A Related Works

Value Divergence with Neural Network. In online reinforcement learning (RL), off-policy algorithms that employ value function approximation and bootstrapping can experience value divergence, a phenomenon known as the deadly triad [35, 4, 36, 37, 11]. Deep Q-Networks (DQN) typify this issue. As they employ neural networks for function approximation, they are particularly susceptible to Q-value divergence [18, 13, 37]. Past research has sought to empirically address this divergence problem through various methods, such as the use of separate target networks [29] and Double-Q learning [18, 13]. Achiam *et al.* [1] analyze a linear approximation of Q-network to characterizes the diverge, while CrossNorm [6] uses a elaborated version of BatchNorm [19] to achieve stable learning. Value divergence becomes even more salient in offline RL, where the algorithm learns purely from a fixed dataset without additional environment interaction [14, 24]. Much of the focus in the field of offline RL has been on controlling the extent of off-policy learning, *i.e.*, policy constraint [12, 30, 23, 14, 40, 38, 8, 32]. Several previous studies [31, 5] have empirically utilized LayerNorm to enhance performance in online and offline RL. These empirical results partially align with the experimental section of our work. However, our study makes a theoretical explanation for how LayerNorm mitigates divergence through the NTK analysis. Specifically, we empirically and theoretically illustrate how LayerNorm reduces SEEM. In addition to LayerNorm, our contribution extends to explaining divergence and proposing promising solutions from the perspective of reducing SEEM. Specially, we discover that WeightNorm can also be an effective tool and explain why other regularization techniques fall short. Finally, we perform comprehensive experiments to empirically verify the effectiveness of LayerNorm on the %X dataset, a practical setting not explored in previous work. Thus, our contributions are multifaceted and extend beyond the mere application of LayerNorm.

Offline RL. Offline RL presents significant challenges due to severe off-policy issues and extrapolation errors. Some existing methods focuses on designs explicit or implicit policy regularizations to minimize the discrepancy between the learned and behavior policies. For example, TD3+BC [13, 12] directly adds a behavior cloning loss to mimic the behavior policy, Diffusion-QL [38] further replace the BC loss with a diffusion loss and using diffusion models as the policy. CRR [39] and AWR [32] impose an implicit policy regularization by performing policy gradient-style policy updates. Meanwhile, some other works try to alleviate the extrapolation errors by modifying the policy evaluation procedure. Specifically, CQL [25] penalizes out-of-distribution actions for having higher Q-values, while IQL [23] and OneStep RL [7] only uses in-distribution data for policy evaluation, thus avoiding querying unseen actions. Alternatively, decision transformer (DT) [9] and trajectory transformer [21] cast offline RL as a sequence generation problem, which are beyond the scope of this paper. Despite the effectiveness of the above methods, they usually neglect the effect of function approximator and are thus sensitive to hyperparameters for trading off performance and training stability. Exploration into the function approximation aspect of the deadly triad is lacking in offline RL. Moreover, a theoretical analysis of divergence in offline RL that does not consider the function approximator would be inherently incomplete. We instead focus on this orthogonal perspective and provide both theoretical understanding and empirical solution to the offline RL problem.

B Proof of Main Theorems

Before proving our main theorem, we first state an important lemma.

Lemma 1. *For any L-layer ReLU-activate MLP and any fixed input $\boldsymbol{x}, \boldsymbol{x}'$. If we scale up every parameter of $f_{\boldsymbol{\theta}}$ to λ times, namely $\boldsymbol{\theta}' = \lambda\boldsymbol{\theta}$ where λ is a large number such that the bias term is negligible, then we have following equations hold*

$$\begin{aligned} f_{\boldsymbol{\theta}'}(\boldsymbol{x}) =& \lambda^L f_{\boldsymbol{\theta}}(\boldsymbol{x}), \\ \nabla_{\boldsymbol{\theta}} f_{\boldsymbol{\theta}'}(\boldsymbol{x}) \approx& \lambda^{L-1} \nabla_{\boldsymbol{\theta}} f_{\boldsymbol{\theta}}(\boldsymbol{x}), \\ \boldsymbol{G}_{\boldsymbol{\theta}'}(\boldsymbol{x}, \boldsymbol{x}') \approx& \lambda^{2(L-1)} \boldsymbol{G}_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{x}'). \end{aligned}$$

Proof. Recursively define

$$\boldsymbol{z}_{\ell+1} = \boldsymbol{W}_\ell \tilde{\boldsymbol{z}}_\ell + \boldsymbol{b}_\ell, \quad \boldsymbol{z}_0 = \boldsymbol{x}$$

$$\tilde{\boldsymbol{z}}_\ell = \sigma(\boldsymbol{z}_\ell).$$

Then it is easy to see that if we multiply each $\boldsymbol{W}_\ell$ and $\boldsymbol{b}_\ell$ by λ, denote the new corresponding value to be $\boldsymbol{z}'_\ell$ we have

$$\begin{aligned} \boldsymbol{z}'_1 =& \lambda \boldsymbol{z}_1 \\ \boldsymbol{z}'_2 =& \lambda^2 \boldsymbol{z}_2 \\ & \cdots \\ \boldsymbol{z}'_L =& \lambda^L \boldsymbol{z}_L \end{aligned}$$

Hence we know $f_{\boldsymbol{\theta}'}(\boldsymbol{x}) = \lambda^L f_{\boldsymbol{\theta}}(\boldsymbol{x})$.

Taking gradient backwards, we know that $\left\|\frac{\partial f}{\partial \boldsymbol{W}_\ell}\right\|$ is proportional to both $\|\tilde{\boldsymbol{z}}_\ell\|$ and $\left\|\frac{\partial f}{\partial \boldsymbol{z}_{\ell+1}}\right\|$. Therefore we know

$$\frac{\partial f_{\boldsymbol{\theta}'}}{\partial \boldsymbol{W}'_\ell} = \tilde{\boldsymbol{z}}'_\ell \frac{\partial f_{\boldsymbol{\theta}'}}{\partial \boldsymbol{z}'_{\ell+1}} = \lambda^\ell \tilde{\boldsymbol{z}}_\ell \cdot \lambda^{L-\ell-1} \frac{\partial f_{\boldsymbol{\theta}}}{\partial \boldsymbol{z}_{\ell+1}} = \lambda^{L-1} \frac{\partial f_{\boldsymbol{\theta}}}{\partial \boldsymbol{W}_\ell}.$$

This suggests that all gradients with respect to the weights become scaled by a factor of λ^{L-1}. The gradients with respect to the biases are proportional to λ^{L-l}. When λ is large enough to render the gradient of the bias term negligible, it follows that $\nabla_{\boldsymbol{\theta}} f_{\boldsymbol{\theta}'}(\boldsymbol{x}) \approx \lambda^{L-1} \nabla_{\boldsymbol{\theta}} f_{\boldsymbol{\theta}}(\boldsymbol{x})$. This equation implies that the gradient updates for the model parameters are dominated by the weights, with negligible contribution from the bias terms. And since NTK is the inner product between gradients, we know $\boldsymbol{G}_{\boldsymbol{\theta}'}(\boldsymbol{x}, \boldsymbol{x}') \approx \lambda^{2(L-1)} \boldsymbol{G}_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{x}')$. $\square$

Theorem 1 *Suppose that the network's parameter at iteration t is $\boldsymbol{\theta}_t$. For each transition (s_i, a_i, s_{i+1}, r_i) in dataset, denote $\boldsymbol{r} = [r_1, \ldots, r_M]^\top \in \mathbb{R}^M$, $\hat{\pi}_{\boldsymbol{\theta}_t}(s) = \arg\max_a \hat{Q}_{\boldsymbol{\theta}_t}(s, a)$. Denote $\boldsymbol{x}^*_{i,t} = (s_{i+1}, \hat{\pi}_{\boldsymbol{\theta}_t}(s_{i+1}))$. Concatenate all $\boldsymbol{x}^*_{i,t}$ to be $\boldsymbol{X}^*_t$. Denote $\boldsymbol{u}_t = f_{\boldsymbol{\theta}_t}(\boldsymbol{X}) - (\boldsymbol{r} + \gamma \cdot f_{\boldsymbol{\theta}_t}(\boldsymbol{X}^*_t))$ to be TD error vector at iteration t. The learning rate η is infinitesimal. We have the following evolving equation for $\boldsymbol{u}_{t+1}$*

$$\boldsymbol{u}_{t+1} = (\boldsymbol{I} + \eta \boldsymbol{A}_t) \boldsymbol{u}_t. \tag{2}$$

where $\boldsymbol{A} = \left(\gamma \phi_{\boldsymbol{\theta}_t}(\boldsymbol{X}^*_t) - \phi_{\boldsymbol{\theta}_t}(\boldsymbol{X})\right)^\top \phi_{\boldsymbol{\theta}_t}(\boldsymbol{X}) = \gamma \boldsymbol{G}_{\boldsymbol{\theta}_t}(\boldsymbol{X}^*_t, \boldsymbol{X}) - \boldsymbol{G}_{\boldsymbol{\theta}_t}(\boldsymbol{X}, \boldsymbol{X})$.

Proof. For the sake of simplicity, denote $\boldsymbol{Z}_t = \nabla_{\boldsymbol{\theta}} f_{\boldsymbol{\theta}}(\boldsymbol{X})\Big|_{\boldsymbol{\theta}_t}$, $\boldsymbol{Z}^*_t = \nabla_{\boldsymbol{\theta}} f_{\boldsymbol{\theta}}(\boldsymbol{X}^*_t)\Big|_{\boldsymbol{\theta}_t}$. The Q-value iteration minimizes loss function $\mathcal{L}$ defined by $\mathcal{L}(\boldsymbol{\theta}) = \frac{1}{2}\left\| f_{\boldsymbol{\theta}}(\boldsymbol{X}) - (\boldsymbol{r} + \gamma \cdot f_{\boldsymbol{\theta}_t}(\boldsymbol{X}^*_t)) \right\|_2^2$. Therefore we have the gradient as

$$\frac{\partial \mathcal{L}(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} = \boldsymbol{Z}_t \left(f_{\boldsymbol{\theta}}(\boldsymbol{X}) - (\boldsymbol{r} + \gamma \cdot f_{\boldsymbol{\theta}_t}(\boldsymbol{X}^*_t)) \right) = \boldsymbol{Z}_t \boldsymbol{u}_t. \tag{3}$$

According to gradient descent, we know $\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \eta \boldsymbol{Z}_t \boldsymbol{u}_t$. Since η is very small, we know $\boldsymbol{\theta}_{t+1}$ stays within the neighborhood of $\boldsymbol{\theta}_t$. We can Taylor-expand function $f_{\boldsymbol{\theta}}(\cdot)$ near $\boldsymbol{\theta}_t$ as

$$f_{\boldsymbol{\theta}}(\boldsymbol{X}) \approx \nabla^\top_{\boldsymbol{\theta}} f_{\boldsymbol{\theta}}(\boldsymbol{X})\Big|_{\boldsymbol{\theta}_t} (\boldsymbol{\theta} - \boldsymbol{\theta}_t) + f_{\boldsymbol{\theta}_t}(\boldsymbol{X}) = \boldsymbol{Z}_t^\top (\boldsymbol{\theta} - \boldsymbol{\theta}_t) + f_{\boldsymbol{\theta}_t}(\boldsymbol{X}). \tag{4}$$

$$f_{\boldsymbol{\theta}}(\boldsymbol{X}^*_t) \approx (\boldsymbol{Z}^*_t)^\top (\boldsymbol{\theta} - \boldsymbol{\theta}_t) + f_{\boldsymbol{\theta}_t}(\boldsymbol{X}^*_t). \tag{5}$$

When η is infinitesimally small, the equation holds. Plug in $\boldsymbol{\theta}_{t+1}$, we know

$$f_{\boldsymbol{\theta}_{t+1}}(\boldsymbol{X}) - f_{\boldsymbol{\theta}_t}(\boldsymbol{X}) = -\eta \boldsymbol{Z}_t^\top \boldsymbol{Z}_t \boldsymbol{u}_t = -\eta \cdot \boldsymbol{G}_{\boldsymbol{\theta}_t}(\boldsymbol{X}, \boldsymbol{X}). \tag{6}$$

$$f_{\boldsymbol{\theta}_{t+1}}(\boldsymbol{X}^*_t) - f_{\boldsymbol{\theta}_t}(\boldsymbol{X}^*_t) = -\eta (\boldsymbol{Z}^*_t)^\top \boldsymbol{Z}_t \boldsymbol{u}_t = -\eta \cdot \boldsymbol{G}_{\boldsymbol{\theta}_t}(\boldsymbol{X}^*_t, \boldsymbol{X}). \tag{7}$$

Since the change of $\boldsymbol{\theta}$ is small, we know $\boldsymbol{X}^*_{t+1} \approx \boldsymbol{X}^*_t$. So $\boldsymbol{u}_{t+1}$ boils down to

$$\begin{aligned} \boldsymbol{u}_{t+1} &= f_{\boldsymbol{\theta}_{t+1}}(\boldsymbol{X}) - \boldsymbol{r} - \gamma f_{\boldsymbol{\theta}_{t+1}}(\boldsymbol{X}^*_{t+1}) && (8) \\ &= f_{\boldsymbol{\theta}_t}(\boldsymbol{X}) - \eta \cdot \boldsymbol{G}_{\boldsymbol{\theta}_t}(\boldsymbol{X}, \boldsymbol{X}) \boldsymbol{u}_t - \boldsymbol{r} - \gamma (f_{\boldsymbol{\theta}_t}(\boldsymbol{X}^*_t) - \eta \cdot \boldsymbol{G}_{\boldsymbol{\theta}_t}(\boldsymbol{X}^*_t, \boldsymbol{X})) \boldsymbol{u}_t && (9) \\ &= \underbrace{f_{\boldsymbol{\theta}_t}(\boldsymbol{X}) - \boldsymbol{r} - \gamma f_{\boldsymbol{\theta}_t}(\boldsymbol{X}^*_t)}_{\boldsymbol{u}_t} + \eta \cdot (\gamma \boldsymbol{G}_{\boldsymbol{\theta}_t}(\boldsymbol{X}^*_t, \boldsymbol{X}) - \boldsymbol{G}_{\boldsymbol{\theta}_t}(\boldsymbol{X}, \boldsymbol{X})) \boldsymbol{u}_t && (10) \\ &= (\boldsymbol{I} + \eta \boldsymbol{A}_t) \boldsymbol{u}_t. && (11) \end{aligned}$$

where $\boldsymbol{A} = \gamma \boldsymbol{G}_{\boldsymbol{\theta}_t}(\boldsymbol{X}^*_t, \boldsymbol{X}) - \boldsymbol{G}_{\boldsymbol{\theta}_t}(\boldsymbol{X}, \boldsymbol{X})$. $\square$

Theorem 3 *Given iteration $t > t_0$, and $\boldsymbol{A} = \gamma \boldsymbol{G}_{\boldsymbol{\theta}_{t_0}}(\boldsymbol{X}^*_{t_0}, \boldsymbol{X}) - \boldsymbol{G}_{\boldsymbol{\theta}_{t_0}}(\boldsymbol{X}, \boldsymbol{X})$. The divergence of $\boldsymbol{u}_t$ is equivalent to whether there exists an eigenvalue λ of $\boldsymbol{A}$ such that $\mathrm{Re}(\lambda) > 0$. If converge, we have $\boldsymbol{u}_t = (\boldsymbol{I} + \eta \boldsymbol{A})^{t-t_0} \cdot \boldsymbol{u}_{t_0}$. Otherwise, $\boldsymbol{u}_t$ becomes parallel to the eigenvector of the largest eigenvalue λ of $\boldsymbol{A}$, and its norm diverges to infinity at following order.*

$$\|\boldsymbol{u}_t\|_2 = O\left(\frac{1}{(1 - C'\lambda\eta t)^{L/(2L-2)}}\right). \tag{12}$$

for some constant C' to be determined and L is the number of layers of MLP. Specially, when $L = 2$, it reduces to $O\left(\frac{1}{1 - C'\lambda\eta t}\right)$.

Proof. According to Assumption 2, max action becomes stable after t_0. It implies $\boldsymbol{X}^*_t = \boldsymbol{X}^*_{t_0} := \boldsymbol{X}^*$. The stability of the NTK direction implies that for some scalar k_t and the specific input $\boldsymbol{X}^*, \boldsymbol{X}$, we have $\boldsymbol{G}_{\boldsymbol{\theta}_t}(\boldsymbol{X}^*, \boldsymbol{X}) = k_t \boldsymbol{G}_{\boldsymbol{\theta}_{t_0}}(\boldsymbol{X}^*, \boldsymbol{X})$ and $\boldsymbol{G}_{\boldsymbol{\theta}_t}(\boldsymbol{X}, \boldsymbol{X}) = k_t \boldsymbol{G}_{\boldsymbol{\theta}_{t_0}}(\boldsymbol{X}, \boldsymbol{X})$. Further, we have $\boldsymbol{A}_t = k_t \boldsymbol{A}$. It equals 1 if the training is convergent, but will float up if the model's predicted Q-value blows up.

we know all the eigenvalues of $\boldsymbol{I} + \eta \boldsymbol{A}$ have form $1 + \eta\lambda_i$. Considering η is small enough, we have $|1 + \eta\lambda_i|^2 \approx 1 + 2\eta\,\mathrm{Re}(\lambda)$. Now suppose if there does not exists eigenvalue λ of $\boldsymbol{A}$ satisfies $\mathrm{Re}(\lambda) > 0$, we have $|1 + \eta\lambda_i| \leq 1$. Therefore, the NTK will become perfectly stable so $k_t = 1$ for $t > t_0$, and we have

$$\boldsymbol{u}_t = (\boldsymbol{I} + \eta \boldsymbol{A}_{t-1})\boldsymbol{u}_{t-1} = (\boldsymbol{I} + \eta \boldsymbol{A}_{t-1})(\boldsymbol{I} + \eta \boldsymbol{A}_{t-2})\boldsymbol{u}_{t-2} = \ldots = \prod_{s=t_0}^{t-1} (\boldsymbol{I} + \eta \boldsymbol{A}_s)\boldsymbol{u}_{t_0} \tag{13}$$

$$= \prod_{s=t_0}^{t-1} (\boldsymbol{I} + \eta \boldsymbol{A})\boldsymbol{u}_{t_0} = (\boldsymbol{I} + \eta \boldsymbol{A})^{t-t_0} \cdot \boldsymbol{u}_{t_0}. \tag{14}$$

Otherwise, there exists an eigenvalue for $\boldsymbol{A}$ satisfying $\mathrm{Re}(\lambda) > 0$. Denote the one with the largest real part as λ, and $\boldsymbol{v}$ to be the corresponding eigenvector. We know matrix $\boldsymbol{I} + \eta \boldsymbol{A}$ also has left eigenvector $\boldsymbol{v}$, whose eigenvalue is $1 + \eta\lambda$. In this situation, we know after each iteration, $\|\boldsymbol{u}_{t+1}\|$ will become larger than $\|\boldsymbol{u}_t\|$. Moreover, to achieve larger and larger prediction values, the model's parameter's norm $\|\boldsymbol{\theta}_t\|$ also starts to explode. We know $\boldsymbol{u}_t$ is homogeneous with respect $\boldsymbol{\theta}_t$ for ReLU networks. The output $f_{\boldsymbol{\theta}_t}(\boldsymbol{X})$ enlarges p^L times when $\boldsymbol{\theta}_t$ enlarges p times. When the reward values is small with respect to the divergent Q-value, TD error $\boldsymbol{u}_t = O(f_{\boldsymbol{\theta}_t}(\boldsymbol{X})) = O(\boldsymbol{\theta}_t^L)$. Besides, according to lemma1, we know $k_t = O(\|\boldsymbol{\theta}_t\|^{2(L-1)}) = O(\|\boldsymbol{u}_t\|^{2(L-1)/L}) = O(\|\boldsymbol{u}_t\|^{2-2/L})$.

Denote $g(\eta t) = \boldsymbol{v}^\top \boldsymbol{u}_t$, left multiply $\boldsymbol{v}$ to equation $\boldsymbol{u}_{t+1} = (\boldsymbol{I} + \eta k_t \boldsymbol{A})\boldsymbol{u}_t$. we have $g(\eta t + \eta) = (1 + \eta\lambda k_t)g(\eta t)$. Since we know such iteration will let $\boldsymbol{u}_t$ to be dominated by $\boldsymbol{v}$ and align with $\boldsymbol{v}$, we know $g(\eta t) = O(\|\boldsymbol{u}_t\|)$ for large t. Therefore $k_t = O(\|\boldsymbol{u}_t\|^{2(L-1)/L}) = C \cdot g(\eta t)^{2-2/L}$. This boils down to $g(\eta t + \eta) = g(\eta t) + C\eta\lambda g(\eta t)^2$, which further becomes

$$\frac{g(\eta t + \eta) - g(\eta t)}{\eta} = C\lambda g(\eta t)^{3-2/L} \tag{15}$$

Let $\eta \to 0$, we have an differential equation $\frac{\mathrm{d}g}{\mathrm{d}t} = C\lambda g(t)^{3-2/L}$. When $L = 1$, the MLP network degenerates to a linear function. The solution of ODE is

$$\|\boldsymbol{u}_t\| = g(\eta t) = C' e^{\lambda t}, \tag{16}$$

reflecting the exponential growth under linear function that has been studied in previous works [36]. When $L > 2$, Solving this ODE gives

$$g(t) = \frac{1}{(1 - C'\lambda t)^{L/(2L-2)}}. \tag{17}$$

So at an infinite limit, we know $\|\boldsymbol{u}_t\| = g(\eta t) = O\left(\frac{1}{(1 - C'\lambda\eta t)^{L/(2L-2)}}\right)$. Specially, for the experimental case we study in Figure 3 where $L = 2$, it reduces to $O\left(\frac{1}{1 - C'\lambda\eta t}\right)$. We conduct more experiments with $L = 3$ in Appendix C.2 to verify our theoretical findings. $\square$

C More Observations and Deduction

C.1 Model Alignment

In addition to the findings presented in Theorem 1 and Theorem 3, we have noticed several intriguing phenomena. Notably, beyond the critical point, gradients tend to align along a particular direction, leading to an infinite growth of the model's parameters in that same direction. This phenomenon is supported by the observations presented in Figure 12, Figure 13, and Figure 14, where the cosine similarity between the current model parameters and the ones at the ending of training remains close to 1 after reaching a critical point, even as the norm of the parameters continually increases.

C.2 Terminal Time

Theorem 3 claims $\|\boldsymbol{u}_t\| = O(f_{\boldsymbol{\theta}_t}(\boldsymbol{X})) = O\left(\frac{1}{(1-C'\lambda\eta t)^{L/(2L-2)}}\right)$, implying the relation

$$1/q^{(2L-2)/L} \propto 1 - C'\lambda\eta t. \tag{18}$$

Beisdes, it implies the existence of a "terminal time" $\frac{1}{C'\eta\lambda}$ that the model must crash at a singular point. When the training approaches this singular point, the estimation value and the model's norm explode rapidly in very few steps. We have run an experiment with $L = 2$ in Figure 3, from which we can see that Q-value's inverse proves to decay linearly and eventually becomes Nan at the designated time step. When $L = 3$, from our theretical analysis, we have $1/q^{\frac{4}{3}} \propto 1 - C'\lambda\eta t$. The experimental results in Figure 10 corroborate this theoretical prediction, where the inverse Q-value raised to the power of $4/3$ is proportional to $1 - C'\lambda\eta t$ after a critical point and it eventually reaches a NAN value at the terminal time step.

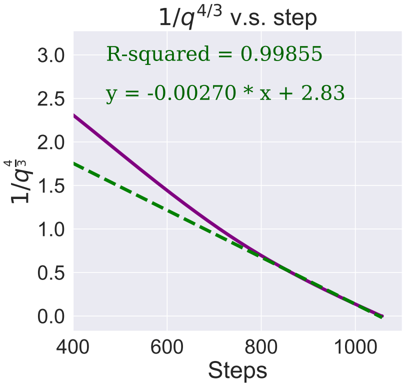


Figure 10: Linear decay with SGD and L=3.

C.3 Adam Case

In this section, we will prove that if the algorithm employs Adam as the optimizer, the model still suffers divergence. Moreover, we demonstrate that the norm of the network increase linearly, of which the slope is $\eta\sqrt{P}$, where P is the number of parameters and η is the learning rate. Also, the Q-value prediction will increase at L_{th}-polynomial's rate, where L is the number of layers of model $f_{\boldsymbol{\theta}}$. Experimental results in Figure 4 verified our findings. Besides, we show that all runnings across D4RL environments represents the linear growth of the norm of the Q-network in Figure 12, Figure 13, and Figure 14.

Theorem 4. *Suppose we use Adam optimizer for Q-value iteration and all other settings are the same as Theorem 3. After $t > t_0$, the model will diverge if and only if $\lambda_{\max}(\boldsymbol{A}) > 0$. If it diverges, we have $\|\boldsymbol{\theta}_t\| = \eta\sqrt{P}t + o(t)$ and $\|\boldsymbol{u}_t\| = \Theta(t^L)$ where P and L are the number of parameters and the number of layers for network $f_{\boldsymbol{\theta}}$, respectively.*

Proof. We only focus on the asymptotic behavior of Adam. So we only care about the dynamics for $t > T$ for some large T. Also, at this regime, we know that the gradient has greatly aligned with the model parameters. So we assume that

$$\nabla L(\theta_t) = -C \cdot \theta_t. \quad C > 0 \tag{19}$$

Recall that each iteration of the Adam algorithm has the following steps.

$$g_t = \nabla L(\theta_t), \tag{20}$$
$$m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t, \tag{21}$$
$$v_t = \beta_2 v_{t-1} + (1-\beta_2) g_t^2, \tag{22}$$
$$\hat{m}_t = \frac{m_t}{1-\beta_1^t}, \tag{23}$$
$$\hat{v}_t = \frac{v_t}{1-\beta_2^t}, \tag{24}$$
$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\hat{v}_t} + \epsilon} \hat{m}_t. \tag{25}$$

Instead of exactly solving this series, we can verify linear growth is indeed the terminal behavior of θ_t since we only care about asymptotic order. Assume that $\theta_t = kt$ for $t > T$, we can calculate m_t by dividing both sides of the definition of m_t by β_1^t, which gives

$$\frac{m_t}{\beta_1^t} = \frac{m_{t-1}}{\beta_1^{t-1}} + \frac{1-\beta_1}{\beta_1^t} g_t. \tag{26}$$

$$\frac{m_t}{\beta_1^t} = \sum_{s=0}^{t} \frac{1-\beta_1}{\beta_1^s} g_s. \tag{27}$$

$$m_t = -C \sum_{s=0}^{t} (1-\beta_1) \beta_1^{t-s} ks = -kCt + o(t) \tag{28}$$

, where g_t is given in Equation (19). Similarly, we have

$$v_t = kC^2 t^2 + o(t^2) \tag{29}$$

Hence we verify that

$$\theta_{t+1} - \theta_t = -\eta \cdot \frac{m_t}{1-\beta_1^t} \cdot \sqrt{\frac{1-\beta_2^t}{v_t}} \to \eta \cdot \frac{kCt}{\sqrt{k^2 C^2 t^2}} = \eta$$

therefore we know each iteration will increase each parameter by exactly constant η. This in turn verified our assumption that parameter θ_t grows linearly. The slope for the overall parameter is thus $\eta\sqrt{P}$. This can also be verified in Figure 4. When we have $\boldsymbol{\theta}_t = \eta\sqrt{P}\bar{\boldsymbol{\theta}}$, where $\bar{\boldsymbol{\theta}}$ is the normalized parameter, we can further deduce the increasing order of the model's estimation. According to lemma 1, the Q-value estimation (also the training error) increase at speed $O(t^L)$. $\square$

D LayerNorm's Effect on NTK

In this section, we demonstrate the effect of LayerNorm on SEEM. Our demonstration is just an intuitive explanation rather than a rigorous proof. We show that adding a LayerNorm can effectively reduce the NTK between any $\boldsymbol{x}_0$ and extreme input $\boldsymbol{x}$ down from linear to constant. Since each entry of Gram matrix $\boldsymbol{G}$ is an individual NTK value, we can informally expect that $\boldsymbol{G}(\boldsymbol{X}_t^*, \boldsymbol{X})$'s eigenvalue are greatly reduced when every individual NTK value between any $\boldsymbol{x}_0$ and extreme input $\boldsymbol{x}$ is reduced.

We consider a two-layer MLP. The input is $\boldsymbol{x} \in \mathbb{R}^{d_{in}}$, and the hidden dimension is d. The parameters include $\boldsymbol{W} = [\boldsymbol{w}_1, \ldots, \boldsymbol{w}_d]^\top \in \mathbb{R}^{d \times d_{in}}, \boldsymbol{b} \in \mathbb{R}^d$ and $\boldsymbol{a} \in \mathbb{R}^d$. Since for the NTK value, the last layer's bias term has a constant gradient, we do not need to consider it. The forward function of the network is

$$f_{\boldsymbol{\theta}}(\boldsymbol{x}) = \sum_{i=1}^{d} a_i \sigma(\boldsymbol{w}_i^\top \boldsymbol{x} + b_i).$$

Proposition 1. *For any input $\boldsymbol{x}$ and network parameter $\boldsymbol{\theta}$, if $\nabla_{\boldsymbol{\theta}} f_{\boldsymbol{\theta}}(\boldsymbol{x}) \neq \mathbf{0}$, then we have*

$$\lim_{\lambda \to \infty} k_{\mathrm{NTK}}(\boldsymbol{x}, \lambda \boldsymbol{x}) = \Omega(\lambda) \to \infty. \tag{30}$$

Proof. Denote $z_i = \boldsymbol{w}_i^\top \boldsymbol{x} + b_i$, according to condition $\nabla_{\boldsymbol{\theta}} f_{\boldsymbol{\theta}}(\boldsymbol{x}) \neq \mathbf{0}$, we know there must exist at least one i such that $z_i > 0$, denote this set as P. Now consider all the $i \in [d]$ that satisfy $z_i > 0$ and $\boldsymbol{w}_i^\top \boldsymbol{x} > 0$ (otherwise take opposite sign of λ), we have

$$\frac{\partial f}{\partial a_i}\Big|_{\boldsymbol{x}} = \sigma(\boldsymbol{w}_i^\top \boldsymbol{x} + b_i) = \boldsymbol{w}_i^\top \boldsymbol{x} + b_i, \tag{31}$$

$$\frac{\partial f}{\partial \boldsymbol{w}_i}\Big|_{\boldsymbol{x}} = a_i \boldsymbol{x}, \tag{32}$$

$$\frac{\partial f}{\partial b_i}\Big|_{\boldsymbol{x}} = a_i. \tag{33}$$

Similarly, we have

$$\frac{\partial f}{\partial a_i}\Big|_{\lambda\boldsymbol{x}} = \sigma(\lambda\boldsymbol{w}_i^\top \boldsymbol{x} + b_i) = \lambda\boldsymbol{w}_i^\top \boldsymbol{x} + b_i, \tag{34}$$

$$\frac{\partial f}{\partial \boldsymbol{w}_i}\Big|_{\lambda\boldsymbol{x}} = \lambda a_i \boldsymbol{x}, \tag{35}$$

$$\frac{\partial f}{\partial b_i}\Big|_{\lambda\boldsymbol{x}} = a_i. \tag{36}$$

So we have

$$\sum_{i \in P} \left\langle \frac{\partial f(\boldsymbol{x})}{\partial \boldsymbol{\theta}_i}, \frac{\partial f(\lambda\boldsymbol{x})}{\partial \boldsymbol{\theta}_i} \right\rangle = \lambda \left((\boldsymbol{w}_i^\top \boldsymbol{x})^2 + b_i \boldsymbol{w}_i^\top \boldsymbol{x} + a_i^2 \|\boldsymbol{x}\|^2 \right) + O(1) = \Theta(\lambda).$$

Denote $N = \{1, \ldots, d\} \setminus P$. We know for every $j \in N$ either $\frac{\partial f(\boldsymbol{x})}{\partial a_j} = \frac{\partial f(\boldsymbol{x})}{\partial b_j} = \frac{\partial f(\boldsymbol{x})}{\partial \boldsymbol{w}_j} = 0$, or $\boldsymbol{w}_j^\top \boldsymbol{x} < 0$. For the latter case, we know $\lim_{\lambda \to \infty} \frac{\partial f(\lambda\boldsymbol{x})}{\partial a_j} = \frac{\partial f(\lambda\boldsymbol{x})}{\partial b_j} = \frac{\partial f(\lambda\boldsymbol{x})}{\partial \boldsymbol{w}_j} = 0$. In both cases, we have

$$\lim_{\lambda \to \infty} \sum_{j \in N} \left\langle \frac{\partial f(\boldsymbol{x})}{\partial \boldsymbol{\theta}_j}, \frac{\partial f(\lambda\boldsymbol{x})}{\partial \boldsymbol{\theta}_j} \right\rangle = 0.$$

Therefore, according to the definition of NTK, we have

$$\lim_{\lambda \to \infty} k_{\mathrm{NTK}}(\boldsymbol{x}, \lambda\boldsymbol{x}) = \lim_{\lambda \to \infty} \left\langle \frac{\partial f(\boldsymbol{x})}{\partial \boldsymbol{\theta}_i}, \frac{\partial f(\lambda\boldsymbol{x})}{\partial \boldsymbol{\theta}_i} \right\rangle = \Theta(\lambda) \to \infty.$$

□

For the model equipped with LayerNorm, the forward function becomes

$$f_{\boldsymbol{\theta}}(\boldsymbol{x}) = \boldsymbol{a}^\top \sigma(\psi(\boldsymbol{W}\boldsymbol{x} + \boldsymbol{b})),$$

where $\psi(\cdot)$ is the layer normalization function defined as

$$\psi(\boldsymbol{x}) = \sqrt{d} \cdot \frac{\boldsymbol{x} - \mathbf{1}\mathbf{1}^\top \boldsymbol{x}/d}{\left\| \boldsymbol{x} - \mathbf{1}\mathbf{1}^\top \boldsymbol{x}/d \right\|}.$$

Denote $\boldsymbol{P} = \boldsymbol{I} - \mathbf{1}\mathbf{1}^\top/d$, note that the derivative of $\psi(\cdot)$ is

$$\dot{\psi}(\boldsymbol{x}) = \frac{\partial \psi(\boldsymbol{x})}{\partial \boldsymbol{x}} = \sqrt{d} \cdot \left(\frac{\boldsymbol{I}}{\|\boldsymbol{P}\boldsymbol{x}\|} - \frac{\boldsymbol{P}\boldsymbol{x}\boldsymbol{x}^\top \boldsymbol{P}}{\|\boldsymbol{P}\boldsymbol{x}\|^3} \right) \boldsymbol{P}. \tag{37}$$

Specially, we have

$$\psi(\lambda\boldsymbol{x}) = \sqrt{d} \cdot \frac{\lambda\boldsymbol{x} - \lambda\mathbf{1}\mathbf{1}^\top \boldsymbol{x}/d}{\lambda \left\| \boldsymbol{x} - \mathbf{1}\mathbf{1}^\top \boldsymbol{x}/d \right\|} = \psi(\boldsymbol{x}). \tag{38}$$

Now we state the second proposition.

Proposition 2. *For any input $\boldsymbol{x}$ and network parameter $\boldsymbol{\theta}$ and any direction $\boldsymbol{v} \in \mathbb{R}^{d_{in}}$, if the network has LayerNorm, then we know there exists a universal constant C, such that for any $\lambda \geq 0$, we have*

$$k_{\mathrm{NTK}}(\boldsymbol{x}, \boldsymbol{x} + \lambda \boldsymbol{v}) \leq C. \tag{39}$$

Proof. Since for finite range, there always exists a constant upper bound, we just need to analyze the case for $\lambda \to +\infty$ and shows that it is constant bounded. First compute $\nabla_{\boldsymbol{\theta}} f_{\boldsymbol{\theta}}(\boldsymbol{x})$ and get

$$\frac{\partial f}{\partial \boldsymbol{a}}\bigg|_{\boldsymbol{x}} = \sigma(\psi(\boldsymbol{W}\boldsymbol{x} + \boldsymbol{b})), \tag{40}$$

$$\frac{\partial f}{\partial \boldsymbol{W}}\bigg|_{\boldsymbol{x}} = \boldsymbol{a}^\top \sigma'(\psi(\boldsymbol{W}\boldsymbol{x} + \boldsymbol{b}))\dot{\psi}(\boldsymbol{W}\boldsymbol{x} + \boldsymbol{b})\boldsymbol{x}, \tag{41}$$

$$\frac{\partial f}{\partial \boldsymbol{b}}\bigg|_{\boldsymbol{x}} = \boldsymbol{a}^\top \sigma'(\psi(\boldsymbol{W}\boldsymbol{x} + \boldsymbol{b}))\dot{\psi}(\boldsymbol{W}\boldsymbol{x} + \boldsymbol{b}). \tag{42}$$

These quantities are all constant bounded. Next we compute $\lim_{\lambda \to \infty} \nabla_{\boldsymbol{\theta}} f_{\boldsymbol{\theta}}(\boldsymbol{x} + \lambda \boldsymbol{v})$

$$\frac{\partial f}{\partial \boldsymbol{a}}\bigg|_{\boldsymbol{x}+\lambda\boldsymbol{v}} = \sigma(\psi(\boldsymbol{W}(\boldsymbol{x} + \lambda\boldsymbol{v}) + \boldsymbol{b}))), \tag{43}$$

$$\frac{\partial f}{\partial \boldsymbol{W}}\bigg|_{\boldsymbol{x}+\lambda\boldsymbol{v}} = \boldsymbol{a}^\top \sigma'(\psi(\boldsymbol{W}(\boldsymbol{x} + \lambda\boldsymbol{v}) + \boldsymbol{b})))\dot{\psi}(\boldsymbol{W}(\boldsymbol{x} + \lambda\boldsymbol{v}) + \boldsymbol{b})(\boldsymbol{x} + \lambda\boldsymbol{v}), \tag{44}$$

$$\frac{\partial f}{\partial \boldsymbol{b}}\bigg|_{\boldsymbol{x}+\lambda\boldsymbol{v}} = \boldsymbol{a}^\top \sigma'(\psi(\boldsymbol{W}\boldsymbol{x} + \boldsymbol{b}))\dot{\psi}(\boldsymbol{W}(\boldsymbol{x} + \lambda\boldsymbol{v}) + \boldsymbol{b}). \tag{45}$$

According to the property of LayerNorm in Equation (38), we have

$$\begin{aligned}
\overline{\lim_{\lambda\to\infty}} \frac{\partial f}{\partial \boldsymbol{a}}\bigg|_{\boldsymbol{x}+\lambda\boldsymbol{v}} &= \overline{\lim_{\lambda\to\infty}} \sigma(\psi(\boldsymbol{W}(\boldsymbol{x} + \lambda\boldsymbol{v}) + \boldsymbol{b})) && (46)\\
&= \sigma(\psi(\boldsymbol{W}(\lambda\boldsymbol{v}))) && (47)\\
&= \sigma(\psi(\boldsymbol{W}\boldsymbol{v})) = \text{Constant} && (48)
\end{aligned}$$

$$\begin{aligned}
\overline{\lim_{\lambda\to\infty}} \frac{\partial f}{\partial \boldsymbol{W}}\bigg|_{\boldsymbol{x}+\lambda\boldsymbol{v}} &= \overline{\lim_{\lambda\to\infty}} \boldsymbol{a}^\top \sigma'(\psi(\boldsymbol{W}(\boldsymbol{x} + \lambda\boldsymbol{v}) + \boldsymbol{b})))\dot{\psi}(\boldsymbol{W}(\boldsymbol{x} + \lambda\boldsymbol{v}) + \boldsymbol{b})(\boldsymbol{x} + \lambda\boldsymbol{v}) && (49)\\
&= \overline{\lim_{\lambda\to\infty}} \boldsymbol{a}^\top \sigma'(\psi(\boldsymbol{W}\boldsymbol{v})))\dot{\psi}(\boldsymbol{W}(\boldsymbol{x} + \lambda\boldsymbol{v}) + \boldsymbol{b})(\boldsymbol{x} + \lambda\boldsymbol{v}) && (50)\\
&= \overline{\lim_{\lambda\to\infty}} \boldsymbol{a}^\top \sigma'(\psi(\boldsymbol{W}\boldsymbol{v})))\sqrt{d} \cdot \left(\frac{\boldsymbol{I}}{\|\boldsymbol{P}\lambda\boldsymbol{W}\boldsymbol{v}\|} - \frac{\boldsymbol{P}(\lambda\boldsymbol{W}\boldsymbol{v})(\lambda\boldsymbol{W}\boldsymbol{v})^\top\boldsymbol{P}}{\|\boldsymbol{P}(\lambda\boldsymbol{W}\boldsymbol{v})\|^3} \right) \boldsymbol{P}(\boldsymbol{x} + \lambda\boldsymbol{v}) && (51)\\
&= \overline{\lim_{\lambda\to\infty}} \boldsymbol{a}^\top \sigma'(\psi(\boldsymbol{W}\boldsymbol{v})))\sqrt{d} \cdot \left(\frac{\boldsymbol{P}(\boldsymbol{x} + \lambda\boldsymbol{v})}{\lambda\|\boldsymbol{P}\boldsymbol{W}\boldsymbol{v}\|} - \frac{\boldsymbol{P}\boldsymbol{W}\boldsymbol{v}\boldsymbol{v}^\top\boldsymbol{W}^\top\boldsymbol{P}(\boldsymbol{x} + \lambda\boldsymbol{v})}{\lambda\|\boldsymbol{P}\boldsymbol{W}\boldsymbol{v}\|^3} \right) && (52)\\
&= \boldsymbol{a}^\top \sigma'(\psi(\boldsymbol{W}\boldsymbol{v})))\sqrt{d} \cdot \left(\frac{\boldsymbol{P}\boldsymbol{v}}{\|\boldsymbol{P}\boldsymbol{W}\boldsymbol{v}\|} - \frac{\boldsymbol{P}\boldsymbol{W}\boldsymbol{v}\boldsymbol{v}^\top\boldsymbol{W}^\top\boldsymbol{P}\boldsymbol{v}}{\|\boldsymbol{P}\boldsymbol{W}\boldsymbol{v}\|^3} \right) && (53)\\
&= \text{Constant}. && (54)\\
\overline{\lim_{\lambda\to\infty}} \frac{\partial f}{\partial \boldsymbol{b}}\bigg|_{\boldsymbol{x}+\lambda\boldsymbol{v}} &= \overline{\lim_{\lambda\to\infty}} \boldsymbol{a}^\top \sigma'(\psi(\boldsymbol{W}\boldsymbol{v})))\sqrt{d} \cdot \left(\frac{\boldsymbol{I}}{\lambda\|\boldsymbol{P}\boldsymbol{W}\boldsymbol{v}\|} - \frac{\boldsymbol{P}\boldsymbol{W}\boldsymbol{v}\boldsymbol{W}\boldsymbol{v})^\top\boldsymbol{P}}{\lambda\|\boldsymbol{P}(\boldsymbol{W}\boldsymbol{v})\|^3} \right) \boldsymbol{P} && (55)\\
&= 0. && (56)
\end{aligned}$$

Therefore we know its limit is also constant bounded. So we know there exists a universal constant with respect to $\boldsymbol{\theta}, \boldsymbol{x}, \boldsymbol{v}$ such that $k_{\mathrm{NTK}}(\boldsymbol{x}, \boldsymbol{x} + \lambda\boldsymbol{v}) = \left\langle \frac{\partial f(\boldsymbol{x})}{\partial \boldsymbol{\theta}_i}, \frac{\partial f(\boldsymbol{x}+\lambda\boldsymbol{v})}{\partial \boldsymbol{\theta}_i} \right\rangle \leq C$.

E Experiment Setup

SEEM Experiments For the experiments presented in Section Section 3.1, we adopted TD3 as our baseline, but with a modification: instead of using an exponential moving average (EMA), we directly copied the current Q-network as the target network. The Adam optimizer was used with a learning rate of 0.0003, $\beta_1 = 0.9$, and $\beta_2 = 0.999$. The discount factor, γ, was set to 0.99. Our code builds upon the existing TD3+BC framework, which can be found at `https://github.com/sfujim/TD3_BC`.

SEEM Reduction Experiments For the experiments discussed in Section Section 4, we maintained the same configuration as in the SEEM experiments, with the exception of adding regularizations and normalizations. LayerNorm was implemented between the linear and activation layers with learnable affine parameters, applied to all hidden layers excluding the output layer. WeightNorm was applied to the output layer weights.

Offline RL Algorithm Experiments For the experiments presented in Section Section 5, we used true offline RL algorithms including TD3+BC, IQL, Diff-QL, and CQL as baselines. We implement our method on the top of official implementations of TD3+BC and IQL; for CQL and Diff-QL, we use reliable JAX implementations. LayerNorm was directly added to the critic network in these experiments.

Linear Decay of Inverse Q-value with SGD Given that the explosion in D4RL environments occurs very quickly in the order of $\frac{1}{1-C'\lambda\eta t}$ and is difficult to capture, we opted to use a simple toy task for these experiments. The task includes a continuous two-dimensional state space $s = (x_1, x_2) \in \mathcal{S} = R^2$, where the agent can freely navigate the plane. The action space is discrete, with 8 possible actions representing combinations of forward or backward movement in two directions. Each action changes the state by a value of 0.01. All rewards are set to zero, meaning that the true Q-value should be zero for all state-action pairs. For this task, we randomly sampled 100 state-action pairs as our offline dataset. The Q-network was implemented as a two-layer MLP with a hidden size of 200. We used SGD with a learning rate of 0.01, and the discount factor, γ was set to 0.99.

F More Experiments

Benchmarking Normalizations. Previously, we have demonstrated that LayerNor, BatchNorm, and WeightNorm can effectively maintain a low SEEM and stabilize Q convergence in Section 4. Our next goal is to identify the most suitable regularization method for the value network in offline RL. Prior research has shown that divergence is correlated with poor control performance[37, 18]. In this context, we evaluate the effectiveness of various regularization techniques based on their performance in two distinct settings - the Antmaze task and the X% Mujoco dataset we mentioned above. Previous offline RL algorithms have not performed particularly well in these challenging scenarios. As displayed in Figure 11, TD3+BC, when coupled with layer normalization or batch normalization, yields significant performance enhancement on the 10% Mujoco datasets. The inability of batch normalization to improve the performance might be attributed to the oscillation issue previously discussed in Section 4. In the case of Antmaze tasks, which contain numerous suboptimal trajectories, we select TD3 with a diffusion policy, namely Diff-QL [38], as our baseline. The diffusion policy is capable of capturing multi-modal behavior. As demonstrated in Figure 11 and Table 2, LayerNorm can markedly enhance performance on challenging Antmaze tasks. In summary, we empirically find LayerNorm to be a suitable normalization for the critic in offline RL.

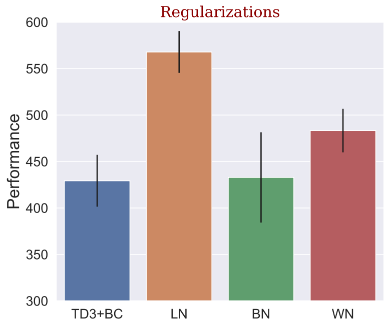


Figure 11: Normalizations effect on 10% Mujoco Locomotion Datasets.

Table 2: Normalizations effect on two challenging Antmaze tasks.

Dataset	diff-QL	LN	BN	WN
antmaze-large-play-v0	1.6	**72.7**	1.0	**35.0**
antmaze-large-diverse-v0	4.4	**66.5**	2.1	**42.5**

How LayerNorm should be added. The inclusion of LayerNorm is situated between the linear and activation layers. However, the ideal configuration for adding LayerNorm can vary and may depend on factors such as 1) the specific layers to which LayerNorm should be added, and 2) whether or not to apply learnable per-element affine parameters. To explore these variables, we conducted an assessment of their impacts on performance in the two most challenging Antmaze environments. Our experimental setup mirrored that of the Antmaze experiments mentioned above, utilizing a three-layer MLP critic with a hidden size configuration of (256,256,256). We evaluated variants where LayerNorm was only applied to a portion of hidden layers and where learnable affine parameters were disabled. As seen in Table 3, the performances with LayerNorm applied solely to the initial layers LN (0), LN (0,1) are considerably lower compared to the other setups in the 'antmaze-large-play-v0' task, while applying LayerNorm to all layers LN(0,1,2) seems to yield the best performance. For the 'antmaze-large-diverse-v0' task, performances seem to be more consistent across different LayerNorm applications. Overall, this analysis suggests that applying LayerNorm to all layers tends to yield the best performance in these tasks. Also, the utilization of learnable affine parameters appears less critical in this context.

Table 3: The effect of LayerNorm implementations on two challenging Antmaze tasks.

Dataset	w.o. LN	LN (0)	LN (0,1,)	LN (1,2)	LN (2)	LN (0,1,2)	LN (no learnable)
antmaze-large-play-v0	1.6	0	0	8.3	17.8	**72.7**	**72.8**
antmaze-large-diverse-v0	4.4	**60.2**	**68**	**77.1**	**65.5**	**66.5**	**66.7**

G Discussion

SEEM and Deadly Triad. Deadly Triad is a term that refers to a problematic interaction observed in reinforcement learning algorithms, where off-policy learning, function approximation, and bootstrapping converge, leading to divergence during training. Existing studies primarily analyze linear functions as Q-values, which tend to limit the analysis to specific toy examples. In contrast, our work uses NTK theory to provide an in-depth understanding of the divergence of Q-values in non-linear neural networks in realistic settings, and introduces SEEM as a tool to depict such divergence. SEEM can be used to understand the Deadly Triad as follows: If a policy is nearly on-policy, $\boldsymbol{X}_t^*$ is merely a perturbation of $\boldsymbol{X}$. Consequently, $\boldsymbol{A}_t = \gamma \boldsymbol{G}_{\boldsymbol{\theta}_t}(\boldsymbol{X}_t^*, \boldsymbol{X}) - \boldsymbol{G}_{\boldsymbol{\theta}_t}(\boldsymbol{X}, \boldsymbol{X}) \approx (\gamma - 1)\boldsymbol{G}_{\boldsymbol{\theta}_t}(\boldsymbol{X}, \boldsymbol{X})$, with $\boldsymbol{G}$ tending to be negative-definite. Without function approximation, the update of $Q(\boldsymbol{X})$ will not influence $Q(\boldsymbol{X}_t^*)$, and the first term in $\boldsymbol{A}_t$ becomes zero. $\boldsymbol{A}_t = -\boldsymbol{G}_{\boldsymbol{\theta}_t}(\boldsymbol{X}, \boldsymbol{X})$ ensures that SEEM is non-positive and Q-value iteration remains non-expansive. If we avoid bootstrapping, the value iteration transforms into a supervised learning problem with well-understood convergence properties. However, when all three components in Deadly Triad are present, the NTK analysis gives rise to the form $\boldsymbol{A}_t = \gamma \boldsymbol{G}_{\boldsymbol{\theta}_t}(\boldsymbol{X}_t^*, \boldsymbol{X}) - \boldsymbol{G}_{\boldsymbol{\theta}_t}(\boldsymbol{X}, \boldsymbol{X})$, which may result in divergence if the SEEM is positive.

Policy Constraint and LayerNorm. We have established a connection between SEEM and value divergence. As shown in Figure 6, policy constraint alone can also control SEEM and prevent divergence. In effect, policy constraint addresses an aspect of the Deadly Triad by managing the degree of off-policy learning. However, an overemphasis on policy constraint, leading to excessive bias, can be detrimental to the policy and impair performance, as depicted in Figure 7. Building on this insight, we focus on an orthogonal perspective in deadly triad - regularizing the generalization capacity of the critic network. Specifically, we propose the use of LayerNorm in the critic network to inhibit value divergence and enhance agent performance. Policy constraint introduces an explicit bias into the policy, while LayerNorm does not. Learning useful information often requires some degree of prior bias towards offline dataset, but too much can hinder performance. LayerNorm, offering an orthogonal perspective to policy constraint, aids in striking a better balance.

H More Visualization Results

In Assumption 2, we posit that the direction of NTK and the policy remains stable following a certain period of training. We validates this assumption through experimental studies. We observe the convergence of the NTK trajectory and policy in all D4RL Mujoco Locomotion and Antmaze tasks, as depicted in the first two columns of Figures Figure 12, Figure 13, and Figure 14. We also illustrate the linear growth characteristic of Adam optimization (as outlined in Theorem Theorem 4) in the fourth column. As a consequence, the model parameter vectors maintain a parallel trajectory, keeping the cosine similarity near 1 as shown in the third column. Figure 15 and Figure 16 showcase how SEEM serves as a ”divergence detector”in Mujoco and Antamze tasks. The surge in the SEEM value is consistently synchronized with an increase in the estimated Q-value.

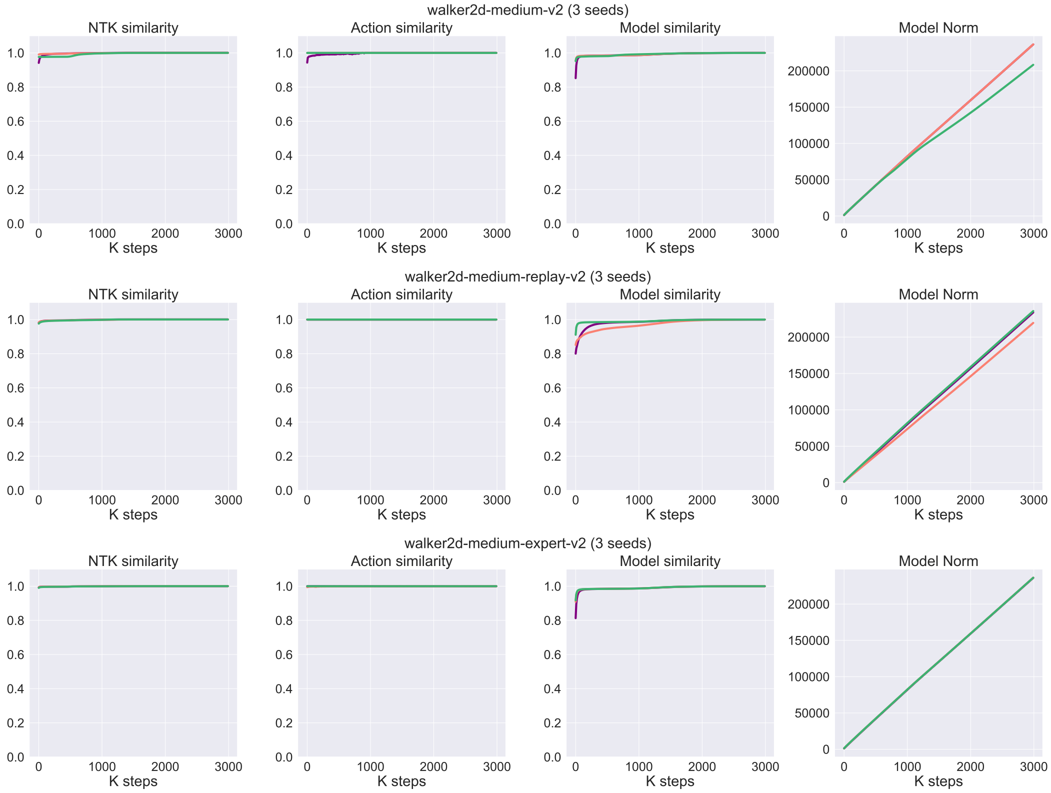


Figure 12: NTK similarity, action similarity, model parameter similarity, and model parameter norm curves in D4RL Mujoco Walker2d tasks.

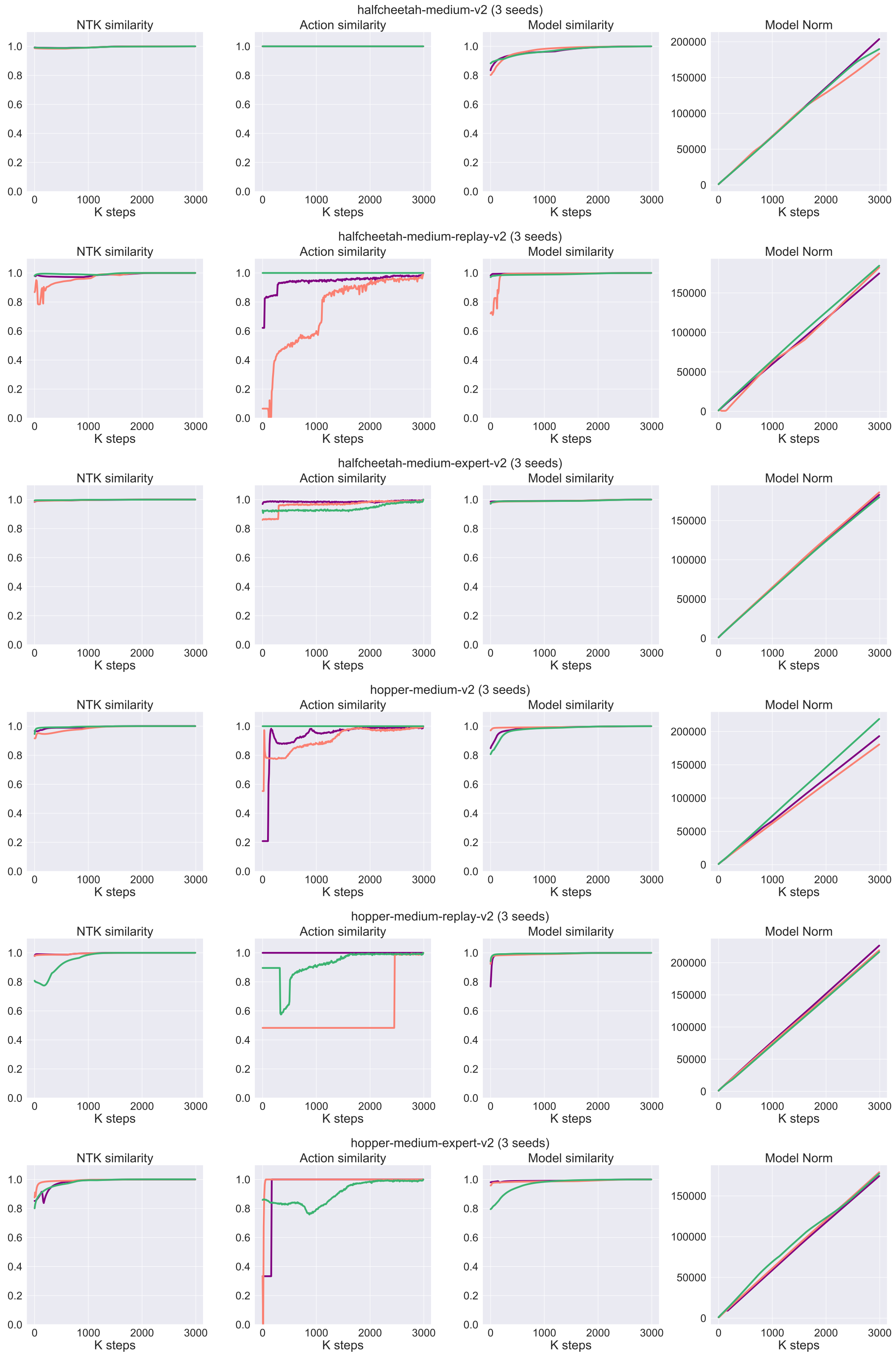


Figure 13: NTK similarity, action similarity, model parameter similarity, and model parameter norm curves in D4RL Mujoco Halfcheetah and Hopper tasks.

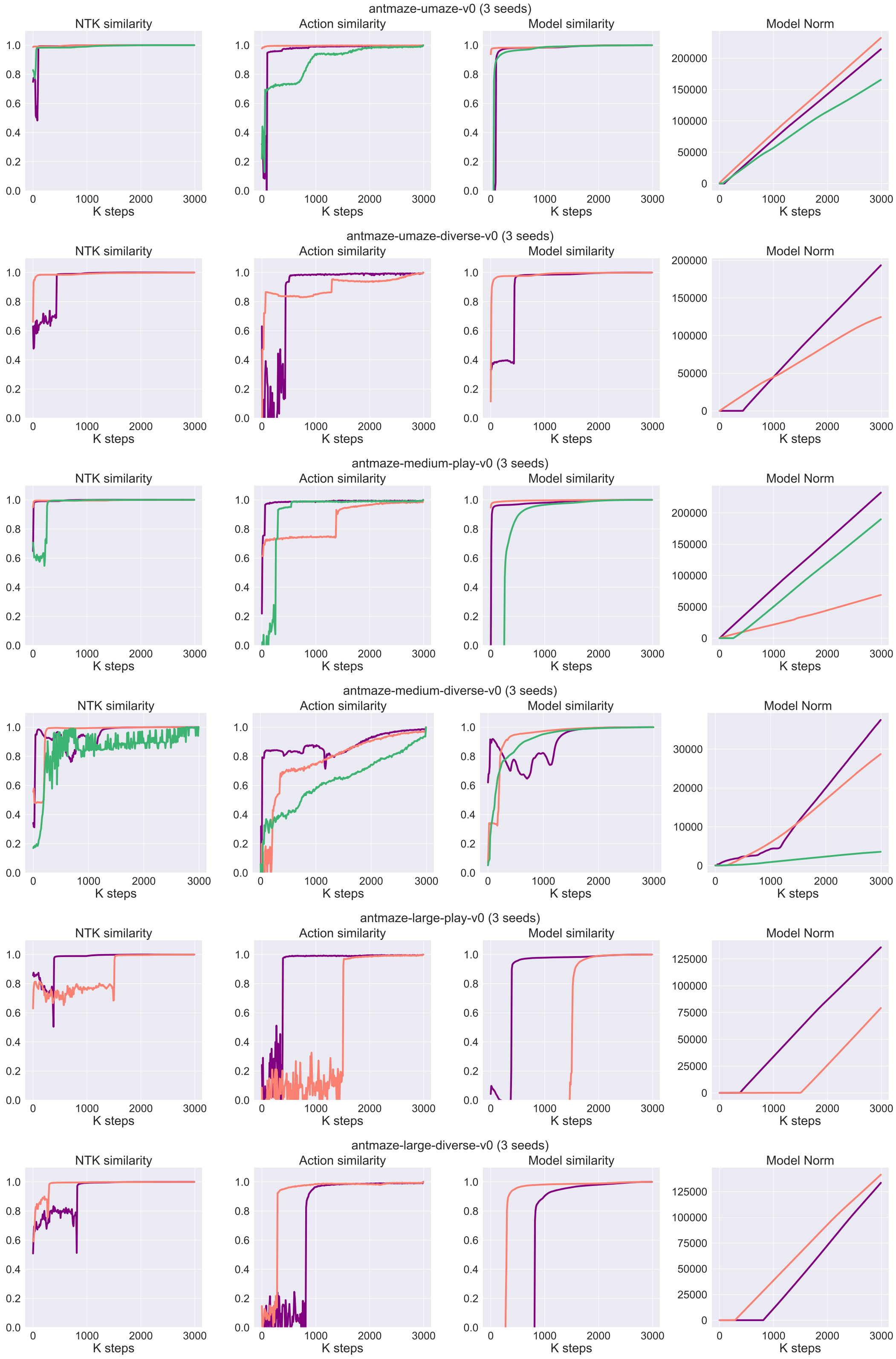


Figure 14: NTK similarity, action similarity, model parameter similarity, and model parameter norm curves in Antmaze tasks.

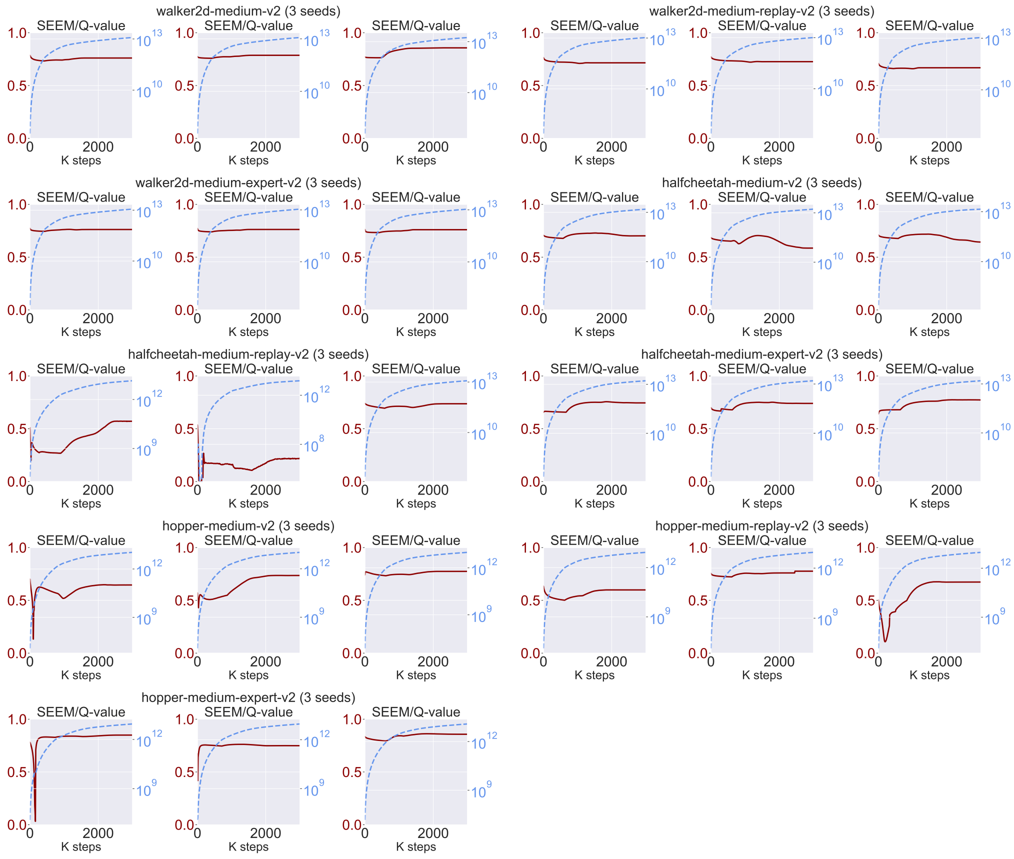
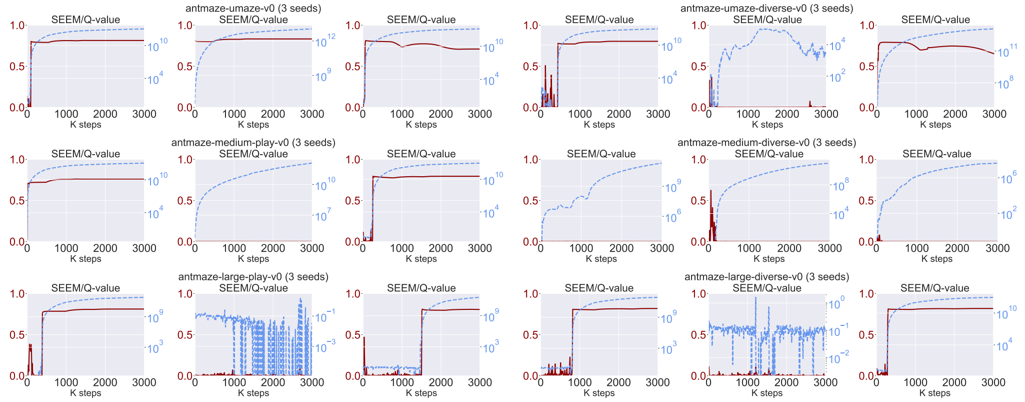
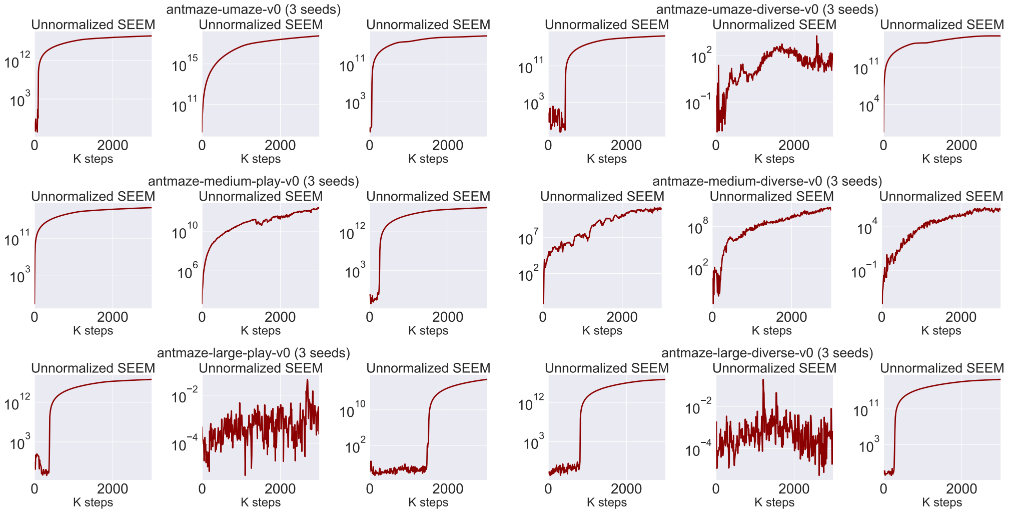


Figure 15: The normalized kernel matrix's SEEM (in red) and the estimated Q-value (in blue) in D4RL Mujoco tasks. For each environment, results from three distinct seeds are reported.



(a) The normalized kernel matrix's SEEM (in red) and the estimated Q-value (in blue) in D4RL Mujoco tasks.



(b) The unnormalized kernel matrix's SEEM. The three curves in each environment correspond directly to those presented in Figure (a)

Figure 16: In Figure (a), an inflation in the estimated Q-value coincides with a surge in the normalized SEEM. However, there are some anomalies, such as the second running in the 'umaze-diverse' environment, where the Q-value rises while the unnormalized SEEM remains low. However, the corresponding normalized SEEM in Figure (b) suggests an actual inflation of SEEM. Furthermore, for scenarios where the Q-value converges, as seen in the second running in 'large-diverse', the unnormalized SEEM maintains an approximate zero value.