcal{X} = \mathcal{S} \times \mathcal{A}$. By Mercer's theorem [26], there exists a set of orthonormal eigenfunctions $\{\phi_k\}_{k=1}^{\infty}$ with eigenvalues $\lambda_1 \geq \lambda_2 \geq \ldots$. The target value function Q^* can be decomposed as:

$$Q^*(x) = \sum_{k=1}^{\infty} \sigma_k \phi_k(x), \tag{4}$$

where $\sigma_k = \langle Q^*, \phi_k \rangle_{\mathcal{H}}$ are the spectral coefficients. $\phi_1(x)$ captures the low-frequency component (corresponding to the largest eigenvalue), while $\{\phi_k(x)\}_{k\geq 2}$ capture the high-frequency residuals.

Deep neural networks exhibit spectral bias [27, 28], acting as low-pass filters that prioritize learning low-frequency components. To analytically characterize how this bias evolves with depth in RL, we build upon the deep linear network framework.

Assumption 3.2. The value network is approximated as a deep linear network of depth L: $Q(x;\theta) = W_L W_{L-1} \dots W_1 x$, where $W = \prod_{l=1}^{L} W_l$. Following [29], we assume isotropic input distribution, i.e., $\mathbb{E}[xx^\top] = I$. Under this condition, the singular values of the weight matrix W are isometric to the singular values of the features of network.

While practical value networks employ non-linear activations (e.g., ReLU), the deep linear assumption is a widely adopted framework for theoretical analysis [30, 31]. Prior work [29] demonstrated that the singular value dynamics in deep linear networks exhibit nonlinear learning phenomena qualitatively similar to their non-linear counterparts, yet the linear framework offers the advantage of admitting exact analytical solutions. Consequently, we adopt this framework to analytically characterize the underlying mechanism of rank collapse in RL. Intuitively, this assumption removes directional differences in the input second moments, allowing us to isolate the effect of the network transformation on the singular spectrum of the learned representations. Further details are provided in Appendix C.2. Under this assumption, we present the following lemma and theorem:

Lemma 3.3 (Bootstrapped Spectral Dynamics)**.** *In TD-learning, the target values are generated by a target network with delayed updates. Based on Assumption 3.2, the effective target singular value s_k for the k-th component is proportional to the current spectral coefficient σ_k. Let t denote the continuous-time approximation of the discrete gradient update steps. The evolution of the value network's singular value $u_k(t)$ follows the differential equation:*

$$\tau \frac{d}{dt} u_k(t) = [u_k(t)]^{2-\frac{2}{L}} \cdot (\sigma_k - u_k(t)), \tag{5}$$

where τ is a time constant.

In supervised learning, the target spectral coefficients σ_k are derived from fixed ground-truth data and remain stationary. Consequently, even if the singular values initialize at small magnitudes and learn slowly, the persistent non-zero error guarantees convergence to the true. However, in the RL setting, the target σ_k is structurally coupled with the spectral bias of the target network. As depth $L \to \infty$, the convergence rate shows a super-linear dependence on the current singular value $u_k(t)$. This intrinsic optimization bias induces a severe temporal scale separation between the fitting dynamics of different singular value modes. Since the target network inherently filters out high-frequency components, the effective target σ_k for high frequencies ($k \geq 2$) decays rapidly to zero.

Theorem 3.4 (Implicit Low-Rank Bias in Deep RL)**.** *Consider a deep value network of depth L ($L \gg 2$) with small random initialization $u_k(0) \ll \sigma_k$. Under the RL bootstrapping, due to the decay of high-frequency target coefficients ($\sigma_1 \gg \sigma_{k\geq 2}$) and the super-linear scaling of effective learning rates ($\frac{d}{dt} u_k \propto [u_k]^{2-\frac{2}{L}}$), the ratio between the dominant component and high-frequency components diverges as $t \to \infty$:*

$$\lim_{t\to\infty} \frac{u_1(t)}{u_k(t)} \to \infty, \quad \forall k \geq 2. \tag{6}$$

Bootstrapped spectral coupling. Theorem 3.4 demonstrates that as the depth increases, RL bootstrapping causes high-frequency optimization targets to couple with the low-frequency bias of the target network. Thus, high-frequency gradient signals are permanently truncated by dominant low-frequency components. This reveals that the fundamental cause of rank collapse in deep RL is bootstrapped spectral coupling, as shown in Figure 2. To address this fundamental bottleneck, the following subsection proposes a spectrally decoupled framework to explicitly mitigate this spectral coupling.

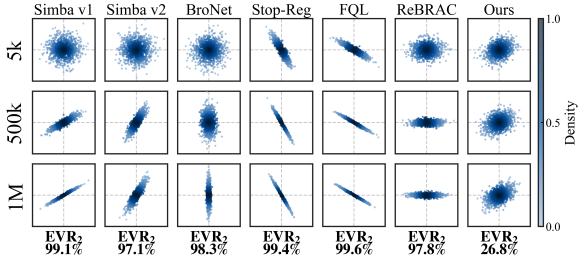

Figure 2: **Evolution of penultimate layer representations in 18-layer value networks.** We visualize the feature representations using PCA at 5k, 500k, and 1M training steps. Values below each column denote the explained variance ratio of the top two principal components ($\text{EVR}_2 = (\sigma_1^2 + \sigma_2^2)/\sum_j \sigma_j^2$) at 1M steps.

TD Target coupling. Beyond bootstrapped spectral coupling, another fundamental bottleneck in scaling value networks is the catastrophic overfitting to noise induced by target couplings. First, current scaling methods typically employ the distributional RL paradigm, providing richer gradients for large-scale parameters [13]. However, in distributional RL, projecting dynamic TD targets onto a fixed boundary leads to a TD target coupling. Specifically, if the fixed boundary is set too large, the fixed number of bins leads to severe quantization error due to insufficient granularity. Conversely, if the boundary fails to fully cover the TD target distribution, it introduces truncation error. Consequently, inappropriate boundary settings severely degrade the overall learning performance, as shown in Appendix E.1.

Additional, in the offline RL setting, the coupling between policy improvement and value evaluation in the traditional actor-critic algorithm introduces out-of-distribution (OOD) actions when computing the TD target. However, the strong fitting capacity and high local curvature of deep value networks cause them to severely overfit the noises. Further formal definitions and empirical analysis are provided in Appendices C.1 and E.2.

Notably, while these couplings paradigm have been standard practice in reinforcement learning algorithm design, its detrimental effects are drastically amplified when scaling value networks to greater depths, compared to their impact in shallow network regimes. To address these fundamental depth-scalability bottlenecks, we present a suite of simple yet highly effective decoupling solutions tailored specifically to each of the three coupling mechanisms we have uncovered.

3.2 Policy-Independent Spectral Decoupling

In this subsection, we propose a spectral decoupling framework based on late fusion [32], as shown in Figure 3, to explicitly decouple the deep value network and mitigate representation collapse.

Specifically, our framework adopts a dual-stream parallel [33] representation fusion design, explicitly decomposing the value function approximation into two independent computational paths. One path captures high-frequency spectral components, and the other models low-frequency ones. Before being fed into the output layer, we concatenate the features h_L extracted by the backbone network with the output representation of the low-frequency component, denoted as $\Phi(s) \in \mathbb{R}^k$. This concatenated feature vector is then projected by the final layer to estimate the value distribution:

$$Q(s, a) = W_{out} \cdot [h_L; \Phi(s)] + b \tag{7}$$

Fundamentally, we aim for the low-frequency component $\Phi(s)$ to be policy-independent and capture the intrinsic dynamics of the environment. This independence prevents tight coupling with the behavior policy, ensuring it neither impedes Q-network optimization nor over-constrains subsequent policy learning. Furthermore, by leveraging the dual-stream parallel representation fusion design, we explicitly construct an additive low-frequency pathway alongside the deep residual backbone. The dominant low-frequency component is effectively captured by the final linear projection, rather than being forced to propagate through all L backbone layers. Once this component is explicitly captured, the deep backbone receives error signals concentrated primarily on the high-frequency residuals, thereby successfully avoiding the coupling of low and high-frequency components within the same bootstrapped target. Therefore, the key to spectrally decoupling the value function lies in constructing a function space that is capable of precisely capturing the policy-independent low-frequency components.

We adopt a pre-training paradigm to extract the low-frequency spectral components. For offline RL, the offline dataset can directly train the low-frequency component extraction module. For online RL, we first collect a batch of data using a standard and efficient online RL algorithm, such as SAC, for pre-training. While this approach is admittedly not the most elegant, our primary goal is to provide a straightforward implementation to decouple spectral coupling and demonstrate the potential of

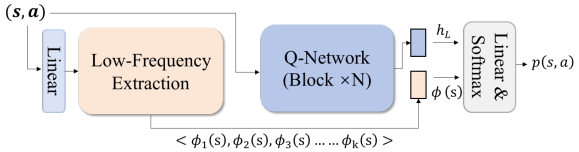

Figure 3: Given (s, a), a pre-trained Low-Frequency Extraction module extracts low-frequency latent bases $\Phi(s) = \langle \phi_1(s), \ldots, \phi_k(s) \rangle$, while an N-block Q-network computes representation h_L. Concatenating $\Phi(s)$ and h_L through a final linear projection yields the Q-value distribution $p(s, a)$.

value networks when scaled to greater depths. We further evaluate the sensitivity of this pre-training module to data quality in Appendix D.1.

We employ Koopman operator theory [34] to construct the policy-independent low-frequency extraction module. By modeling the environment dynamics in a latent linear evolution space, Koopman spectral decomposition naturally isolates low-frequency components capturing its intrinsic dynamics of the environment.

Specifically, we adopt a bilinear Koopman formulation [35] with a deep autoencoder parameterization to model the latent transition as $z_{t+1} = K_0 z_t + \sum_{i=1}^{m} a_t^i K_i z_t$. In this formulation, the drift operator K_0 governs the intrinsic dynamics of the environment independent of action (i.e., when $a_t = 0$), while K_i describes the action-dependent deviations, thereby explicitly separating the intrinsic environment drift from the action-dependent control dynamics. The detailed architecture and training objectives are provided in Appendix B.6.

After pre-training, the encoder E_ϕ maps the input state s into a latent representation $z = E_\phi(s)$.We then perform eigen-decomposition on the drift operator and select the dominant eigenvectors corresponding to the slow-varying modes to form the basis matrix V of the low-frequency spectral subspace. The resulting spectral projection function is defined as:

$$\Phi(s) = V^\top E_\phi(s), \tag{8}$$

which projects the latent representation onto the low-frequency spectral subspace to extract policy-independent low-frequency components. Finally, we concatenate the output of this low-frequency component function $\Phi(s)$ to the penultimate layer of the value network to explicitly spectrally decouple the value function.

3.3 Boundary-Target Decoupling

To address the TD target coupling induced by fixed distributional supports in distributional RL, we introduce a novel adaptive support adjustment method. The key intuition behind this approach is that by continuously adapting the support boundaries to match the evolving TD target distribution, we can ensure high-confidence coverage and eliminate the quantization and truncation errors that limit depth scalability. By introducing a confidence constraint combined with projection loss, this method dynamically adjusts the support boundaries during training, thereby explicitly decoupling the TD targets from fixed distributional supports.

First, we formulate $Z^\pi(s,a)$ as a categorical distribution defined on a symmetric support set $[-\xi, \xi]$, where ξ serves as a learnable boundary parameter. We uniformly discretize this interval into m bins, where the center of each bin is denoted as z_i ($i = 1, \ldots, m$) and the bin width is $w = \frac{2\xi}{m}$. Following [13], the Bellman target $\mu = \mathcal{T}Q(s,a)$ is constructed as a truncated Gaussian distribution centered at μ with variance σ^2 as the target distribution. Consequently, the probability $p_i(s,a)$ assigned to each discrete position z_i is calculated as:

$$p_i(s,a) = \frac{1}{2Z}\left[\text{erf}\left(\frac{z_i + w/2 - \mu}{\sqrt{2}\sigma}\right) - \text{erf}\left(\frac{z_i - w/2 - \mu}{\sqrt{2}\sigma}\right)\right] \tag{9}$$

The key to constructing value distribution learning with an adaptive support set requires minimizing projection error (defined as $\mathbb{E}_\mathcal{D}[\sum_{i=1}^{m} p_i(\xi)(z_i(\xi) - \mathcal{T}Q(s,a))]$) while simultaneously ensuring sufficient boundary coverage to accommodate the complete target distribution during Bellman updates. However, prior work [36] has relied exclusively on minimizing this projection error, neglecting the critical necessity of boundary adaptation for maintaining complete distribution coverage, which induces systematic truncation bias that is progressively amplified during the bootstrapping process. Therefore, we define the boundary update loss as follows:

$$\mathcal{L}_\xi = \min_\xi \mathbb{E}_\mathcal{D}\big[\sum_{i=1}^{m} p_i(s,a)\big(z_i(\xi) - \mathcal{T}Q(s,a)\big)^2 + \lambda \cdot \max\big(|\mu_\mathcal{B}| + k \cdot \sigma_\mathcal{B} - \xi, 0\big)\big] \tag{10}$$

where $\mathcal{B}$ denotes the current training batch, while $\mu_\mathcal{B}$ and $\sigma_\mathcal{B}$ represent the mean and standard deviation of Bellman targets $\{\mathcal{T}Q(s,a)\}_{s,a\in\mathcal{B}}$ within the batch, respectively, and k is a hyperparameter controlling the confidence level.

Intuitively, whenever the current boundary ξ fails to cover the target Q-values with sufficient confidence, the loss drives an expansion of the support set boundary to satisfy the coverage requirement.

This achieves smooth and stable boundary updates, preventing the early stagnation or late divergence of the boundary that leads to the coupling between TD targets and distributional supports, thereby effectively improving the training stability.

3.4 Policy-Value Learning Decoupling

In offline reinforcement learning, AWR framework [21] represents a common approach that follows the two-stage paradigm: first in-sample learning a value function, then extracting a policy based on the estimated advantage function. While this paradigm was not considered mandatory in the era of shallow value networks, we emphasize that with the dramatically increased representational capacity and heightened noise sensitivity of deeply scaled value networks, this decoupled value-policy learning paradigm has become an absolute necessity.

4 Experiments

4.1 Experimental Setup

We evaluate our method in both online and offline continuous control settings. For the online setting, we integrate SEED with SAC [22] and evaluate it on the MuJoCo and Antmaze environments. A warmup buffer of 5×10^5 transitions is extracted by interacting with the environment using a standard SAC algorithm, which then serves as the pre-training dataset for online setting. Noting that the total training budget is held strictly consistent for all compared algorithms. For the offline setting, we employ the corresponding D4RL datasets [37], building SEED upon the FQL(AWR) [23] and ReBRAC [38] offline RL algorithms. Here, FQL(AWR) denotes the FQL algorithm with its policy extraction component replaced by the AWR paradigm. All hyperparameters and pre-training details are provided in Appendix B.

Architectural Components. To support deep scaling and ensure gradient stability during training, we employ a variant of the Post-Activation residual network structure. Unlike the standard Post-Activation residual network, this variant removes the nonlinear activation after the residual addition, thereby achieving a Full Identity Mapping [39] to effectively alleviate the gradient vanishing problem in deep networks. Complete architectural details are provided in B.1. In this paper, we define the depth of the network as the total number of Dense layers across all residual blocks in the architecture.

4.2 Main Results

To evaluate the effectiveness of SEED in addressing the scaling bottlenecks of deep value networks, we comprehensively compared it against state-of-the-art scaling RL methods (Stop Reg, Simba v1/v2, BroNet) and SOTA offline baselines. For the compared scaling RL methods, SAC and FQL (AWR) are combined for online and offline setting, respectively. All results on scaling methods were conducted in this work. The results of all standard offline baselines, including TD3+BC, CQL, IQL, TD7+BC and DT, are token from their original publications. The main results are summarized in Table 1. Although deeper architectures (e.g., 32 layers) achieve superior performance, we adopt an 18-layer value network for our main results to achieve an optimal balance between efficiency and performance.

In the online continuous control setting, SEED demonstrates significant scaling advantages. Evaluated on the standard MuJoCo benchmark, integrating SEED with SAC [22] achieves the highest average improvement of 37.6%, and consistently outperforms other recent scaling network architectures designed for RL, such as Simba [10, 11] and Bronet [12]. Furthermore, we demonstrate that SEED also scales robustly to high-dimensional and complex environments in Appendix E.3.

On the D4RL offline benchmarks, integrating SEED with the diffusion-based offline algorithm FQL(AWR) achieves a state-of-the-art average score of 91.4. Compared to its corresponding counterparts, SEED yields substantial relative gains of 13.12%. Moreover, we also support results of different scaling methods combining ReBRAC in Appendix E.4.

Table 1: **Main results on Online and D4RL benchmarks.** Results for our method are obtained using an 18-layer value network over 5 random seeds. The best scores and competitive scores are highlighted. In Part I, values in parentheses indicate the percentage improvement relative to the SAC baseline.

Part I: Online Environments

Method	HalfCheetah-v4 Return	Hopper-v4 Return	Walker2d-v4 Return	Antmaze-v4 (3 tasks) Success Rate	Avg. Improv. (%)
SAC[22]	11,507.1 (+0.0%)	2,196.7 (+0.0%)	4,632.1 (+0.0%)	20.2 (+0.0%)	+0.0%
Stop-Reg[13]	939.1 (-91.8%)	1,278.9 (-41.8%)	4,001.3 (-13.6%)	10.7 (-47.0%)	-48.6%
Simba v1[10]	10,791.6 (-6.2%)	1,419.1 (-35.4%)	3,272.2 (-29.4%)	31.7 (+56.9%)	-3.5%
Simba v2[11]	13,718.6 (+19.2%)	1,318.5 (-40.0%)	4,028.4 (-13.0%)	35.1 (+73.8%)	+10.0%
BroNet[12]	11,992.9 (+4.2%)	1,631.0 (-25.8%)	4,978.0 (+7.5%)	30.3 (+50.0%)	+9.0%
OFENet[40]	12,216.9 (+6.2%)	1,669.4 (-24.0%)	4,891.3 (+5.6%)	30.6 (+51.5%)	+9.8%
Ota21[41]	11,398.1 (-0.9%)	1,581.6 (-28.0%)	5,359.5 (+15.7%)	31.1 (+54.0%)	+10.2%
StableGrad[42]	16,771.7 (+45.8%)	1,993.1 (-9.3%)	5,491.3 (+18.5%)	32.9 (+62.9%)	+29.5%
SAC+SEED-18	14,612.7 (+27.0%)	2,381.4 (+8.4%)	5,676.3 (+22.5%)	38.9 (+92.6%)	+37.6%

Part II: D4RL Offline Benchmarks

Method	HalfCheetah m	HalfCheetah m-r	HalfCheetah m-e	Hopper m	Hopper m-r	Hopper m-e	Walker2d m	Walker2d m-r	Walker2d m-e	Umaze Play	Umaze Div	Medium Play	Medium Div	Large Play	Large Div	Avg. -
TD3+BC[43]	48.1	44.6	93.7	59.1	52.0	98.1	84.3	81.0	110.5	78.6	71.4	10.6	3.0	0.2	0.0	55.7
CQL[44]	46.7	45.5	76.8	59.3	78.9	79.9	81.4	79.9	108.5	74.0	84.0	61.2	53.7	15.8	14.9	64.0
IQL[45]	47.4	44.2	86.7	66.3	94.7	91.5	78.3	73.9	109.3	87.5	62.2	71.2	70.0	39.6	47.5	71.4
TD7-BC[46]	58.1	53.8	104.6	76.1	91.1	108.2	91.1	89.7	111.8	-	-	-	-	-	-	-
DT[47]	42.6	36.6	86.8	67.6	82.7	107.6	74.0	66.6	108.1	59.2	53.0	0.0	0.0	0.0	0.0	52.3
FQL (AWR)	48.6	51.3	108.1	70.3	65.3	98.4	90.2	86.9	100.9	95.1	87.2	75.1	69.7	80.1	84.8	80.8
Stop-Reg[13]	42.6	33.4	67.1	52.4	19.3	54.0	64.4	17.2	54.0	37.5	29.7	21.1	19.4	10.2	5.9	35.2
Simba v1[10]	49.3	45.7	90.9	74.0	95.3	60.8	79.3	91.9	100.3	42.1	39.7	30.9	23.1	36.3	33.8	59.6
Simba v2[11]	52.3	46.8	95.8	82.6	101.2	63.8	85.5	95.4	111.4	53.3	40.0	33.9	40.5	39.9	51.3	66.2
BroNet[12]	50.9	45.4	94.9	82.5	101.8	66.3	71.8	93.3	107.9	43.3	56.6	23.1	26.9	13.3	15.7	59.6
OFENet[40]	51.2	47.1	96.3	81.2	73.6	94.8	88.9	94.8	103.2	47.1	43.6	21.7	20.1	19.7	18.3	60.1
Ota21[41]	51.8	49.5	97.1	82.7	61.2	99.1	89.9	81.7	101.3	49.3	40.1	27.4	24.6	30.1	29.3	61.0
StableGrad[42]	56.6	48.9	106.7	86.3	71.1	99.6	89.3	89.1	101.7	67.3	63.6	58.1	53.9	49.7	41.3	72.2
FQL (AWR)+SEED-18	61.1	54.4	110.2	89.4	79.9	108.0	100.2	90.4	109.3	100.0	95.7	87.5	89.4	95.9	100.0	91.4

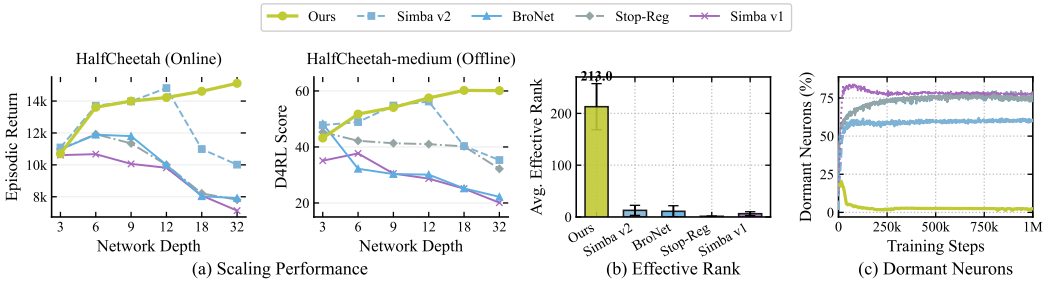

Figure 4: **Comprehensive analysis of scaling performance and internal network dynamics. (a) Scalability analysis across varying network depths.** We report the episodic return on HalfCheetah (left) in online settings and the normalized D4RL score on HalfCheetah-medium-v2 (right) in offline settings, as the value network depth increases from 3 to 32 layers. **(b) Comparison of the average effective rank** across different methods. **(c) Evolution of the dormant neuron ratio** during training.

4.3 Scalability Analysis

To answer the question of whether the performance improvement truly comes from the expansion of depth, we conducted a systematic scalability analysis to observe the evolution of algorithm performance as the value network depth increases. We progressively expanded the value network depth from 3 to 32 layers to demonstrate SEED's potential in deep value networks.

Scaling Performance. As shown in Figure 4(a), SEED maintains a steady improvement in performance as the depth increases, achieving stability even at 32 layers. In contrast, existing scaling methods exhibit varying degrees of performance degradation with increasing network depth. While Simba V2 maintains some stability at medium depths, it suffers from significant performance collapse at larger scales. This suggests that relying solely on architectural regularization is insufficient to fundamentally overcome the low-rank bias and overfitting issues that emerge as the network scales.

Internal Network Dynamics. To further investigate the internal dynamics supporting this scalability, we analyzed the representation properties of 32-layer networks using two key metrics: the Effective Rank of the penultimate layer representations and the Dormant Neuron Ratio [48].

Table 2: **Ablation Study.** Performance comparison of different architectural designs. Results show the impact of removing or replacing key components in our method across both online and offline environments. SEED represents our complete proposed architecture. All results are averaged over 5 random seeds.

Design (idx) Original → Changed	HalfCheetah-v4	Hopper-v4	Walker2d-v4	H-m-v2 (offline)
SEED	**14,612.7**	**2,381.4**	**5,676.3**	**61.1**
Policy-Independent Spectral Decoupling				
(a) Spectral Decoupling → No Spectral Decoupling	11,204.5	1,450.2	3,892.1	45.2
(b) Koopman → MLP	12,543.8	1,892.7	4,510.6	45.8
Boundary-target decoupling				
(c) Adaptive Boundary → No Adaptive Boundary	10,172.3	1,231.8	3,120.5	39.7
(d) Confidence → No Confidence	11,850.3	1,510.8	4,120.5	48.3
(e) Categorical Loss → MSE Loss	12,800.6	1,750.3	4,680.2	51.1
Policy-Value learning Decoupling				
(f) AWR → BC	-	-	-	41.2

As illustrated in Figure 4(b), existing scaling methods lacking explicit spectral decoupling suffer from severe representation collapse. Despite employing techniques such as LayerNorm and value distributional learning, these methods fail to prevent the rank collapse induced by training dynamics as depth increases. While prior work [49] suggests that Layer Normalization can mitigate network linearization, our empirical analysis in Appendix E.7 demonstrates that LayerNorm [49] alone remains inadequate to prevent rank collapse as depth increases. In contrast, our method explicitly decouples the value network, effectively avoiding the optimization shortcuts that lead to collapse, thereby maintaining the effective rank of the scaled value network at a consistently high level.

Furthermore, compared to existing methods, SEED maintains an extremely low ratio of dormant neurons, as shown in Figure 4(c). In summary, these results demonstrate that our method facilitates the learning of rich representations while maximizing the parameter efficiency of scaled value networks.

4.4 Ablation Studies

Policy-Independent Spectral Decoupling. Spectral decoupling is crucial for the performance of deep value networks (Table 2(a)); omitting this module leads to a severe performance drop. An equally important design is using the Koopman operator rather than a standard MLP to extract low-frequency components (Table 2(b)). In contrast, Koopman leverages a bilinear evolution model to explicitly isolate the action-free drift operator and construct the low-frequency subspace, thereby fundamentally guaranteeing the policy-independence of the low-frequency components. Furthermore, we also explicitly isolate the effects of our spectral decoupling mechanism from the generic benefits of representation pre-training in Appendix E.5.

Boundary-target decoupling. Employing a confidence-aware adaptive boundary update effectively decouples the TD target from the distributional supports, thereby significantly enhancing the algorithm's performance. Reverting to standard distributional learning with fixed boundaries (Table 2(c)) or a traditional MSE loss (Table 2(e)) causes a substantial performance drop. Without the confidence-aware constraint (Table 2(d)), the updated boundary fails to fully cover the TD target distribution, which inevitably introduces truncation noise, ultimately degrading the performance. Furthermore, while a recent alternative technique [50] addresses support mismatch in value distribution learning via reward scaling, we find that in offline settings, it fundamentally distorts the advantage regression weights in AWR, leading to severe performance degradation, as detailed in Appendix E.8.

Policy-Value learning Decoupling. In offline reinforcement learning, Behavior Cloning (BC) is another widely used conservative regularization technique. BC-regularized algorithms typically adopt an alternating policy-value update paradigm, where out-of-distribution (OOD) actions generated by the policy are used to construct TD targets, thereby introducing policy-value learning coupling. Our ablation studies show that reverting from the decoupled AWR paradigm to this coupled learning scheme results in significant performance degradation (Table 2(f)). Given the high local curvature of

deep value networks, querying out-of-distribution (OOD) actions in traditional Actor-Critic methods induces severe value overestimation and leading to policy performance degradation.

5 Conclusion

In this paper, we investigated the scalability bottleneck of value networks in RL. Through theoretical and empirical analysis, we identified that performance degradation in deep networks fundamentally stems from the bootstrapped spectral coupling and TD target couplings. Crucially, the detrimental effects caused by these couplings are significantly amplified as network depth increases, directly leading to severe performance collapse.

To address these fundamental bottlenecks, we argue that decoupling is the key to scaling value networks to greater depths. We propose simple yet effective decoupling solutions targeting each of the identified coupling points, and validate their effectiveness in enabling depth scaling through extensive comparative experiments and ablation studies.

The limitation of our spectral decoupling framework is that it requires pre-training on early experiences collected by an efficient and standard online RL algorithm and may appear less elegant, while two critical points must be emphasized. First, all our experiments are conducted under an identical total training budget to ensure fairness in the comparison. Second, the primary focus of this work is to uncover the coupling bottlenecks that have long hindered RL depth scaling, and to demonstrate that deep scaling of value networks is indeed feasible and effective through straightforward decoupling approaches. We hope this work provides new path forward for future research on RL depth scaling and we will continue to explore more unified training paradigms in subsequent work.

Acknowledgments and Disclosure of Funding

The authors received no external third-party funding in direct support of this work and declare no competing interests.

References

[1] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. ImageNet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 25, 2012.

[2] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[3] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

[4] Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Liam Fedus, Maarten Li, Zhao Tu, Kevin Wu, Denny Zhou, et al. Emergent abilities of large language models. *Transactions on Machine Learning Research (TMLR)*, 2022.

[5] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 30, 2017.

[6] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations (ICLR)*, 2015.

[7] Zhi-Hua Zhou and Ji Feng. Deep forest. *National Science Review*, 6(1):74–86, 2019.

[8] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the North American Chapter of the Association for Computational Linguistics (NAACL)*, pages 4171–4186, 2019.

[9] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, et al. Language models are few-shot learners. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 33, pages 1877–1901, 2020.

[10] H Lee, D Hwang, D Kim, H Kim, J J Tai, K Subramanian, et al. Simba: Simplicity bias for scaling up parameters in deep reinforcement learning. *arXiv preprint arXiv:2410.09754*, 2024.

[11] H Lee, Y Lee, T Seno, D Kim, P Stone, and J Choo. Hyperspherical normalization for scalable deep reinforcement learning. *arXiv preprint arXiv:2502.15280*, 2025.

[12] M Nauman, M Ostaszewski, K Jankowski, P Miłoś, and M Cygan. Bigger, regularized, optimistic: Scaling for compute and sample efficient continuous control. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 37, pages 113038–113071, 2024.

[13] Jesse Farebrother, Joshua Orbay, Quan Vuong, Adrien Ali Taïga, Yevgen Chebotar, Ted Xiao, et al. Stop regressing: Training value functions via classification for scalable deep RL. *arXiv preprint arXiv:2403.03950*, 2024.

[14] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

[15] Marc G Bellemare, Will Dabney, and Rémi Munos. A distributional perspective on reinforcement learning. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 449–458, 2017.

[16] Clare Lyle, Zico Zheng, Evgenii Nikishin, Bernardo Avila Pires, Razvan Pascanu, and Will Dabney. Understanding plasticity in neural networks. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 23190–23211, 2023.

[17] Shibhansh Dohare, J Fernando Hernandez-Garcia, Parash Rahman Lan, A Rupam Mahmood, and Richard S Sutton. Loss of plasticity in deep continual learning. *Nature*, 632(8026):768–774, 2024.

[18] Aviral Kumar, Rishabh Agarwal, Dibya Ghosh, and Sergey Levine. Implicit under-parameterization inhibits data-efficient deep reinforcement learning. *arXiv preprint arXiv:2010.14498*, 2020.

[19] Aviral Kumar, Rishabh Agarwal, Tengyu Ma, Aaron Courville, George Tucker, and Sergey Levine. Value-based deep reinforcement learning requires explicit regularization. 2022.

[20] Nan Qiao, Sheng Yue, Shuning Wang, Yongheng Deng, and Ju Ren. Less is more: Clustered cross-covariance control for offline RL. In *The Fourteenth International Conference on Learning Representations*, 2026.

[21] Xue Bin Peng, Aviral Kumar, Grace Zhang, and Sergey Levine. Advantage-weighted regression: Simple and scalable off-policy reinforcement learning. *arXiv preprint arXiv:1910.00177*, 2019.

[22] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pages 1861–1870. Pmlr, 2018.

[23] Seohong Park, Qiyang Li, and Sergey Levine. Flow q-learning. In *Forty-second International Conference on Machine Learning*, 2025.

[24] Richard S Sutton, Andrew G Barto, et al. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge, 1998.

[25] Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural tangent kernel: Convergence and generalization in neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 31, pages 8571–8580, 2018.

[26] James Mercer. Functions of positive and negative type, and their connection with the theory of integral equations. *Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character*, 209(441-458):415–446, 1909.

[27] Nasim Rahaman, Aristide Baratin, Devansh Arpit, Felix Draxler, Min Lin, Fred Hamprecht, Yoshua Bengio, and Aaron Courville. On the spectral bias of neural networks. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 5301–5310, 2019.

[28] Zhi-Qin John Xu, Yaoyu Zhang, Tao Luo, Yanyang Xiao, and Zheng Ma. Frequency principle: Fourier analysis sheds light on deep neural networks. *arXiv preprint arXiv:1901.06523*, 2019.

[29] Andrew M Saxe, James L McClelland, and Surya Ganguli. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. In *International Conference on Learning Representations (ICLR)*, 2014.

[30] Sanjeev Arora, Nadav Cohen, and Elad Hazan. On the optimization of deep networks: Implicit acceleration by overparameterization. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 244–253, 2018.

[31] Kenji Kawaguchi. Deep learning without poor local minima. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 29, 2016.

[32] Andrej Karpathy, George Toderici, Sanketh Shetty, Thomas Leung, Rahul Sukthankar, and Li Fei-Fei. Large-scale video classification with convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1725–1732, 2014.

[33] Karen Simonyan and Andrew Zisserman. Two-stream convolutional networks for action recognition in videos. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 27, 2014.

[34] Milan Korda and Igor Mezić. On convergence of extended dynamic mode decomposition to the koopman operator. *Journal of Nonlinear Science*, 28(2):687–710, 2018.

[35] Debdipta Goswami and Derek A Paley. Global bilinearization and controllability of control-affine nonlinear systems: A koopman spectral approach. In *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, pages 6107–6112. IEEE, 2017.

[36] Chen Chen, Daming Shi, Yixiu Mao, and Xiangyang Ji. Adaptive HL-gaussian: A value function learning method with dynamic support adjustment, 2025.

[37] 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.

[38] Denis Tarasov, Vladislav Kurenkov, Alexander Nikulin, and Sergey Kolesnikov. Revisiting the minimalist approach to offline reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 36, pages 11592–11620, 2023.

[39] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 630–645. Springer, 2016.

[40] Kei Ota, Tomoaki Oiki, Devesh Jha, Toshisada Mariyama, and Daniel Nikovski. Can increasing input dimensionality improve deep reinforcement learning? In *International conference on machine learning*, pages 7424–7433. PMLR, 2020.

[41] Kei Ota, Devesh K Jha, and Asako Kanezaki. Training larger networks for deep reinforcement learning. *arXiv preprint arXiv:2102.07920*, 2021.

[42] Roger Creus Castanyer, Johan Obando Ceron, Lu Li, Pierre-Luc Bacon, Glen Berseth, Aaron Courville, and Pablo Samuel Castro. Stable gradients for stable learning at scale in deep reinforcement learning. *Advances in Neural Information Processing Systems*, 38:35051–35087, 2026.

[43] Scott Fujimoto and Shixiang Shane Gu. A minimalist approach to offline reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 34, pages 20132–20145, 2021.

[44] Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative Q-learning for offline reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 33, pages 1179–1191, 2020.

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

[46] Scott Fujimoto, Wei-Di Chang, Edward Smith, Shixiang Shane Gu, Doina Precup, and David Meger. For sale: State-action representation learning for deep reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 36, pages 61573–61624, 2023.

[47] Lili Chen, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Michael Laskin, Pieter Abbeel, Aravind Srinivas, and Igor Mordatch. Decision transformer: Reinforcement learning via sequence modeling. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 34, pages 15084–15097, 2021.

[48] Ghada Sokar, Rishabh Agarwal, Pablo Samuel Castro, and Utku Evci. The dormant neuron phenomenon in deep reinforcement learning. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 32145–32168, 2023.

[49] Yang Yue, Rui Lu, Bingyi Kang, Shiji Song, and Gao Huang. Understanding, predicting and better resolving q-value divergence in offline-rl. *Advances in Neural Information Processing Systems*, 36:60247–60277, 2023.

[50] Donghu Kim, Youngdo Lee, Minho Park, Kinam Kim, I Made Aswin Nahendra, Takuma Seno, Sehee Min, Daniel Palenicek, Florian Vogt, Danica Kragic, Jan Peters, Jaegul Choo, and Hojoon Lee. Flashsac: Fast and stable off-policy reinforcement learning for high-dimensional robot control, 2026.

[51] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale, 2021.

[52] Xiaohua Zhai, Alexander Kolesnikov, Neil Houlsby, and Lucas Beyer. Scaling vision transformers, 2022.

[53] Scott Reed, Konrad Zolna, Emilio Parisotto, Sergio Gomez Colmenarejo, Alexander Novikov, Gabriel Barth-Maron, et al. A generalist agent. *arXiv preprint arXiv:2205.06175*, 2022.

[54] Kuang-Huei Lee, Ofir Nachum, Mengjiao Yang, Lisa Lee, Daniel Freeman, Sergio Guadarrama, et al. Multi-game decision transformers. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 35, pages 27921–27936, 2022.

[55] Michal Nauman, Marek Cygan, Carmelo Sferrazza, Aviral Kumar, and Pieter Abbeel. Bigger, regularized, categorical: High-capacity value functions are efficient multi-task learners, 2025.

[56] Yevgen Chebotar, Quan Vuong, Alex Irpan, Karol Hausman, Fei Xia, Yao Lu, Aviral Kumar, Tianhe Yu, Alexander Herzog, Karl Pertsch, Keerthana Gopalakrishnan, Julian Ibarz, Ofir Nachum, Sumedh Sontakke, Grecia Salazar, Huong T Tran, Jodilyn Peralta, Clayton Tan, Deeksha Manjunath, Jaspiar Singht, Brianna Zitkovich, Tomas Jackson, Kanishka Rao, Chelsea Finn, and Sergey Levine. Q-transformer: Scalable offline reinforcement learning via autoregressive q-functions, 2023.

[57] Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. Mastering diverse domains through world models. *arXiv preprint arXiv:2301.04104*, 2023.

[58] Vincent Micheli, Eloi Alonso, and François Fleuret. Transformers are sample-efficient world models. *arXiv preprint arXiv:2209.00588*, 2022.

[59] Eloi Alonso, Adam Jelley, Vincent Micheli, Anssi Kanervisto, Amos J Storkey, Tim Pearce, and François Fleuret. Diffusion for world modeling: Visual details matter in atari. *Advances in Neural Information Processing Systems*, 37:58757–58791, 2024.

[60] Tete Xiao, Ilija Radosavovic, Trevor Darrell, and Jitendra Malik. Masked visual pre-training for motor control. *arXiv preprint arXiv:2203.06173*, 2022.

[61] Suraj Nair, Aravind Rajeswaran, Vikash Kumar, Chelsea Finn, and Abhinav Gupta. R3m: A universal visual representation for robot manipulation. *arXiv preprint arXiv:2203.12601*, 2022.

[62] Nils Bjorck, Carla P Gomes, and Kilian Q Weinberger. Towards deeper deep reinforcement learning with spectral normalization. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 34, pages 8242–8255, 2021.

[63] Florin Gogianu, Tudor Berariu, Mihaela C Rosca, Claudia Clopath, Lucian Busoniu, and Razvan Pascanu. Spectral normalisation for deep reinforcement learning: An optimisation perspective. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 3734–3744, 2021.

[64] Evgenii Nikishin, Max Schwarzer, Pierluca D'Oro, Pierre-Luc Bacon, and Aaron Courville. The primacy bias in deep reinforcement learning. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 16828–16847, 2022.

[65] Mohammad Pezeshki, Oumar Kaba, Yoshua Bengio, Aaron C Courville, Doina Precup, and Guillaume Lajoie. Gradient starvation: A learning proclivity in neural networks. *Advances in Neural Information Processing Systems*, 34:1256–1272, 2021.

[66] Yicheng Li, Zixiong Yu, Guhan Chen, and Qian Lin. On the eigenvalue decay rates of a class of neural-network related kernel functions defined on general domains. *Journal of Machine Learning Research*, 25(82):1–47, 2024.

[67] Luke Melas-Kyriazi, Christian Rupprecht, Iro Laina, and Andrea Vedaldi. Deep spectral methods: A surprisingly strong baseline for unsupervised semantic segmentation and localization. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 8364–8375, 2022.

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

[69] Aäron Van Den Oord, Nal Kalchbrenner, and Koray Kavukcuoglu. Pixel recurrent neural networks. In *International conference on machine learning*, pages 1747–1756. PMLR, 2016.

[70] Alex Kendall, Hayk Martirosyan, Saumitro Dasgupta, Peter Henry, Ryan Kennedy, Abraham Bachrach, and Adam Bry. End-to-end learning of geometry and context for deep stereo regression. In *Proceedings of the IEEE international conference on computer vision*, pages 66–75, 2017.

[71] Gregory Rogez, Philippe Weinzaepfel, and Cordelia Schmid. Lcr-net++: Multi-person 2d and 3d pose detection in natural images. *IEEE transactions on pattern analysis and machine intelligence*, 42(5):1146–1161, 2019.

[72] Ehsan Imani, Kai Luedemann, Sam Scholnick-Hughes, Esraa Elelimy, and Martha White. Investigating the histogram loss in regression. *arXiv preprint arXiv:2402.13425*, 2024.

[73] Will Dabney, Mark Rowland, Marc Bellemare, and Rémi Munos. Distributional reinforcement learning with quantile regression. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

[74] Will Dabney, Georg Ostrovski, David Silver, and Rémi Munos. Implicit quantile networks for distributional reinforcement learning. In *International conference on machine learning*, pages 1096–1105. PMLR, 2018.

[75] Pushi Zhang, Xiaoyu Chen, Li Zhao, Wei Xiong, Tao Qin, and Tie-Yan Liu. Distributional reinforcement learning for multi-dimensional reward functions. *Advances in Neural Information Processing Systems*, 34:1519–1529, 2021.

[76] Ludvig Killingberg and Helge Langseth. The multiquadric kernel for moment-matching distributional reinforcement learning. *Transactions on Machine Learning Research*, 2023.

[77] Thanh Nguyen-Tang, Sunil Gupta, and Svetha Venkatesh. Distributional reinforcement learning via moment matching. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pages 9144–9152, 2021.

[78] Xiaoteng Ma, Junyao Chen, Li Xia, Jun Yang, Qianchuan Zhao, and Zhengyuan Zhou. Dsac: Distributional soft actor-critic for risk-sensitive reinforcement learning. *Journal of Artificial Intelligence Research*, 83, 2025.

[79] Gabriel Barth-Maron, Matthew W Hoffman, David Budden, Will Dabney, Dan Horgan, Dhruva Tb, Alistair Muldal, Nicolas Heess, and Timothy Lillicrap. Distributed distributional deterministic policy gradients. *arXiv preprint arXiv:1804.08617*, 2018.

[80] Rahul Singh, Keuntaek Lee, and Yongxin Chen. Sample-based distributional policy gradient, 2020.

[81] Prajit Ramachandran, Barret Zoph, and Quoc V Le. Searching for activation functions. *arXiv preprint arXiv:1710.05941*, 2017.

[82] Milan Korda and Igor Mezić. Linear predictors for nonlinear dynamical systems: Koopman operator meets model predictive control. *Automatica*, 93:265–278, 2018.

[83] Matthias Weissenbacher, Samarth Sinha, Animesh Garg, and Kawahara Yoshinobu. Koopman Q-learning: Offline reinforcement learning via symmetries of dynamics. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 23645–23667, 2022.

[84] Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ international conference on intelligent robots and systems*, pages 5026–5033. IEEE, 2012.

[85] Yuval Tassa, Yotam Doron, Alistair Muldal, Tom Erez, Yazhe Li, Diego de Las Casas, David Budden, Abbas Abdolmaleki, Josh Merel, Andrew Lefrancq, et al. Deepmind control suite. *arXiv preprint arXiv:1801.00690*, 2018.

Appendix Table of Contents

- **A Related Work**
  - A.1 Network Scaling
  - A.2 Spectral Decoupling
  - A.3 Value Distribution Learning
- **B Implementation Details**
  - B.1 Network Architectures
  - B.2 Koopman Pre-training Details
  - B.3 Pseudocode
  - B.4 Hyperparameter Settings
  - B.5 Environments and Datasets
  - B.6 Detailed Formulation of the Koopman Extraction Module
- **C Proofs for Theoretical Motivation**
  - C.1 Formal Mathematical Definitions of Coupling Mechanisms
  - C.2 Clarification on Assumption 3.2
  - C.3 Proofs for Lemma 3.3
  - C.4 Proof of Theorem 3.4
- **D Extensive Sensitivity Analyses**
  - D.1 Sensitivity to Pre-training Quality
  - D.2 Sensitivity to the Latent Dimension
  - D.3 Sensitivity to Adaptive-Boundary Hyperparameters
- **E Additional Results**
  - E.1 Motivation: TD Target Coupling with Fixed Distributional Boundaries
  - E.2 Motivation: Hessian Norm and Overfitting to OOD Actions
  - E.3 Effectiveness of SEED on High-Dimensional Complex Tasks
  - E.4 Effectiveness of SEED on ReBRAC Backbone
  - E.5 The Necessity of Spectral Decoupling for Deep Scaling
  - E.6 The Necessity of In-Sample Policy-Evaluation Decoupling
  - E.7 The Insufficiency of Layer Normalization for Preventing Rank Collapse
  - E.8 The Insufficiency of Reward Scaling for Boundary-Target Decoupling
  - E.9 Computational Cost and Parameter Overhead
  - E.10 Empirical Training Dynamics on RL Benchmarks

A Related Work

A.1 Network Scaling

Network scaling has demonstrated substantial performance gains in fields such as NLP [9, 8, 3] and CV [51, 52, 2, 1]. In reinforcement learning, network scaling has also exhibited significant advantages in multi-task learning [53–55], sequence-based modeling paradigms [47, 56] and world models [57–59]. Particularly in high-dimensional visual tasks[60, 61], deep networks are capable of extracting more expressive features compared to their shallow counterparts. Existing scaling methods typically incorporate architectural designs and training strategies, such as Spectral Normalization [62, 63], Layer Normalization [14], periodic parameter resets[64, 48], auxiliary representation learning [40, 41] and distributional learning [13, 15], to maintain training stability in large-scale networks. Recent studies [10, 12, 11] further indicate that leveraging residual architectures to scale the width of value networks leads to superior performance in online RL. In addition, multi-skip connections and the Kron optimizer have been shown to stabilize gradient flow and support the scaling of both network depth and width [42].

However, existing studies on value network scaling have primarily focused on online RL, while systematic evaluations of depth scaling in offline RL remain limited. In this work, we observe that when employing these methods to scale value networks in depth, performance suffers from varying degrees of degradation. Our analysis identifies the low-rank bias that becomes more pronounced with increasing network depth as a key factor contributing to this degradation. To address this issue, we propose explicitly spectrally decoupling the value network to mitigate the impact of low-rank bias, thereby unlocking the scaling potential of deep networks in online and offline RL.

A.2 Spectral decoupling

Deep learning theory [27, 25, 28] suggests that neural networks exhibit a spectral bias in their optimization dynamics, prioritizing the fitting of low-frequency components of the target function. Influenced by this bias, models tend to minimize the loss function by exploiting these low-frequency features, causing the gradients of the overall loss to rapidly decay to zero. Consequently, deep networks often fall into the trap of Gradient Starvation [65]. Some theoretical studies [66] characterize this spectral bias as a form of early-stopping bias in neural networks.

Spectral decoupling methods aim to mitigate the impact of this bias by adjusting the gradient descent trajectory, which prevents the network from converging solely on a few dominant features to facilitate the capture of high-frequency components. Currently, existing methods [67] primarily employ an implicit decoupling strategy by adding an L2 penalty to the output logits. In this work, we observe that in reinforcement learning, deep value networks remain susceptible to spectral bias, with their representation exhibiting varying degrees of collapse as network depth increases. Motivated by this observation, we theoretically characterize the low-rank implicit bias inherent in deep value networks. Our analysis demonstrates that under bootstrapping TD-learning [68], this intrinsic bias results in the permanent truncation of high-frequency signals, necessitating a structurally explicit decoupling approach to maintain representational stability.

A.3 Value distribution learning

Reformulating regression tasks as classification problems has been demonstrated to significantly enhance algorithm performance in many fields [69–71]. [72] introduced the HL-Gauss cross-entropy loss, showing that it yields performance gains even in small-scale supervised regression tasks. In RL, distributional methods learn the distribution of returns rather than only their expected value. One approach represents the return distribution using probabilities over a fixed set of return values and learns these probabilities through cross-entropy minimization, as in C51 [15]. Quantile-based methods [73, 74] instead learn return quantiles through quantile regression. Beyond these approaches, moment-matching methods [75–77] learn a set of particles to represent the return distribution and minimize the maximum mean discrepancy between the predicted and target distributions.

In continuous control, prior works [78–80] incorporate distributional critics into actor–critic frameworks to improve value estimation and control performance. More recently, [13] demonstrated that classification-based value learning effectively stabilizes gradient norms when scaling deep networks. However, projecting dynamic TD targets onto a fixed categorical support inevitably introduces a

support mismatch. To mitigate this, recent methods like FlashSAC [50] propose adaptive reward scaling to compress the targets within predefined boundaries. Inspired by these findings, we adopt cross-entropy training and introduce an adaptive support mechanism for boundary–target decoupling. By dynamically adapting the support boundaries during training, our approach explicitly achieves decoupling, effectively disentangling the evolving TD targets from the categorical distributional supports.

B Implementation Details

In this section, we provide a comprehensive description of the environments and datasets, network architectures, and hyperparameter settings to ensure the reproducibility of our results. All experiments were implemented in **PyTorch 2.4.1** and executed on an **NVIDIA RTX A4000 GPU**.

B.1 Network Architectures

Deep Critic Network (Q-Network). In the Q-network part, as shown in Figure 5, to further support deep scaling in large-scale networks and ensure gradient stability during training, we propose a variant of the Post-Activation residual network structure. Unlike the standard Post-Activation residual network, this variant removes the nonlinear activation function after the residual addition, thereby achieving a Full Identity Mapping [39] to effectively alleviate the gradient vanishing problem in deep networks.

Considering the continuous distribution shift induced by policy updates in offline RL, we incorporate a Layer Normalization within each residual block to standardize the feature distribution in the latent space. Furthermore, to mitigate the neuron dormancy [48] caused by the hard truncation of the ReLU activation function in the negative region and preserve network plasticity, we select the Swish activation [81] function, which provides rich gradient signals for deep networks. Therefore, the computation process of the l-th residual block is as follows:

$$h_{l+1} = h_l + \mathcal{F}(h_l; \theta_l) \tag{11}$$

where the residual mapping function $\mathcal{F}$ is defined as:

$$\mathcal{F}(x) \triangleq W_2 \cdot \text{Swish}(\text{LN}(W_1 x; \gamma, \beta)) \tag{12}$$

$$\text{LN}(u) = \frac{u - \mu_u}{\sqrt{\sigma_u^2 + \epsilon}} \odot \gamma + \beta \tag{13}$$

where μ_u and σ_u^2 are the mean and variance of the input u, ϵ is a small constant for numerical stability, $\odot$ denotes the Hadamard product, and γ, β are learnable affine parameters.

Actor (Policy Network). The actor network also incorporates spectral information. It encodes the state s and Koopman features $\Phi(s)$ separately. The Koopman features are projected via a LayerNorm-normalized MLP. These embeddings are concatenated and passed through a 2-layer MLP (hidden dim 256) with **ReLU** activations. The final output is squashed by a **Tanh** function and scaled to the maximum action limit.

Koopman Bilinear Module. The module consists of an Encoder and a Decoder, both implemented as 3-layer MLPs with **Tanh** activations to bound the latent dynamics. The latent evolution is explicitly modeled using a bilinear form without bias terms: $z' = K_0 z + \sum a_i K_i z$.

B.2 Koopman Pre-training Details

Prior to the formal policy learning phase, the Koopman module requires pre-training on an initial dataset to extract the policy-independent low-frequency features. To maintain a unified framework across different RL paradigms, the pre-training dataset is constructed as follows:

- **Offline Setting:** The module is pre-trained directly on the provided static offline datasets (e.g., D4RL benchmarks).

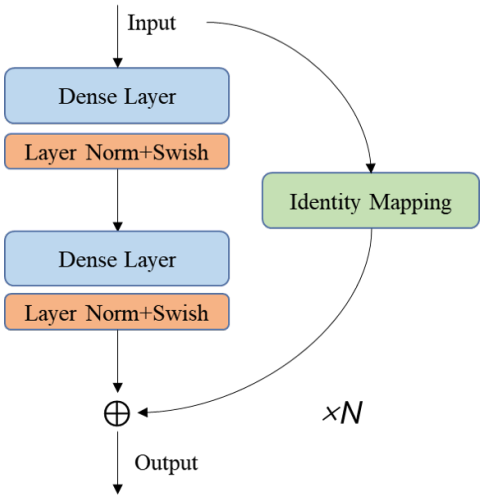

Figure 5: **The architecture of our proposed Q-network.** We employ a modified Post-Activation residual structure featuring a Full Identity Mapping to ensure gradient stability in deep networks. Each residual block integrates Layer Normalization (LN) to stabilize latent feature distributions against offline RL distribution shifts, and uses the Swish activation function to mitigate neuron dormancy and preserve plasticity. Note that the non-linear activation is omitted after the residual addition to maintain a clean identity path.

- **Online Setting:** We first collect a warmup buffer of 5×10^5 transitions by interacting with the environment using a standard SAC baseline, which then serves as the pre-training dataset.

Following the data collection, the pre-training process proceeds identically for both settings based on the following details:

- **Objective:** We minimize a joint loss comprising a multi-step prediction loss and a reconstruction loss, weighted by $\lambda_{rec} = 0.1$:

$$\mathcal{L}_{total} = \mathcal{L}_{pred} + 0.1 \cdot \mathcal{L}_{rec} \tag{14}$$

- **Spectral Basis Selection:** Post-training, we perform eigen-decomposition on the drift operator K_0. The eigenvalues λ are sorted based on their proximity to the unit circle (i.e., minimizing $||\lambda| - 1|$) to identify the most persistent, slow-varying dynamical modes. We select the top-k eigenvectors to construct the low-frequency spectral basis.

B.3 Pseudocode

The complete algorithmic process of SEED is summarized in Algorithm 1. Notably, our spectral decoupling and adaptive boundary modules serve as a universal value network enhancement. Therefore, SEED can be naturally integrated into both online SAC (via maximum entropy objective) and offline RL settings (via in-sample policy extraction) with minimal modifications.

B.4 Hyperparameter Settings

Table 15 details the hyperparameters used in our experiments.

B.5 Environments and Datasets

To comprehensively evaluate SEED across different RL paradigms, we conduct experiments in both online and offline settings. We adopt the standard Gymnasium `-v4` environments for online continuous control and the D4RL benchmark [37] for offline evaluations. Our evaluation focuses on two distinct domains that pose different fundamental challenges: MuJoCo locomotion tasks and AntMaze sparse-reward navigation tasks.

MuJoCo Locomotion Tasks. These environments involve high-dimensional continuous state and action spaces with dense reward signals. They are designed to test the agent's ability to learn complex local dynamics, balance, and forward progress.

- **Online Setting:** We evaluate on four standard environments: `HalfCheetah-v4`, `Hopper-v4`, `Walker2d-v4`.
- **Offline Setting (D4RL):** We evaluate on HalfCheetah, Hopper, and Walker2d. For each environment, we assess performance across three dataset types representing different quality levels:
  - *Medium:* Generated by a policy trained to approximately 1/3 of the expert performance. This tests the ability to learn from sub-optimal, single-mode data.
  - *Medium-Replay:* Consists of all transitions recorded during the training of the medium policy, characterized by high variance and a mixture of behaviors, testing robustness to noisy data.
  - *Medium-Expert:* A mixture of expert demonstrations and sub-optimal data, testing the ability to disentangle good behaviors from bad ones.

AntMaze (Sparse-Reward Navigation Tasks). These environments require an 8-DoF Ant robot to navigate through complex mazes to reach a target goal. Unlike locomotion tasks, AntMaze features strictly sparse binary rewards (1 for reaching the goal, 0 otherwise) and requires extensive long-horizon planning and temporal credit assignment.

- **Online Setting:** We utilize the `AntMaze-v4` environments to evaluate the agent's online exploration and representation learning capabilities under sparse rewards. Specifically, we evaluate across three progressive task complexities: `AntMaze_UMaze-v4`, `AntMaze_Medium-v4`, and `AntMaze_Large-v4`.
- **Offline Setting (D4RL):** The offline datasets largely consist of undirected trajectories ("Play" data) that do not necessarily reach the goal from the start, rigorously testing the algorithm's trajectory stitching capabilities. We evaluate on three maze topologies:
  - *AntMaze-Umaze:* Features a simple U-shaped layout (`antmaze-umaze` and `antmaze-umaze-diverse`). These datasets serve as a fundamental sanity check for handling sparse rewards in a low-complexity topology.
  - *AntMaze-Medium:* Consists of a U-shaped maze with a moderate path length. This primarily tests the agent's ability to stitch multiple short trajectory segments to solve a sparse-reward task.
  - *AntMaze-Large:* Features a significantly larger maze requiring hundreds of time steps to solve. This environment serves as a rigorous benchmark for the algorithm's capacity to capture global low-frequency value components over extremely long horizons.

B.6 Detailed Formulation of the Koopman Extraction Module

In the SEED framework, we leverage Koopman operator theory [34] to construct the policy-independent low-frequency extraction module. By linearizing the non-linear system dynamics, Koopman spectral decoupling naturally isolates the slow-varying low-frequency components via dominant eigenfunctions. This ensures that the resulting representation captures the intrinsic environmental dynamics and remains strictly policy-independent.

To approximate the infinite-dimensional Koopman operator within a computationally tractable finite-dimensional space, we employ a deep autoencoder structure (E_ϕ, D_ψ) to learn an approximately invariant observation subspace. For the dynamic evolution, we adopt the bilinear formulation from [82, 83] to explicitly decouple the inherent transition laws of the dynamical system from the action-related control laws. The dynamic evolution in the latent space is formulated as:

$$z_{t+1} = \underbrace{K_0 z_t}_{\text{Free Dynamics}} + \underbrace{\sum_{i=1}^{m} a_t^i K_i z_t}_{\text{Control Dynamics}}, \tag{15}$$

where $z_t = E_\phi(s_t) \in \mathbb{R}^N$ represents the encoded latent state. Here, $K_0 \in \mathbb{R}^{N\times N}$ is the *drift operator*, which characterizes the inherent transition laws of the system under no control input. The term $K_i \in \mathbb{R}^{N\times N}$ is the *control operator* corresponding to the i-th dimension of the continuous action a^i, describing the control laws associated with that specific action channel, and m denotes the total dimension of the action space.

To ensure that the latent space retains essential physical information of the original states and strictly obeys the bilinear dynamic evolution, we jointly minimize the reconstruction and dynamics prediction losses. Let $K_a = K_0 + \sum_{i=1}^{m} a^i K_i$ denote the control-dependent Koopman operator. The joint self-supervised objective is defined as:

$$\min_{\phi,\psi,\mathcal{K}} \mathcal{L} = \underbrace{\mathbb{E}_{s,a,s'\sim\mathcal{D}} \left\| s' - D_\psi \left(K_a E_\phi(s) \right) \right\|_2^2}_{\text{Prediction}} + \underbrace{\mathbb{E}_{s\sim\mathcal{D}} \left\| s - D_\psi(E_\phi(s)) \right\|_2^2}_{\text{Reconstruction}}, \tag{16}$$

where $\mathcal{K} = \{K_0, K_1, \ldots, K_m\}$ represents the set of learnable operator matrices.

By pre-training this Koopman bilinear model via self-supervision on the offline dataset $\mathcal{D}$, we can exploit the spectral properties of the drift operator K_0 to characterize the low-frequency function space. Crucially, since K_0 is mathematically decoupled from the action inputs, its spectral structure exclusively reflects the environment's underlying physical laws, invariant to any behavioral policy.

Formally, we perform eigen-decomposition on the fixed drift operator K_0: $K_0 V = V\Lambda$. We then select the top k dominant eigenvectors $V_k = \{v_1, \ldots, v_k\}$—which correspond to the eigenvalues with the largest real parts (i.e., the slow-varying modes)—as the basis for the low-frequency subspace. Finally, we construct the spectral projection function:

$$\Phi(s) = V_k^\top E_\phi(s), \tag{17}$$

which explicitly projects the encoded latent state into this low-frequency subspace. The output of $\Phi(s)$ is then concatenated to the penultimate layer of the deep value network, structurally injecting a policy-independent, low-frequency inductive bias to mitigate spectral bias and representation rank collapse during deep scaling.

C Proofs for Theoretical Motivation

In this appendix, we first provide formal mathematical definitions for the three coupling mechanisms discussed in Section 3.1. Following this, we clarify Assumption 3.2 regarding singular value correspondence under isotropic inputs, followed by the detailed proofs for Lemma 3.3 and Theorem 3.4. We analyze the spectral dynamics of deep value networks under the specific non-stationary conditions of Reinforcement Learning (RL). Our analysis bridges the exact solutions of deep linear network dynamics derived by [29] with the spectral bias findings of [27].

C.1 Formal Mathematical Definitions of Coupling Mechanisms

To formalize the concept of coupling discussed in Section 3.1, we define it fundamentally through non-zero gradient dependencies. Specifically, we call critic updates *coupled* with respect to a component c_t if the target $Y_t = F(\bar{\theta}_t, c_t)$ or its gradient explicitly depends on c_t, i.e., $D_c F \neq 0$. Based on this general principle, we provide the rigorous definitions for the three coupling mechanisms addressed in this work.

Definition C.1 (Bootstrapped Spectral Coupling)**.** Let $\{\phi_k\}$ be orthonormal NTK eigenfunctions and decompose the target critic as

$$\bar{Q}_t(x) = \sum_j \bar{q}_{j,t}\phi_j(x). \tag{18}$$

For a fixed policy π, the target $Y_t = r + \gamma P^\pi \bar{Q}_t$ yields the mode coefficients:

$$\sigma_{k,t} = \langle Y_t, \phi_k \rangle = r_k + \gamma \sum_j B^\pi_{kj} \bar{q}_{j,t}, \qquad \text{where} \quad B^\pi_{kj} = \langle P^\pi \phi_j, \phi_k \rangle. \tag{19}$$

Taking the partial derivative yields:

$$\frac{\partial \sigma_{k,t}}{\partial \bar{q}_{j,t}} = \gamma B^\pi_{kj}. \tag{20}$$

This explicit dependence on the target critic defines bootstrapped spectral coupling: nonzero low–high blocks of B^π directly couple frequency components.

Definition C.2 (TD-Target Coupling)**.** For a scalar TD target Y_t, let

$$p_t^\xi = \Pi_{\xi,m}(Y_t) \tag{21}$$

be its projection onto m discrete bins over the support $[-\xi, \xi]$. Because both the bin locations and their projected probabilities inherently depend on the boundary parameter ξ, we have:

$$\frac{\partial p_t^\xi}{\partial \xi} \neq 0. \tag{22}$$

This defines TD-target coupling. For a fixed number of bins m, forcing a larger ξ reduces truncation errors but inevitably increases the bin width $2\xi/m$ (quantization error); conversely, a smaller ξ reverses this trade-off, strongly coupling the distribution learning fidelity to the fixed boundary ξ. **We empirically demonstrate the impact of this exact trade-off on deep scaling in Appendix E.1.**

Definition C.3 (Coupling between Policy and Value Evaluation)**.** In traditional **offline actor-critic learning**, the scalar TD target Y_t evaluates the current evolving policy π_ψ:

$$Y_t = r + \gamma \bar{Q}(s', \pi_\psi(s')). \tag{23}$$

Consequently, the target inherently depends on the actor's parameters ψ, yielding a non-zero gradient dependency via the chain rule:

$$D_\psi Y_t = \gamma \nabla_a \bar{Q}(s', a)\big|_{a=\pi_\psi(s')} \nabla_\psi \pi_\psi(s') \neq 0. \tag{24}$$

Because the actor π_ψ is simultaneously optimized to maximize Q, it outputs out-of-distribution (OOD) actions that extrapolate beyond the dataset's conditional support. The strict coupling ($D_\psi Y_t \neq 0$) allows these OOD actions to directly bias the value targets without supervision.

C.2 Clarification on Assumption 3.2: Singular Value Correspondence Under isotropic inputs

In Assumption 3.2, we state that the singular values of the weight matrix directly characterize the empirical feature singular values under isotropic inputs. Here we provide the explicit derivation.

Let $X = [x_1, \ldots, x_n]$ and the penultimate feature matrix $H = W_h X$. Under $XX^\top/n = I$,

$$\frac{1}{n} HH^\top = W_h \frac{XX^\top}{n} W_h^\top = W_h W_h^\top.$$

Therefore,

$$\sigma_i(H/\sqrt{n}) = \sigma_i(W_h).$$

Thus, W_h's singular-value dynamics characterize empirical feature-rank collapse.

C.3 Proofs for Lemma 3.3

We consider the value network approximated as a deep linear network of depth L (Assumption 3.2). The function is parameterized as $Q(x) = W_L W_{L-1} \ldots W_1 x$, where the total transformation is $W_{\text{total}} = \prod_{l=1}^{L} W_l$.

Step 1: Gradient Descent Dynamics in Deep Linear Networks. Following the framework of Saxe et al, we analyze the gradient descent dynamics on the squared error loss $E = \frac{1}{2}\|y - Q(x)\|^2$. For a network with isotropic inputs $\mathbb{E}[xx^\top] = I$, the learning dynamics of the weights can be decomposed into independent modes using the Singular Value Decomposition (SVD) of the input-output correlation matrix $\Sigma^{yx} = USV^\top$.

We focus on the "decoupled" initial condition regime, where the matrices W_l are aligned such that their singular value vectors match. In this regime, the dynamics of the entire network can be described by the evolution of scalar singular values $u_k(t)$ associated with the k-th mode. Let $a_l(t)$ denote the singular value of the l-th layer matrix W_l for a specific mode k. The end-to-end singular value for this mode is:

$$u_k(t) = \prod_{l=1}^{L} a_l(t). \tag{25}$$

Assuming balanced initialization and gradients, the magnitudes across layers remain approximately equal during training, i.e., $a_l(t) = a(t)$ for all l. Thus, we relate the layer-wise magnitude to the total magnitude by:

$$a(t) = [u_k(t)]^{1/L}. \tag{26}$$

The gradient descent update for a single layer weight $a(t)$ is driven by the error signal scaled by the transmission through the other $L-1$ layers. The gradient flow is given by:

$$\frac{d}{dt}a(t) \propto -\frac{\partial E}{\partial a} = (\sigma_k - u_k(t)) \cdot \frac{\partial u_k}{\partial a}, \tag{27}$$

where σ_k is the target singular value (spectral coefficient). Using $u_k = a^L$, the partial derivative is $\frac{\partial u_k}{\partial a} = L \cdot a^{L-1}$. Substituting this into the flow equation, we obtain:

$$\frac{d}{dt}a(t) \propto (\sigma_k - u_k(t)) \cdot L \cdot a^{L-1}. \tag{28}$$

Step 2: Deriving the End-to-End Singular Value Evolution. We are interested in the evolution of the effective singular value $u_k(t)$. By the chain rule:

$$\frac{d}{dt}u_k(t) = \frac{d}{dt}(a^L) = L \cdot a^{L-1} \cdot \frac{d}{dt}a(t). \tag{29}$$

Substituting the expression for $\frac{d}{dt}a(t)$ derived above:

$$\frac{d}{dt}u_k(t) \propto L \cdot a^{L-1} \cdot \left[(\sigma_k - u_k(t)) \cdot L \cdot a^{L-1}\right]. \tag{30}$$

Absorbing constants L^2 into a time constant τ, and grouping the terms for a:

$$\tau\frac{d}{dt}u_k(t) = [a(t)]^{2L-2} \cdot (\sigma_k - u_k(t)). \tag{31}$$

We now substitute $a(t) = [u_k(t)]^{1/L}$ back into the equation to express the dynamics solely in terms of u_k:

$$[a(t)]^{2L-2} = \left([u_k(t)]^{1/L}\right)^{2L-2} = [u_k(t)]^{\frac{2L-2}{L}} = [u_k(t)]^{2-\frac{2}{L}}.$$

This yields the differential equation characterizing the dynamics:

$$\tau\frac{d}{dt}u_k(t) = [u_k(t)]^{2-\frac{2}{L}} \cdot (\sigma_k - u_k(t)). \tag{32}$$

C.4 Proof of Theorem 3.4 (Implicit Low-Rank Bias)

Step 1: Asymptotic Dynamics at Large Depth. We analyze the singular value dynamics derived in Lemma 1 as the network depth $L \to \infty$. The exponent limits to:

$$\lim_{L\to\infty}\left(2 - \frac{2}{L}\right) = 2.$$

Consequently, the dynamics in Eq. (32) converge to:

$$\tau\frac{d}{dt}u_k(t) = u_k(t)^2(\sigma_k(t) - u_k(t)). \tag{33}$$

This creates a "rich-get-richer" dynamic regime where the learning speed is super-linearly dependent on the current magnitude u_k. Modes initialized with small magnitudes remain trapped near zero for exponentially long periods unless driven by a strong forcing term σ_k.

Step 2: The RL Twist – Spectral Filtering in the Target. In supervised learning, σ_k is a constant ground-truth value. Even if $u_k(0)$ is small, a non-zero constant σ_k eventually drives convergence. In Reinforcement Learning, however, the target is non-stationary and bootstrapped:

$$\sigma_k(t) = \langle r, \phi_k\rangle + \gamma\langle Q_{\text{target}}, \phi_k\rangle. \tag{34}$$

Let $u_k^-(t)$ be the singular values of the target network. Since the target network is a lagged copy of the online network (e.g., via Polyak averaging), its spectral composition reflects the past state of the online network.

Spectral Bias: Rahaman et al. [27] established that neural networks learn low-frequency components ($k = 1$) significantly faster than high-frequency components ($k \geq 2$). Consequently, during the early and intermediate phases of training, we observe a dichotomy:

- **Low Frequency ($k = 1$):** The network fits the dominant trend rapidly. Thus, $u_1^-(t)$ is large, providing a strong bootstrapping signal $\sigma_1(t) \approx \langle r, \phi_1 \rangle + \gamma u_1^-(t)$.
- **High Frequency ($k \geq 2$):** Due to spectral bias, the network has not yet learned these components. Thus, $u_k^-(t) \approx 0$. This implies the target signal is dominated solely by the immediate reward projection: $\sigma_k(t) \approx \langle r, \phi_k \rangle$.

Crucially, since the target network acts as a *low-pass filter*, any high-frequency noise or signal in the target values is attenuated, starving the high-frequency modes of necessary gradient feedback.

Step 3: Divergence of the Spectral Ratio. We consider the ratio $R_k(t) = \frac{u_1(t)}{u_k(t)}$ for $k \geq 2$. Differentiating with respect to time using the quotient rule and substituting Eq. (33):

$$\begin{aligned} \frac{d}{dt} R_k &= \frac{\dot{u}_1 u_k - u_1 \dot{u}_k}{u_k^2} \\ &= \frac{\frac{1}{\tau} u_1^2 (\sigma_1 - u_1) u_k - u_1 \frac{1}{\tau} u_k^2 (\sigma_k - u_k)}{u_k^2}. \end{aligned} \tag{35}$$

Simplifying under the valid assumption of small random initialization ($u_k \ll u_1$ and $u_k \ll \sigma_k$):

$$\tau \frac{d}{dt} R_k \approx \frac{u_1^2 \sigma_1 u_k}{u_k^2} = u_1^2 \frac{\sigma_1}{u_k} = u_1 \sigma_1 R_k. \tag{36}$$

This differential equation approximates to exponential growth for the ratio R_k. Furthermore, a distinct failure mode arises from the feedback loop. Following Saxe et al., the time t_{learn} required to escape the initial saddle point for mode k scales as $O(1/\sigma_k)$.

- For $k = 1$, σ_1 is large (amplified by the target network), so $t_{\text{learn}}^{(1)}$ is small.
- For $k \geq 2$, bootstrapping fails. The target network cannot provide a teaching signal for mode k. σ_k contains only the high-frequency residual of the reward r, which is often sparse. If σ_k remains below the threshold required to escape the depth-induced saddle point, $u_k(t)$ remains permanently trapped at its initialization magnitude ϵ.

As $t \to \infty$, we have:

$$u_1(t) \to \sigma_1^* \quad \text{(Converged)}, \qquad u_k(t) \approx \epsilon \quad \text{(Stalled)}.$$

Therefore:

$$\lim_{t \to \infty} \frac{u_1(t)}{u_k(t)} \approx \frac{\sigma_1^*}{\epsilon} \to \infty. \tag{37}$$

This completes the proof. The interaction between the super-linear gradient scaling of deep networks ($u^{2-2/L}$) and the high-frequency filtering of the target network prevents the effective propagation of high-rank gradients, resulting in permanent rank collapse. □

D Extensive Sensitivity Analyses

In this section, we conduct extensive empirical evaluations to demonstrate the robustness of SEED against variations in pre-training data quality, latent dimensions, and adaptive-boundary hyperparameters.

D.1 Sensitivity to Pre-training Quality

To evaluate the robustness of our online Koopman pre-training paradigm, we investigate the sensitivity of downstream performance to the pre-training data quality along two dimensions: the amount of warmup data (data quantity) and the collection distribution (data coverage). Specifically, we first vary the warmup buffer size among 0.3M, 0.5M, and 0.8M transitions using a fixed collector. Then, maintaining a fixed buffer size of 0.5M, we compare datasets collected via a completely random policy against those sampled from an evolving-policy replay buffer.

As demonstrated in Table 3, all configurations achieve comparable final returns across `HalfCheetah-v4`, `Hopper-v4`, and `Walker2d-v4` (averaged over 5 random seeds). These results confirm that because the self-supervised pre-training focuses purely on capturing the intrinsic transition dynamics, adding more samples or altering the data-collection policy has a negligible impact on downstream performance, provided that sufficient state-action coverage is achieved.

Table 3: Sensitivity analysis of online Koopman pre-training quality. The downstream performance (average return over 5 seeds) remains highly robust across different pre-training data quantities and collection strategies.

Setting	Buffer Size	Collection Strategy	HalfCheetah-v4	Hopper-v4	Walker2d-v4
Data quantity	0.3M	Fixed collector	14126.4	2274.1	5431.6
Data quantity	0.5M	Fixed collector	14475.2	2346.8	5583.2
Data quantity	0.8M	Fixed collector	14608.5	2412.3	5614.5
Data coverage	0.5M (default)	Random policy	14612.7	2381.4	5676.3
Data coverage	0.5M	Evolving-policy replay	14385.9	2327.6	5542.7

D.2 Sensitivity to the Latent Dimension

To investigate the algorithmic sensitivity to the latent dimension, we vary the number of retained Koopman modes $k \in \{8, 16, 24, 32\}$. As shown in Table 4, the final returns demonstrate remarkable consistency regardless of the chosen k across both online and offline benchmarks.

Since the objective of this module is to decouple spectral representations rather than to completely reconstruct the value function, a small number of dominant modes is sufficient to capture the low-frequency dynamics, leaving the main network to efficiently fit the high-frequency residuals. Thus, we select $k = 24$ as the default configuration to optimally balance representation capacity and computational cost.

Table 4: Sensitivity to the number of retained Koopman modes k (5-seed average). The performance remains highly consistent, indicating that extracting a small subset of dominant low-frequency modes is sufficient for effective decoupling.

Retained Modes k	HalfCheetah-medium-v2	Hopper-medium-play-v2	HalfCheetah-v4
8	58.2	76.9	13246.1
16	60.5	78.5	14124.1
24 (default)	**61.1**	**79.9**	**14612.7**
32	61.3	78.7	14531.2

D.3 Sensitivity to Adaptive-Boundary Hyperparameters

Regarding the boundary-target decoupling mechanism, we evaluate the algorithmic sensitivity by varying the boundary confidence penalty λ and the confidence level k_c across online and offline tasks.

As detailed in Table 5 and Table 6, the empirical results exhibit high stability across a wide range of values in both settings. Because this boundary constraint is activated exclusively when the current support set inadequately covers the target distribution, it dynamically deactivates once the coverage requirement is met during training. Consequently, the mechanism effectively achieves boundary-target decoupling and strictly prevents extrapolation errors, without introducing any adverse impact on the final convergence performance.

Table 5: Sensitivity to the boundary confidence penalty λ (5-seed average).

Boundary Penalty λ	HalfCheetah-medium-v2	Hopper-medium-play-v2	HalfCheetah-v4
5	56.3	74.3	14493.6
10 (default)	**58.4**	**77.5**	**14612.7**
15	61.1	79.9	14371.9
20	59.7	78.1	13962.1

Table 6: Sensitivity to the confidence level k_c (5-seed average).

Confidence Level k_c	HalfCheetah-medium-v2	Hopper-medium-play-v2	HalfCheetah-v4
4	56.4	71.2	13427.5
8	59.7	76.3	14493.6
10 (default)	**61.1**	**79.9**	**14612.7**
12	58.6	73.3	13991.4

E Additional Results

E.1 Motivation: TD Target Coupling with Fixed Distributional Boundaries

To empirically demonstrate the severe limitations of employing fixed distributional boundaries in continuous control tasks, we evaluate the performance of a standard distributional RL baseline across various boundary scales (δ) on the *HalfCheetah-v4* and *Walker2d-v4* environments. In standard distributional RL, the dynamic TD target is projected onto a discrete support defined by a fixed number of bins within a predefined range $[-\delta, \delta]$.

As illustrated in Figure 6, this fixed-boundary paradigm introduces a fundamental target coupling dilemma that severely bottlenecks scaling:

- **Truncation Error (Inadequate δ):** When the boundary is heavily restricted (e.g., $\delta = 10$), it fails to cover the shifting value scales during training. The target distribution is severely clipped at the edges, causing the network to lose critical magnitude information, which leads to complete learning failure.
- **Quantization Error (Excessive δ):** Conversely, if an excessively large boundary is applied (e.g., $\delta = 5000$) to safely encompass all possible returns, the fixed number of bins is stretched over a massive range. This lack of granularity causes severe quantization error, blurring the subtle value distinctions necessary for accurate policy evaluation and destabilizing the training process.

These results clearly indicate that an optimal, fixed boundary is highly task-dependent and fundamentally incapable of adapting to the dynamically growing value scales in online RL.

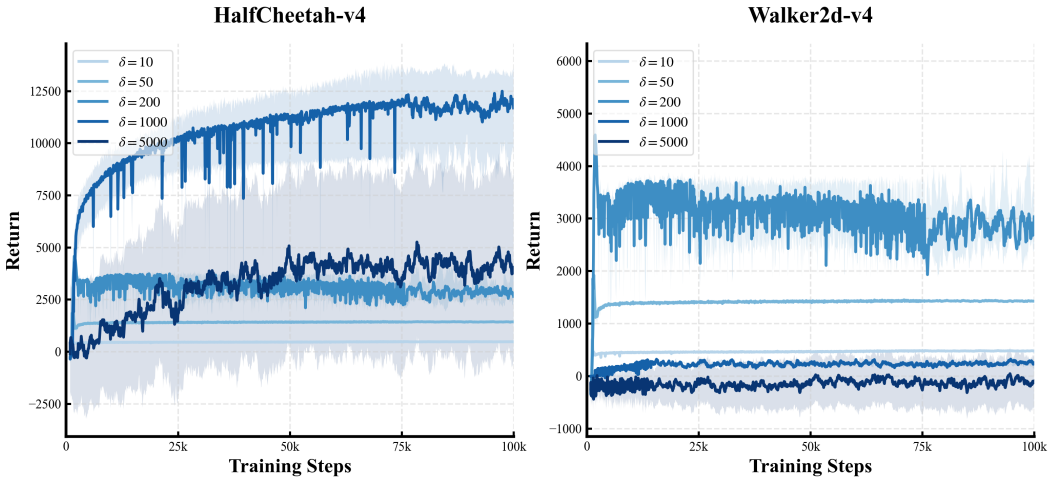

Figure 6: **Impact of fixed distributional boundaries in online environments.** The learning curves on HalfCheetah-v4 (left) and Walker2d-v4 (right) demonstrate the fundamental dilemma of target coupling: excessively small boundaries ($\delta = 10$) cause catastrophic truncation errors, while excessively large boundaries ($\delta = 5000$) introduce severe quantization errors, both of which significantly degrade performance.

E.2 Motivation: Hessian Norm and Overfitting to OOD Actions

In the offline RL setting, traditional actor-critic frameworks (e.g., TD3+BC) suffer from a severe coupling between policy improvement and value evaluation. Due to their strong expressive power, deep value networks are highly susceptible to overfitting the sparse or noisy signals within these OOD regions.

To quantify this phenomenon, we monitor the **Hessian norm** with respect to actions, defined as $\|\nabla_a^2 Q(s,a)\|$, which represents the local curvature of the value manifold. A high Hessian norm indicates a sharp, non-smooth value surface, often a precursor to training instability and catastrophic overestimation.

As shown in Figure 7, we compare the Hessian norm evolution of TD3+BC and TD3+AWR across different network depths ($L = 3, 9, 18$):

- **TD3+BC:** As the network depth increases to 18 layers, the Hessian norm escalates and remains at a high magnitude (10^{-2} to 10^{-3}). This confirms that deep networks under BC-regularization heavily overfit to OOD noise, leading to a highly rugged value landscape that hinders stable scaling.
- **TD3+AWR:** By contrast, TD3+AWR—which utilizes an in-sample advantage-weighted regression—effectively decouples value evaluation from OOD actions. Regardless of the network depth, the Hessian norm consistently drops and stabilizes at a significantly lower level ($\sim 10^{-5}$), demonstrating that the AWR paradigm provides the necessary gradient stability for scaling deep value networks.

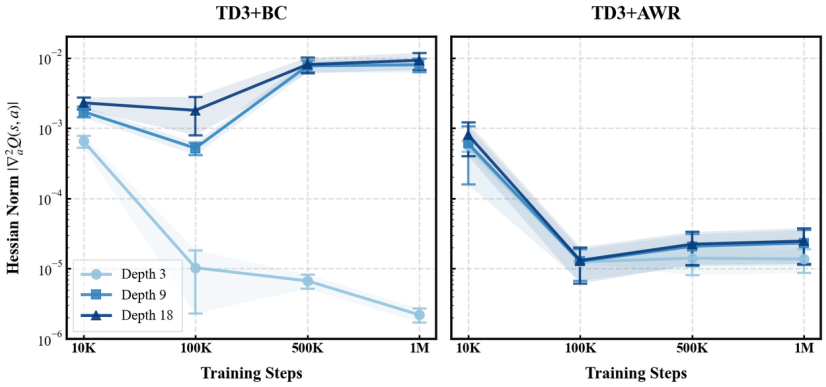

Figure 7: **Evolution of Hessian Norm $\|\nabla_a^2 Q(s,a)\|$ during training.** The results compare the standard BC-regularized paradigm (left) with the in-sample AWR paradigm (right) across various network depths. In the BC setting, deeper networks exhibit higher local curvature, indicating severe overfitting to OOD actions. In the AWR setting, the value landscape remains smooth regardless of depth, facilitating stable deep scaling.

E.3 Effectiveness of SEED on High-Dimensional Complex Tasks

In this section, we additionally evaluate SEED on `Humanoid-v4` [84] and DeepMind Control (DMC) Hard Dog [85] tasks (`Dog-Trot` and `Dog-Run`). These environments are characterized by higher dimensionality and complex articulated dynamics compared to standard locomotion benchmarks.

Under the same experimental protocol, we compare SEED against three representative value-network scaling baselines: SimbaV1, SimbaV2, and BroNet. As shown in Table 7, SEED consistently outperforms all baselines across these complex tasks.

E.4 Effectiveness of SEED on ReBRAC Backbone

In the offline RL setting, decoupling policy improvement from value evaluation is crucial for algorithmic performance. Therefore, we select the ReBRAC algorithm, which natively employs

Table 7: Performance comparison on high-dimensional and complex continuous control tasks. Results report the average return over 5 random seeds.

Task	Obs. / Act. Dim.	SimbaV1	SimbaV2	BroNet	SEED (Ours)
Humanoid-v4	376 / 17	6513.72	10546.13	4757.21	**13117.38**
Dog-Trot	223 / 38	824.28	861.61	817.19	**891.33**
Dog-Run	223 / 38	544.79	562.11	491.30	**601.73**

advantage-weighted policy extraction, as the foundational backbone across all evaluated scaling methods, including our SEED. The full experimental results under this setting are presented in Table 8.

Table 8: Full evaluation results on D4RL offline benchmarks compared with ReBRAC and scaling baselines. The highest score for each task is highlighted in **bold**.

Environment	ReBRAC	Simba V1	Simba V2	BroNet	SEED (Ours)
halfcheetah-medium	65.6	60.1	70.2	61.5	**81.2**
halfcheetah-medium-replay	51.0	51.5	60.2	55.3	**79.7**
halfcheetah-medium-expert	101.1	105.8	101.7	90.2	**112.2**
hopper-medium	102.0	89.3	**110.3**	99.3	92.6
hopper-medium-replay	98.1	90.2	90.3	80.2	**108.7**
hopper-medium-expert	107.0	101.3	110.2	99.3	**110.8**
walker2d-medium	82.5	85.6	79.8	79.2	**89.1**
walker2d-medium-replay	77.3	70.3	80.3	80.7	**87.9**
walker2d-medium-expert	111.6	**120.7**	110.3	110.5	109.8
antmaze-umaze-play	97.8	80.3	**100.2**	79.2	100.0
antmaze-umaze-diverse	88.3	79.2	80.5	70.2	**99.0**
antmaze-medium-play	84.0	85.3	**85.6**	80.2	81.2
antmaze-medium-diverse	76.3	76.4	79.6	70.3	**80.0**
antmaze-large-play	60.4	67.2	73.5	56.2	**79.6**
antmaze-large-diverse	54.4	55.6	80.2	54.3	**86.7**
Overall Average	83.8	81.3	87.5	77.8	**93.2**

E.5 The Necessity of Spectral Decoupling for Deep Scaling

To explicitly isolate **the effects of our proposed spectral decoupling mechanism from the generic benefits of representation pre-training**, and to evaluate its plug-and-play applicability to existing baselines, we conduct additional ablation studies. Specifically, we compare the baseline Simba V2 with variants augmented by either a generic pre-trained encoder or our Koopman spectral features on the `halfcheetah-medium-v2` dataset. As shown in Table 9, we observe two critical findings regarding the necessity and generality of our approach:

Source of Depth-Scaling Gains. To determine whether the performance improvements stem merely from representation pre-training, we compare the Koopman spectral features against a generic pre-trained encoder. The results demonstrate that simply concatenating a generic pre-trained representation (*Simba V2 + generic pre-trained encoder*) fails to improve either the final return or the effective rank. This aligns with our main ablation study (e.g., Table 2(b), Koopman $\rightarrow$ MLP), confirming that generic pre-training itself cannot unlock depth scaling. Instead, it is the explicit spectral decoupling mechanism that fundamentally prevents rank collapse and drives the scaling gains.

Applicability to Other Baselines. To investigate whether our Koopman-based decoupled representation can benefit other architectures, we integrate it into the baseline Simba V2. Table 9 provides clear evidence that augmenting the 18-layer Simba V2 with Koopman spectral features substantially improves its performance (yielding a 28.9% relative increase) and significantly increases its effective rank (from 17.83 to 101.33). This demonstrates that our decoupling mechanism is not limited to

the SEED architecture, but can effectively serve as a plug-and-play module to unlock depth-scaling benefits for other existing RL baselines.

Table 9: Ablation study on the source of depth-scaling gains in the `halfcheetah-medium-v2` environment (5-seed average). Integrating a generic pre-trained encoder provides negligible benefits, whereas our Koopman spectral decoupling mechanism significantly improves both the performance and the effective rank of 18-layer networks, even when applied to existing baselines like Simba V2.

Method	**Return (18 layers)**	**Effective Rank**
Simba V2	41.31	17.83
Simba V2 + generic pre-trained encoder	40.97	16.91
Simba V2 + Koopman spectral feature	53.27	101.33
SEED (Ours)	**61.10**	**213.19**

E.6 The Necessity of In-Sample Policy-Evaluation Decoupling for Deep Scaling

To investigate the impact of policy extraction paradigms on the scaling potential of deep value networks, we conduct a comprehensive ablation study comparing BC-regularized policy improvement and AWR.

As presented in Table 10, simply scaling the value network from 3 to 18 layers causes severe performance degradation under both paradigms due to inherent representation and target couplings. Under the BC-regularized paradigm, although SEED prevents catastrophic failure and recovers performance, the overall scores remain strictly bottlenecked. In contrast, integrating SEED under the strictly in-sample AWR paradigm completely unlocks the capacity of the deep network, achieving a dominant average score of 91.4 and securing the highest performance across all 15 tasks. This ablation empirically validates a core insight of our work: in-sample policy-evaluation decoupling is the absolute prerequisite to unleash the maximum performance of deep value networks in offline RL.

Table 10: Ablation study on the policy extraction paradigm. We compare FQL using Behavioral Cloning (BC) versus Advantage-Weighted Regression (AWR) across different value network depths (3-layer vs. 18-layer), with and without SEED. The highest score for each task is highlighted in **bold**.

Environment	**FQL (BC)**			**FQL (AWR)**		
	3-Layer	**18-Layer**	**+SEED-18**	**3-Layer**	**18-Layer**	**+SEED-18 (Ours)**
halfcheetah-medium	49.11	32.57	41.26	48.58	36.51	**61.11**
halfcheetah-medium-replay	47.21	40.16	40.33	51.31	43.29	**54.37**
halfcheetah-medium-expert	109.81	71.27	73.69	108.10	79.32	**110.23**
hopper-medium	67.21	50.91	52.87	70.26	52.11	**89.37**
hopper-medium-replay	66.12	57.32	59.32	65.33	53.36	**79.91**
hopper-medium-expert	101.51	77.43	86.52	98.39	70.91	**107.97**
walker2d-medium	96.67	80.15	85.21	90.22	81.39	**100.21**
walker2d-medium-replay	85.11	79.81	82.36	86.91	71.23	**90.37**
walker2d-medium-expert	101.29	90.77	97.34	100.99	92.11	**109.31**
antmaze-umaze-play	96.20	81.14	60.30	95.80	85.30	**100.00**
antmaze-umaze-diverse	89.00	76.55	85.00	83.20	79.31	**95.70**
antmaze-medium-play	78.30	60.31	71.80	70.60	65.30	**87.50**
antmaze-medium-diverse	71.70	60.15	50.70	80.60	61.10	**89.40**
antmaze-large-play	84.30	59.31	68.00	89.90	67.70	**95.90**
antmaze-large-diverse	83.50	60.19	91.40	85.10	79.31	**100.00**
Overall Average	81.8	65.2	69.7	81.7	67.9	**91.4**

E.7 The Insufficiency of Layer Normalization for Preventing Rank Collapse

To further investigate the underlying mechanisms of representation collapse, we evaluated the evolution of the effective rank during training across three 18-layer value network variants: the standard architecture (Vanilla: Dense + ReLU), a normalized variant (+LayerNorm) [49], and SEED.

As demonstrated in Table 11, while the addition of Layer Normalization indeed mitigates representation collapse to some extent during the early stages of training, the deep network inevitably still deteriorates into severe rank collapse as training progresses over long horizons. In contrast, SEED effectively maintains a healthy effective rank throughout the entire training process. Consequently, relying solely on Layer Normalization [49] is fundamentally insufficient to prevent rank collapse in deep value networks.

Table 11: Performance comparison across different training steps to evaluate the effect of LayerNorm and SEED on mitigating performance degradation. Values represent the normalized scores on MuJoCo tasks.

Environment	**Method**	**Steps = 5K**	**Steps = 500K**	**Steps = 1M**
Halfcheetah-medium-v2	Vanilla	5.2	1.6	1.1
	+ LayerNorm	109.2	13.2	9.8
	SEED (Ours)	**121.2**	**205.5**	**220.1**
Halfcheetah-v4	Vanilla	15.2	5.9	2.8
	+ LayerNorm	123.3	23.7	3.6
	SEED (Ours)	**211.2**	**201.7**	**231.7**

E.8 The Insufficiency of Reward Scaling for Boundary-Target Decoupling

To address the support mismatch problem in distributional RL, recent methods such as FlashSAC [50] adapt to a fixed categorical support interval by dynamically scaling the rewards and return targets based on running variance and maximum return magnitude. In this section, we demonstrate that while this reward scaling paradigm is highly effective within the online maximum entropy SAC framework, it introduces critical challenges when applied to offline RL.

Specifically, in our offline setting, applying reward scale transformations fundamentally distorts the policy extraction process. The AWR objective relies on the absolute scale of the estimated advantage function to compute regression weights $\exp(A(s,a)/\beta)$. If the underlying rewards are heavily compressed by a dynamic scaling factor C_t, the resulting advantage values are proportionally diminished, i.e., $A' \approx A/C_t$. Consequently, the exponentiated weights collapse towards a uniform constant of 1, completely stripping the ability of the algorithm to distinguish high-quality actions from sub-optimal ones within the dataset, and effectively degrading the optimization into unweighted behavior cloning.

To empirically illustrate this critical difference, we conduct an ablation study comparing fixed support, reward scaling, and our adaptive boundary approach across both online (SAC) and offline (AWR) settings. The results are summarized in Table 12.

Table 12: Performance comparison between reward scaling and our adaptive boundary across online and offline settings. Results are averaged over 5 random seeds.

Method	HalfCheetah-v4	HalfCheetah-medium-v2(Offline)
Fixed support	939.1	42.6
Reward scaling, original coefficients	13,971.6	45.1
Reward scaling, co-scaled β	-	57.7
Adaptive boundary (Ours)	**14,612.7**	**61.1**

As shown in Table 12, while reward scaling performs competitively in the online setting, it suffers severe degradation in the offline setting (HalfCheetah-medium-v2). One might attempt to recover the mathematical scale of the advantage weights by synchronously co-scaling the temperature parameter ($\bar{\beta}_t = \beta/C_t$). However, because the scaling factor C_t is highly dependent on batch statistics and exhibits significant variance, directly coupling it into the exponential denominator induces severe gradient instability. Thus, our approach explicitly avoids these optimization issues, offering superior adaptability and stability across both online and offline environments.

E.9 Computational Cost and Parameter Overhead

To provide a comprehensive evaluation of the computational overhead introduced by our framework, we quantify the costs from two perspectives: the one-time cost of the Koopman pre-training module, and the training throughput (Iterations-Per-Second, IPS) during the formal RL phase.

Pre-training Efficiency. As summarized in Table 13, the self-supervised pre-training phase is highly efficient, requiring only approximately 123 seconds of wall-clock time on a single GPU. Furthermore, the module introduces a negligible parameter overhead. During the pre-training phase, the entire autoencoder structure accounts for merely 0.16M parameters.

Training Throughput. Beyond the pre-training phase, we further evaluate the computational efficiency of SEED during RL. Table 14 reports the Iterations-Per-Second (IPS) for each baseline under identical hardware conditions, with a fixed batch size of 512 and a hidden dimension of 512.

Remarkably, despite employing a deeper architecture (18 blocks) and larger parameter capacity, SEED achieves the highest training throughput (46.70 ± 0.47 iterations/s) and lowest variance among evaluated methods. Even when compared to baselines configured with substantially fewer blocks (e.g., Simba V2 with 4 blocks), SEED operates faster. Consequently, given the significant performance gains unlocked by deep value network scaling, this minimal computational overhead demonstrates that SEED is both computationally efficient and highly practical for large-scale RL deployments.

Table 13: Computational cost and parameter overhead of the Koopman module.

Metric	**Koopman Module**
Wall-clock Time for Pre-training	$\sim$123 s
Computational Cost (FLOPs)	$\sim$153 K
Parameter Overhead (Training)	0.16 M
Parameter Overhead (Inference)	0.08 M

Table 14: Training throughput comparison. Results report the Iterations-Per-Second (IPS) evaluated under the same hardware and batch size settings. SEED achieves the highest training speed despite utilizing a much deeper 18-layer architecture.

Method	**Blocks**	**Params**	**Iterations/s (IPS)** $\uparrow$
BroNet	2	1.093M	19.07 ± 1.40
Simba V1	4	8.416M	36.42 ± 2.71
Simba V2	4	11.110M	41.74 ± 3.32
SEED (Ours)	**18**	**21.280M**	$\mathbf{46.70 \pm 0.47}$

E.10 Empirical Training Dynamics on RL Benchmarks

To comprehensively illustrate training dynamics and validate method stability, we present SEED's full learning curves across diverse online and offline RL tasks.

Online Environments. Figure 8 presents the learning curves in the active interaction setting. Evaluated on standard MuJoCo locomotion and complex AntMaze navigation tasks, SEED achieves rapid and stable convergence. Notably, it effectively prevents the performance degradation commonly observed during online deep network scaling.

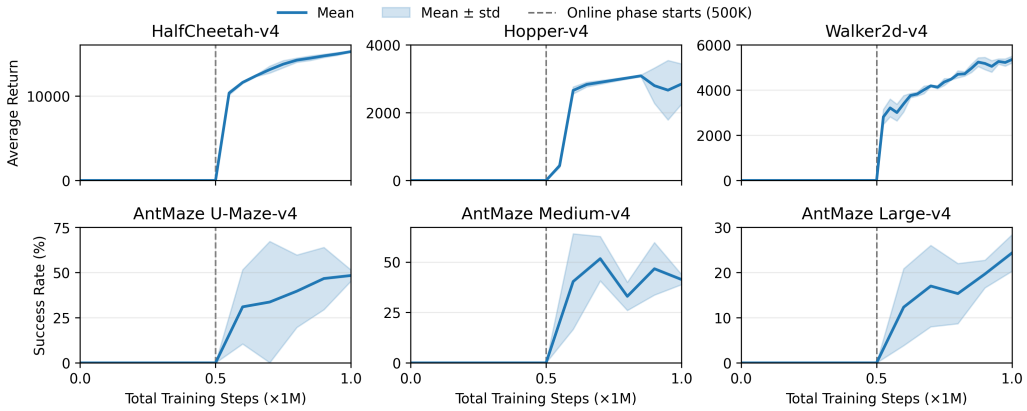

Figure 8: **Learning curves on online benchmarks.** Solid lines and shaded regions denote the mean episodic returns and standard deviations (5-seed average) on MuJoCo and AntMaze environments, respectively.

Offline Environments. Building upon the online results, Figure 9 illustrates the training dynamics on 15 offline tasks from the D4RL benchmark. Consistent with the online findings, SEED exhibits highly stable convergence across all evaluated tasks. It maintains robust performance without experiencing the catastrophic drops typical of deep value networks.

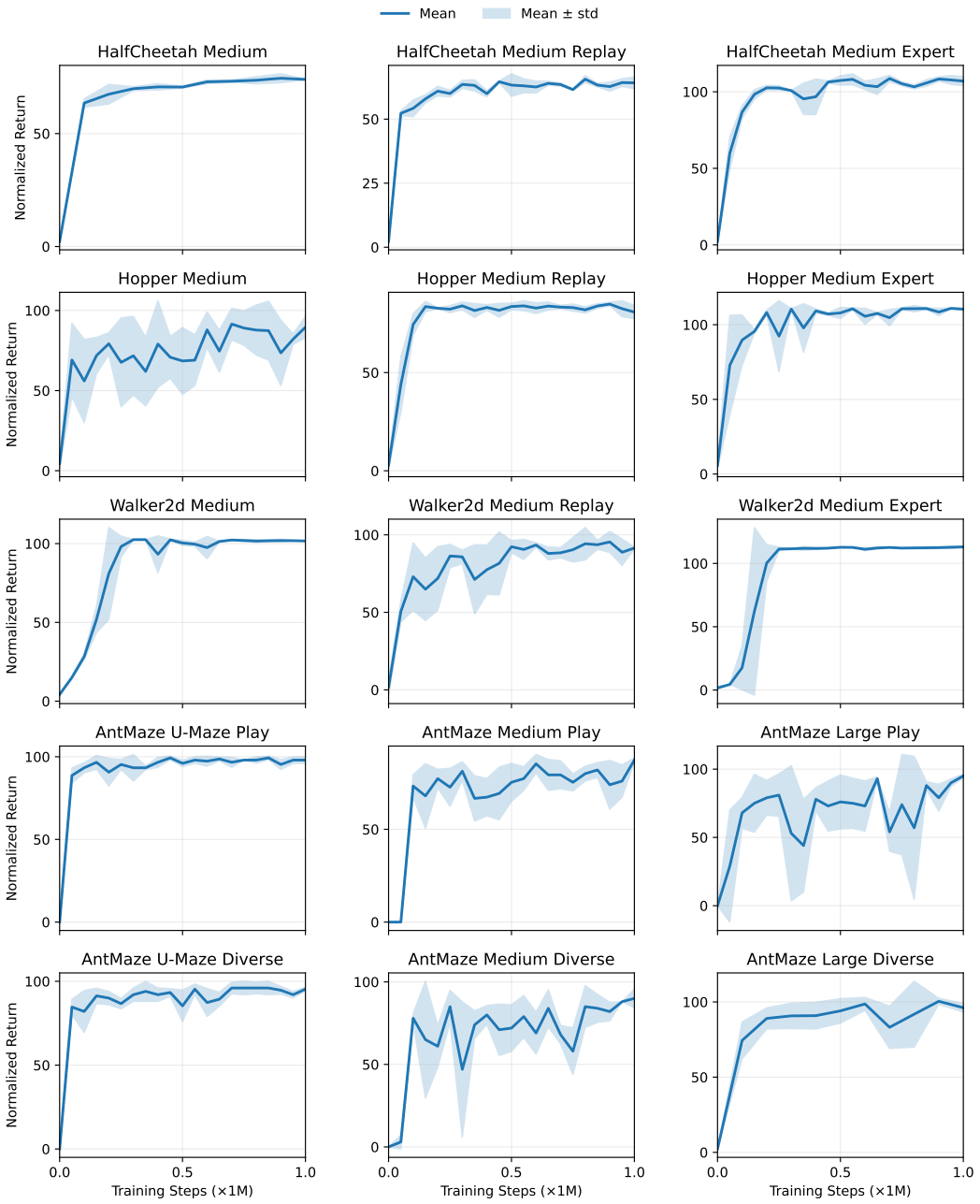

Figure 9: **Learning curves on D4RL offline benchmarks.** Solid lines and shaded regions denote the mean normalized returns and standard deviations (5-seed average), respectively.

Table 15: Hyperparameter settings for SEED training.

Hyperparameter	**Offline**	**Online**
Optimization		
Optimizer	Adam	Adam
Critic Learning Rate	6×10^{-4}	1×10^{-3}
Actor / Koopman Learning Rate	3×10^{-4}	3×10^{-4}
Batch Size	512	512
Discount Factor (γ)	0.99	0.99
Target Smoothing (τ)	0.005	0.005
Critic Network (Value)		
Architecture	ResNet (Post-activation)	ResNet (Post-activation)
Hidden Dimension	512	512
Network Depth	18 Layers	18 Layers
Activation Function	Swish	Swish
Policy Network (Actor)		
Architecture	MLP	MLP
Hidden Dimension	256	256
Activation Function	ReLU	ReLU
Output Activation	Tanh	Tanh
Koopman Module Specs		
Hidden Dimension	256	256
Pre-train Steps	30,000	–
Warmup Steps (Standard SAC)	–	5×10^5
Spectral Pre-train Epochs	–	50
Algorithm		
Distributional Bins	51	51
AWR Temperature (β)	3.0	–
Initial Temperature (α)	–	0.2
Advantage Clip Max	100.0	–
Boundary Penalty (λ)	15.0	10.0
Confidence Interval (k)	10.0	10.0

Algorithm 1 Spectrally Decoupled Distributional Learning (SEED)

Input: Environment $\mathcal{E}$ (if Online), Offline dataset $\mathcal{D}_{\text{off}}$ (if Offline), Autoencoder $\{E_\phi, D_\psi\}$, Koopman operators $\mathcal{K}$, Critic θ, Actor π, Boundary ξ.
Hyperparams: Temperature β (Offline), Entropy coef α (Online), Boundary penalty λ, Confidence k, Pre-train steps T_{pre}, RL steps T_{train}.
// Phase 1: Policy-Independent Spectral Extraction (Pre-training)
if Online **then** Collect warmup buffer $\mathcal{D}_{\text{pre}}$ using standard SAC **else** $\mathcal{D}_{\text{pre}} \leftarrow \mathcal{D}_{\text{off}}$ **end if**
for step $= 1$ to T_{pre} **do**
    Sample transitions $(s, a, s') \sim \mathcal{D}_{\text{pre}}$, and update $\phi, \psi, \mathcal{K}$ via Reconstruction & Prediction losses.
end for
Perform eigen-decomposition on the drift operator $K_0 V = V\Lambda$, and extract top-n eigenvectors V.
Define low-frequency projection: $\Phi(s) = V^\top E_\phi(s)$, and **Freeze** encoder E_ϕ and eigenvectors V.

// Phase 2: SEED Training Loop
Initialize Replay Buffer $\mathcal{R} \leftarrow \mathcal{D}_{\text{pre}}$ (Online) or $\mathcal{R} \leftarrow \mathcal{D}_{\text{off}}$ (Offline).
for iteration $t = 1$ to T_{train} **do**
    if Online Setting **then**
        Execute action $a_t \sim \pi(\cdot|s_t)$ in $\mathcal{E}$, observe (r_t, s_{t+1}), and store transition in $\mathcal{R}$.
    end if
    Sample batch $\mathcal{B} = \{(s, a, r, s')\} \sim \mathcal{R}$.
    // 1. Target Computation & Spectral Decoupling
    Extract frozen low-frequency features: $z' = \Phi(s')$.
    Sample next actions $a' \sim \pi(\cdot|s')$, compute $Q_{\text{targ}} = \min_{j=1,2} Q_{\text{targ},j}(s', a')$.
    if Online **then** $y = r + \gamma(Q_{\text{targ}} - \alpha \log \pi(a'|s'))$ **else** $y = r + \gamma Q_{\text{targ}}$ **end if**
    // 2. Adaptive Boundary Optimization
    Compute batch target statistics: $\mu_{\mathcal{B}} = \text{mean}(y), \sigma_{\mathcal{B}} = \text{std}(y)$.
    Calculate projection error $\mathcal{L}_{\text{proj}}$ mapping y onto current boundary $[-\xi, \xi]$.
    Update boundary $\xi \leftarrow \xi - \eta_\xi \nabla_\xi \big[\mathcal{L}_{\text{proj}} + \lambda \max(|\mu_{\mathcal{B}}| + k\sigma_{\mathcal{B}} - \xi, 0)\big]$.
    // 3. Distributional Critic Update (Late Fusion)
    Project target y to discrete distribution $\mathbf{p}$ using updated ξ.
    Forward Critic: $Q_\theta(s, a) = W_{\text{out}} \cdot [\text{Backbone}_\theta(s, a); \Phi(s)] + b$.
    Update Critic θ via Cross-Entropy loss with target distribution $\mathbf{p}$.
    // 4. Unified Actor Update
    if Online Setting **then**
        Update Actor π by maximizing: $\mathbb{E}_{s\sim\mathcal{B}, a\sim\pi}\big[Q_\theta(s, a) - \alpha \log \pi(a|s)\big]$.
    else
        Estimate advantage $A(s, a)$ for actions within dataset $\mathcal{B}$.
        Update Actor π by maximizing: $\mathbb{E}_{\mathcal{B}}\left[\exp\left(\frac{1}{\beta} A(s, a)\right) \log \pi(a|s)\right]$.
    end if
end for

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 claims made in the abstract and introduction accurately reflect our core contributions. The three main coupling mechanisms limiting depth scalability and the proposed SEED framework are detailed in Section 3, and the claims regarding depth scalability are validated in Section 4.

   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 spectral decoupling framework, particularly regarding its multi-stage training process and reliance on pre-training, 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: The full set of assumptions and the complete proof for the theoretical mechanisms of bootstrapped spectral coupling are provided in Appendix C.

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: The complete algorithmic details of SEED are described in Section 3, and the comprehensive experimental setups are provided in Appendix B.

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: [No]

   Justification: We commit to fully open-sourcing the code and models upon acceptance.

   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: All training and evaluation details, including hyperparameters, optimizer choices, and environment setups, are explicitly specified in Appendix B.

   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: All experimental results report the mean and standard deviation (indicated by shaded regions or error bars) across 5 independent runs with different random seeds, as detailed in Section 4.

   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: The hardware environment (e.g., GPU specifications) is described in Section 4, and a detailed analysis of the computational cost and parameter overhead for our spectral decoupling module is provided in Appendix E.9.

   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: The research strictly conforms 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: N/A. This paper focuses on foundational reinforcement learning algorithms and theoretical analysis, which do not have direct or foreseeable negative societal impacts.

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: N/A. Our proposed method does not involve releasing high-risk models or sensitive scraped 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 existing assets used in our experiments, including benchmark environments and datasets, are properly cited and credited in Section 4 and Appendix B.

    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: [N/A]

    Justification: N/A. We do not introduce or release any new datasets or assets in this 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: N/A. Our research does not involve human subjects or crowdsourcing.

    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: N/A. This paper does not involve human subjects research.

    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: N/A. LLMs were only used for minor text editing and grammar correction, not as part of the core methodology.

    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.