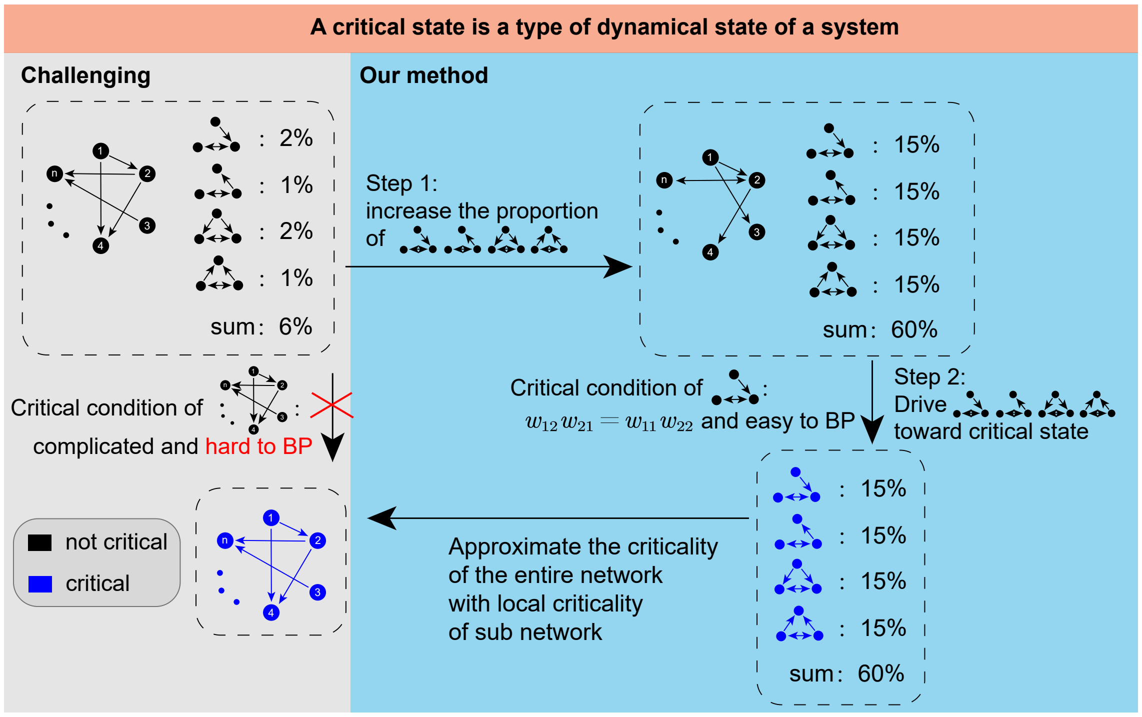
Graphical Abstract

Critical-state-accelerated RNN-based reinforcement learning*

Wangzi Yao, Yue Sun, Bo Xu, Tielin Zhang



Highlights

Critical-state-accelerated RNN-based reinforcement learning

Wangzi Yao, Yue Sun, Bo Xu, Tielin Zhang

- Addresses the difficulty of embedding critical states into high dimensional networks with a backpropagation compatible method
- Central idea is to approximate the global network critical state using many local three node subnetworks (motifs)
- Matrix based and differentiable counting increases target motif counts during training
- Simple critical state conditions derived for three node subnetworks
- Faster learning on MuJoCo, Atari and POMDP reinforcement learning tasks in standard benchmarks

Critical-state-accelerated RNN-based reinforcement learning

Wangzi Yao[a,b,1], Yue Sun[c,1], Bo Xu[a,*], Tielin Zhang[b,c,*]

[a]*Institute of Automation, Chinese Academy of Sciences, Beijing, China*
[b]*School of Artificial Intelligence, University of Chinese Academy of Sciences, Beijing, China*
[c]*Center for Excellence in Brain Science and Intelligence Technology, State Key Laboratory of Brain Cognition and Brain-inspired Intelligence Technology, Institute of Neuroscience, Chinese Academy of Sciences, Shanghai, China*

Abstract

Numerous studies have highlighted the benefits of critical states (criticality). However, deriving closed-form criticality conditions in high-dimensional networks remains challenging. As a result, most existing methods for embedding criticality rely on iterative or numerical approaches, which are often intractable and do not support backpropagation (BP). To overcome this limitation, we introduced a matrix-based criticality regularizer as an additional loss function, and that the criticality of the whole network could be approximated by integrating multiple critical states of 3-node meta-networks. We showed that recurrent neural networks (RNNs) using the proposed criticality loss function could be effectively incorporated into various popular reinforcement learning architectures (e.g., SAC, PPO, TD3, DDPG, VPG).

[⋆]This work was supported by the Strategic Priority Research Program of Chinese Academy of Sciences (Grant No. XDB1010302), the National Key R&D Program of China (2025ZD0217200), CAS Project for Young Scientists in Basic Research (YSBR-116), Youth Innovation Promotion Association CAS, the Lingang Laboratory Fund (Grant No. LG-GG-202402-06-07, LGL-1987-09), the Shanghai Municipal Science and Technology Project (Grant No. 25ZR1401370, 25LN3200400), and Special Support Project of Guangdong Province (Grant No.0720240209). The numerical calculations in this study were carried out on the ORISE Supercomputer.

[*]Corresponding authors
Email address: `zhangtielin@ion.ac.cn` (Tielin Zhang)

[1]These authors contributed equally to this work.

Furthermore, they achieved higher reward performance and faster convergence speed in MuJoCo Atari and POMDP tasks compared to both vanilla RNNs and some benchmark algorithms. Hence, the critical states were verified important for dynamical network learning.

Keywords: Critical state, dynamics, motif, reinforcement learning

1. Introduction

Critical state (criticality) is a dynamic property that exists between stability and chaos in dynamic systems, which has been identified containing many advantageous characteristics. Since Langton firstly proposed "computation at the critical point" and discussed embedding criticality into computational systems and networks [1], many followers have proved the upcoming benefits of critical states, including optimal computational capability [2], maximal signal transmission [3, 4], efficient information storage [5], and heightened sensitivity to sensory input [6, 7, 8]. Among dynamical systems, the human brain is distinctive, combining rich dynamics and energy efficiency while maintaining high performance [9].

Many efforts have been made to embed criticality in high-dimensional artificial neural networks (ANNs). An efficient method to measure criticality is proposed by estimating the spectral radius of high-order matrices, and then successfully embedded to the echo state network [10, 11, 12, 13, 14], whereby the majority of weights are frozen to maintain criticality, leaving only the input and output layers trainable. Another non-freezing synapse-inspired method is proposed where synaptic weights are adjusted according to spike timing-dependent plasticity (STDP), towards spontaneous adjustment of critical states [15, 16, 17]. However, the goals of these methods are mainly reproducing physically- or biologically-interpretable, with serious limitations of dynamic adjustment of synaptic weights. As a result, conventional methods could only test for relatively simple reinforcement learning tasks, such as a planar navigation task (containing a random start position, a fixed target, and no obstacles [18]) or a working memory task [19].

Hence, a new method is required that could balance both learning ability and criticality in more complex and practical tasks. To address this issue, we design an additional tractable loss function (criticality loss), along with a traditional error-based loss function, to automatically measure the criticality of networks during the learning procedure. Furthermore, to overcome the challenge of deriving high-dimensional ordinary differential equations (ODEs) in the loss of criticality, we decompose the criticality of the global network to those of local diverse circuits (defined as four special types of network motifs [20]). A further insight underlying this approach is that criticality constraints in local circuits (low-dimensional ANNs) are inherently simpler and more analyzed than those in global network (high dimensional ANNs). In addition, we can also increase the proportion of some specific motif types, such as those with high criticality features (in four candidate network motifs), to enhance the critical states of the whole network.

Furthermore, we extend the proposed method to more complex and practical Open-AI MuJoCo [21], Atari [22] and POMDP [23] reinforcement learning tasks, and observe accelerated reward performance and convergence speed during network learning. We found that criticality here plays an important role in balancing the trade off between exploration and exploitation, exhibiting diverse stable, critical, and chaotic behaviors, as shown in Fig. 1.

In summary, our main contributions are shown below.

- We successfully embedded critical states into an RNN during reinforcement learning, by identifying a new criticality loss function that is computational friendly with the original error loss function. Different from some traditional iterative, numerical, or intractable methods that sacrificed learning ability, this new design is fully supported by the BP procedure [24].

- We resolved the challenging problem of deriving high-dimentional ODE by decomposing it into local circuits of network motifs first, selecting four types of specific motifs with criticality, and improving network criticality by enhancing the proportion of selected motif types.

- Higher reward performance and convergence speed are reached by the proposed algorithm in the reinforcement learning of MuJoCo, Atari and POMDP benchmark tasks. Compared to previous work confined to toy tasks, the proposed algorithm could scale the learning capability to handle more difficult RL tasks and learn faster.

- We experimentally verified that approximating the critical state of the entire network using a large number of local critical states is successful. Specifically, the spectral norm of the entire network converged to 1 during training, which is a hallmark of the network's critical state.

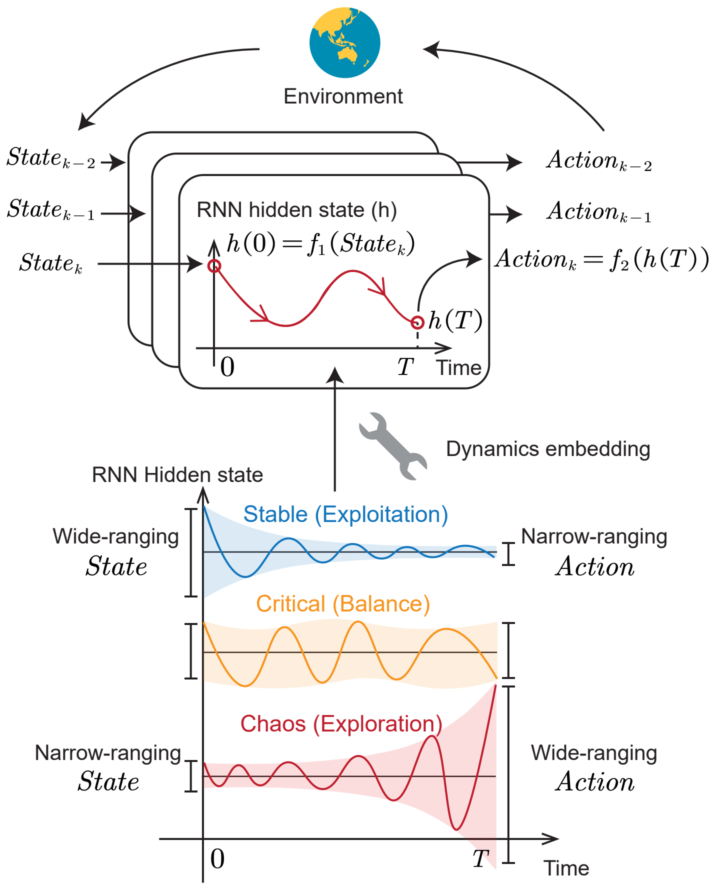


Figure 1: The diagram depicting the reinforcement learning paradigm using RNNs and three dynamical states of RNNs. **Upper diagram:** A dynamical RNN agent receives states (and rewards) from and send actions to environment. $h(t)$: Hidden states of RNNs at time t. $State_k$: The step k input of RNN-Actor. $Action_k$: The network output at step K. **Bottom diagram:** Three dynamical states of RNN during network learning to balance exploration and exploitation, including stable (high-to-low variance), critical (balanced variance), and chaotic (low-to-high variance) states.

2. Related Work

In neuroscience and statistical physics, critical systems are widely regarded as maximizing information transmission and tuning their dynamic range, enabling adaptation to complex environments while maintaining rich internal dynamics [3, 25, 26, 27, 28]. Early work focused on studying the properties of critical networks using theoretical models, and controlling the critical state of relatively small neural networks.

Mean-field theory has been proposed to maintain the input covariance from diverging, allowing some wide networks to retrain critical states and long-term information propagation ability after network initialization [29, 30]. Further research identified that the network could have higher criticality after adding residual connections [31]. Orthogonal initialization and dynamic equidistance ensure that the backpropagation gradient does not explode or decay with the number of layers, which is another method of achieving critical state propagation [32].

However, these previous two types of methods are mainly based on initialization rules without further adaptation to learning tasks. Conventional echo state network uses both initialization methods (i.e., make the spectral radius approaches 1, requiring calculating the maximum eigenvalue of a high-order matrix) and adaptation methods (i.e., linear scaling during netowrk learning) [33, 34, 35]. Another adaptation method is to observe the network transition from a subcritical state to a critical state by adjusting the average connectivity, whereby once the avalanche size distribution follows a power law, the system reaches a critical state [10, 36]. There are also some self-organized adaptation method to achieve criticality, such as brain-inspired spike timing-dependent plasticity that allows the network to gradually conform to the power-law distribution structure [15]. However, these criticality-control methods are not gradient-friendly without a defined loss function, which stopped their further application on more complex tasks, such as reinforcement learning (RL) tasks.

In RL, an agent interacts with an environment by observing states (observations), selecting actions, and receiving scalar rewards. Formally, this interaction is modeled as a Markov Decision Process (MDP), defined by the tuple $(\mathcal{S}, \mathcal{A}, P, R, \gamma)$, where $\mathcal{S}$ is the set of observations, $\mathcal{A}$ the action space, $P(s' \mid s, a)$ the transition dynamics, $R(s, a)$ the reward function, and $\gamma \in [0, 1)$ the discount factor [37]. Early RL algorithms, such as Q-learning [38] and SARSA [39], learn value functions that estimate expected returns for state–action pairs, while policy-based methods directly optimize stochastic policies $\pi(a \mid s; \theta)$ via gradient ascent on expected cumulative reward. Actor–critic architectures combine these two approaches by maintaining both a parameterized policy (the actor) and a value function estimator (the critic) that provides low-variance gradient estimates of the advantage $A(s, a)$ [40]. Notable examples include A3C/A2C [41] and DDPG [42] for continuous control. Among the policy-gradient methods, proximal policy optimization (PPO) [43, 44, 45] has become especially popular due to its simplicity and robustness. PPO maximizes a surrogate objective that clips policy updates within a trust region, thereby stabilizing training without the complexity of second-order methods like TRPO [46]. Our work mainly builds on the PPO framework to study how inducing network criticality can accelerate convergence and improve policy exploration.

3. Preliminaries

In this section, we briefly review the basic notions that our method builds on: reinforcement learning, recurrent neural networks, ordinary differential equations, and dynamical criticality.

3.1. Reinforcement Learning

We consider a standard reinforcement learning (RL) setting in which an agent interacts with an environment modeled as a Markov decision process (MDP). At each discrete time step t, the agent receives a state (or observation) s_t, selects an action $a_t \sim \pi_\theta(\cdot \mid s_t)$ according to a policy π_θ with parameters θ, and obtains a scalar reward r_t. The environment then transitions to a new state s_{t+1} according to an unknown transition kernel. The goal of RL is to learn a policy that maximizes the expected discounted return $\mathbb{E}_\pi\big[\sum_{t=0}^{\infty} \gamma^t r_t\big]$ with discount factor $\gamma \in (0, 1)$. In partially observable environments, the policy can be conditioned on the entire history through a recurrent state rather than on s_t alone, which naturally motivates the use of recurrent neural networks.

3.2. Recurrent Neural Networks

Recurrent neural networks (RNNs) provide a compact way to summarize past inputs into a hidden state. A typical discrete-time RNN update can be written as

$$h_{t+1} = \phi(W h_t + U x_{t+1} + b), \tag{1}$$

where h_t is the hidden state, x_{t+1} is the input at time $t + 1$, W and U are weight matrices, b is a bias vector, and $\phi(\cdot)$ is a nonlinear activation function. The hidden state h_t can then be used to parameterize a policy and/or value function in RL, allowing the agent to integrate information over time and handle partial observability. In this work, we view the recurrent core as a dynamical system whose qualitative behavior can be shaped by structural constraints on W.

3.3. Ordinary Differential Equations

A useful perspective is to regard the RNN dynamics as a discretization of an underlying continuous-time system. Let $h(t)$ denote a continuous-time hidden state and $u(t)$ a time-varying input. An ordinary differential equation (ODE)

$$\frac{dh(t)}{dt} = f(h(t), u(t); \theta) \tag{2}$$

specifies how the state evolves over time under a vector field f. Discrete-time RNNs can be seen as applying a numerical integration scheme (e.g., Euler method) to such an ODE, with the weights playing the role of parameters in f. This connection allows us to analyze the recurrent dynamics with tools from dynamical systems theory and to formulate structural conditions under which the system operates in different dynamical regimes.

3.4. Criticality in Dynamical Systems

Dynamical systems often exhibit qualitatively different regimes of behavior. In an *ordered* (or stable) regime, small perturbations to the state are gradually damped out: trajectories starting from slightly different initial conditions quickly converge to the same fixed point or a simple limit cycle. In a *chaotic* regime, the opposite happens: arbitrarily small differences in initial conditions are amplified over time, and trajectories diverge and explore a large portion of the state space.

A system is said to operate near *criticality* when it is close to the boundary between these ordered and chaotic regimes. At criticality, perturbations neither vanish immediately nor blow up uncontrollably; instead, they propagate over intermediate time scales and spatial scales. This leads to rich but still controllable dynamics, with long-range correlations and a balance between sensitivity and stability. In recurrent networks, operating near criticality has been argued to support both information retention and flexible computation. Our method exploits this view by designing motif-based structural constraints that steer the recurrent dynamics toward such a critical regime.

4. Methods

In this section, we first introduce four special types of three-node subnetworks, referred to as network motifs. Then we analyze the closed-form criticality condition of one class of these motifs. Finally, we approximate the critical state of the entire network by increasing the proportion of these critical-state motifs.

4.1. Methods overview

Our framework couples four components in a single training pipeline. First, we represent the RNN as a collection of three-node motifs and write each motif as a low-dimensional dynamical system. Based on this formulation, we derive an explicit criticality condition that characterizes when a motif operates on the boundary between stable and unstable dynamics. We then construct a differentiable motif-counting loss that matches the empirical distribution of motif types in the RNN to a desired target (e.g., brain-inspired statistics). In parallel, we define a differentiable criticality loss that measures how far each motif is from the criticality condition and drives motifs toward the critical state. During RL training, these two losses are added to the standard policy/value objective so that the network simultaneously learns a good control policy, the desired motif structure, and near-critical dynamics.

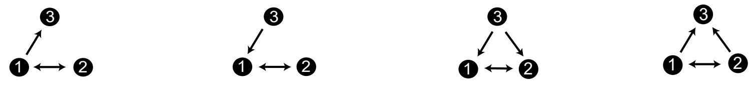
4.2. Four types of critical network Motifs

An n-dimensional hidden-layer RNN with residual connections without network inputs can be viewed as a discrete dynamical system, for every $i \in 1, 2, \cdots, N$,

$$h_i(t+1) = h_i(t) + \tanh\left(\sum_{j=1}^{N} w_{ij} h_j(t) + b\right), \quad (3)$$

where i, j are the index of neurons and N is the total number of hidden neurons. t is the propagation time. $w_{i,j}$ is the synaptic weights. b is the bias.

This iterative equation (3) can be seen as an Euler discretization of the continuous dynamics of the following ODEs [47], shown as following:



Motif 4 *Motif* 5 *Motif* 9 *Motif* 10

$$\begin{pmatrix} w_{11} & w_{12} & w_{13} \\ w_{21} & w_{22} & 0 \\ 0 & 0 & w_{33} \end{pmatrix} \quad \begin{pmatrix} w_{11} & w_{12} & 0 \\ w_{21} & w_{22} & 0 \\ w_{31} & 0 & w_{33} \end{pmatrix} \quad \begin{pmatrix} w_{11} & w_{12} & 0 \\ w_{21} & w_{22} & 0 \\ w_{31} & w_{32} & w_{33} \end{pmatrix} \quad \begin{pmatrix} w_{11} & w_{12} & w_{13} \\ w_{21} & w_{22} & w_{23} \\ 0 & 0 & w_{33} \end{pmatrix}$$

Figure 2: Four types of topologically distinct motifs that have higher criticality capability.

$$\frac{dh_i(t)}{dt} = \tanh\left(\sum_{j=1}^{N} w_{ij}h_j + b\right), \quad i = 1, 2, \cdots, N. \tag{4}$$

In this study, we focus on more three-node network motifs, which contain 13 topologically distinct types [20] where four main subtypes are plotted with higher criticality capability (Fig. 2, upper panel). We then convert these network topology to a matrix format and find that most of these matrices have second-order principal submatrix, which is a good tool to help us to make a further analysis about the high-dimensional networks.

4.3. Matrix-based Motifs and their criticality

Prior to analyzing the criticality of motifs, we employ **theorem** 1 to simplify the form of the motif's ODE.

Theorem 1. *For an RNN with residual connections, the local linearization of a motif around its fixed point is given as follows,*

$$\begin{pmatrix} \frac{dh_1}{dt} \\ \frac{dh_2}{dt} \\ \frac{dh_3}{dt} \end{pmatrix} = \begin{pmatrix} w_{11} & w_{12} & w_{13} \\ w_{21} & w_{22} & w_{23} \\ w_{31} & w_{32} & w_{33} \end{pmatrix} \begin{pmatrix} h_1 \\ h_2 \\ h_3 \end{pmatrix}. \tag{5}$$

Proof. The continuous-time formulation of an RNN with residual connection is equation (4), and the three-node motif is a 3-dimensional RNN with node number $n = 3$. We first compute its Jacobian matrix at the fixed point $\boldsymbol{h}^*$,

$$J = \begin{pmatrix} w_{11}G_1(\boldsymbol{h}^*) & w_{12}G_1(\boldsymbol{h}^*) & w_{13}G_1(\boldsymbol{h}^*) \\ w_{21}G_2(\boldsymbol{h}^*) & w_{22}G_2(\boldsymbol{h}^*) & w_{23}G_2(\boldsymbol{h}^*) \\ w_{31}G_3(\boldsymbol{h}^*) & w_{32}G_3(\boldsymbol{h}^*) & w_{33}G_3(\boldsymbol{h}^*) \end{pmatrix}, \tag{6}$$

where $\boldsymbol{h}^* = (h_1^*, h_2^*, h_3^*)$ and for every $i \in 1, 2, 3$,

$$G_i\left(\boldsymbol{h}^*\right) := 1 - \left[\tanh\left(\sum_{j=1}^{3} w_{ij} h_j^* + b_i\right)\right]^2 .$$

Furthermore, with the help of the Hartman-Grobman theorem, we obtain the linearization form of the three-node motif's ODE around the fixed point $\boldsymbol{h}^*$ as

$$\frac{dh_i}{dt} = \sum_{j=1}^{3} J_{ij} h_j, i = 1, 2, 3. \tag{7}$$

Moreover, since $\boldsymbol{h}^*$ is defined as the fixed point, which means that

$$\tanh\left(\sum_{j=1}^{n} w_{ij} h_j^* + b\right) = 0,$$

and

$$G_i\left(\boldsymbol{h}^*\right) = 1, i = 1, 2, 3. \tag{8}$$

In combination with (6)-(8), we obtain the conclusion that the linearization of the motif around its fixed point could be simplified as (5). $\square$

The connection matrices of four special motifs with ID 4, 5, 9, and 10 can be expressed as

$$w_4 = \begin{pmatrix} w_{11} & w_{12} & w_{13} \\ w_{21} & w_{22} & 0 \\ 0 & 0 & w_{33} \end{pmatrix}, w_5 = \begin{pmatrix} w_{11} & w_{12} & 0 \\ w_{21} & w_{22} & 0 \\ w_{31} & 0 & w_{33} \end{pmatrix},$$
$$w_9 = \begin{pmatrix} w_{11} & w_{12} & w_{13} \\ w_{21} & w_{22} & w_{23} \\ 0 & 0 & w_{33} \end{pmatrix}, w_{10} = \begin{pmatrix} w_{11} & w_{12} & 0 \\ w_{21} & w_{22} & 0 \\ w_{31} & w_{32} & w_{33} \end{pmatrix}.$$

For convenience, we let $\hat{w} = \begin{pmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \end{pmatrix}$, then the characteristic equations of the above four matrices are the same in the following,

$$\det(\lambda I - w_i) = (\lambda - w_{33})(\lambda^2 - (tr\hat{w})\lambda + det\hat{w}) = 0,$$

which implies that one of the eigenvalues is w_{33}. We assume $max\{w_{11}, w_{22}, w_{33}\} < 0$, then the solution along with the eigenvector's direction of eigenvalues w_{33}

converges to zero. Hence, stability analysis of four special motifs only needs to consider the following two-dimensional ODEs,

$$\begin{pmatrix} \frac{dh_1}{dt} \\ \frac{dh_2}{dt} \end{pmatrix} = \begin{pmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \end{pmatrix} \begin{pmatrix} h_1 \\ h_2 \end{pmatrix}. \tag{9}$$

Theorem 2. *Under the assumption that* $max\{w_{11}, w_{22}\} < 0$*, when* $w_{12}w_{21} - w_{11}w_{22} > 0$*, the dynamic system is unstable. when* $w_{12}w_{21} - w_{11}w_{22} < 0$*, the dynamic system is stable. It implies that condition* $w_{12}w_{21} = w_{11}w_{22}$ *is a sufficient condition for criticality of equations (B.1).*

Proof. The characteristic equation of the matrix $\hat{w}$ is

$$\lambda^2 - (w_{11} + w_{22})\,\lambda + w_{11}w_{22} - w_{12}w_{21}. \tag{10}$$

According to the root formula for quadratic equations, we obtain that

$$\lambda_{1,2} = \frac{1}{2}\left(\operatorname{tr}\hat{w} \pm \sqrt{(\operatorname{tr}\hat{w})^2 - 4\det\hat{w}}\right). \tag{11}$$

For convenience, let $\Delta = (\operatorname{tr}\hat{w})^2 - 4\det\hat{w}$. If $w_{12}w_{21} - w_{11}w_{22} > 0$, then $det\hat{w} < 0$. It implies that $\Delta < tr\hat{w}$ and $\lambda_1 > 0, \lambda_2 < 0$. In this situation, (B.1) can be similar diagonalized to a diagonal matrix of the following form

$$\begin{pmatrix} \frac{dh_1}{dt} \\ \frac{dh_2}{dt} \end{pmatrix} = P^{-1} \begin{pmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{pmatrix} P \begin{pmatrix} h_1 \\ h_2 \end{pmatrix}. \tag{12}$$

Let $\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = P \begin{pmatrix} h_1 \\ h_2 \end{pmatrix}$, then (B.4) can be rewrite as

$$\begin{pmatrix} \frac{dx_1}{dt} \\ \frac{dx_2}{dt} \end{pmatrix} = \begin{pmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}. \tag{13}$$

The general solution of the above equation is

$$x_1 = c_1 e^{\lambda_1 t}, x_2 = c_2 e^{\lambda_2 t}, \tag{14}$$

where c_1, c_2 are constants related to the initial value.

In combination with the results that when $w_{12}w_{21} - w_{11}w_{22} > 0$, we obtain that $\lambda_1 > 0, \lambda_2 < 0$, $\begin{pmatrix} h_1 \\ h_2 \end{pmatrix} = P^{-1} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$ and $x_1 = c_1 e^{\lambda_1 t}$ divergence with increasing t. So we determined that when $w_{12}w_{21} - w_{11}w_{22} > 0$, the dynamic system of (B.1) is unstable.

The situation $w_{12}w_{21} - w_{11}w_{22} < 0$ is sufficient for the stability of the dynamical system in (B.1). The proof is analogous, with more details referred to in the Appendix.

$\square$

It is worth noting that **theorem** 2 offers only a sufficient condition, but in fact it is also necessary. This result can be verified by analyzing the general solution of ODE (B.1) under all parameter combinations. Because of the complexity, we present the results in Fig. 3 and discuss the computations in the Appendix.

In our proof, we used “unstable” instead of “chaotic”. This is because saddle points are not always chaotic. However, this analysis applies to the two-dimensional case. In higher dimensions, unstable systems are highly likely to be chaotic. Therefore, it is reasonable to replace “chaotic” with “unstable” to determine the critical location.

4.4. Gradient-friendly loss function

Here, we first introduce a gradient-friendly motif counting term using matrix calculation, which is then added to the loss function to guide network learning. Let w be the adjacency matrix of a network with N neuron nodes. In motifs, network connections are represented discretely as 0 or 1. To approximate this representation, we apply a sigmoid function.

$$W_{ij} = sigmoid\left(amp\left(w_{ij}^2 - bias^2\right)\right), \tag{15}$$

to the weight w_{ij}, mapping its values into $(0,1)$ and preserving differentiability. The sigmoid is defined with a threshold of *bias* and a steepness (gain) of *amp*. We define the $N_1 \times N_2$ matrix where all elements are 1 as $\mathbf{1}_{N_1 \times N_2}$. $S = \mathbf{1}_{N \times N} - I$ and $L = \mathbf{1}_{N,1}$ are two constant matrices. Next, we introduce variant $\bar{W} = S - W$ of W. $W_{ij} = 1$ indicates a link from i to j, whereas $\bar{W}_{ij} = 1$ indicates no link from i to j. By combining W and $\bar{W}$, we obtain four new variants,

$$\begin{aligned} W_0 &= \bar{W} \odot \bar{W}^T, \\ W_1 &= W \odot \bar{W}^T, \\ W_2 &= \bar{W} \odot W^T, \\ W_3 &= W \odot W^T. \end{aligned} \tag{16}$$

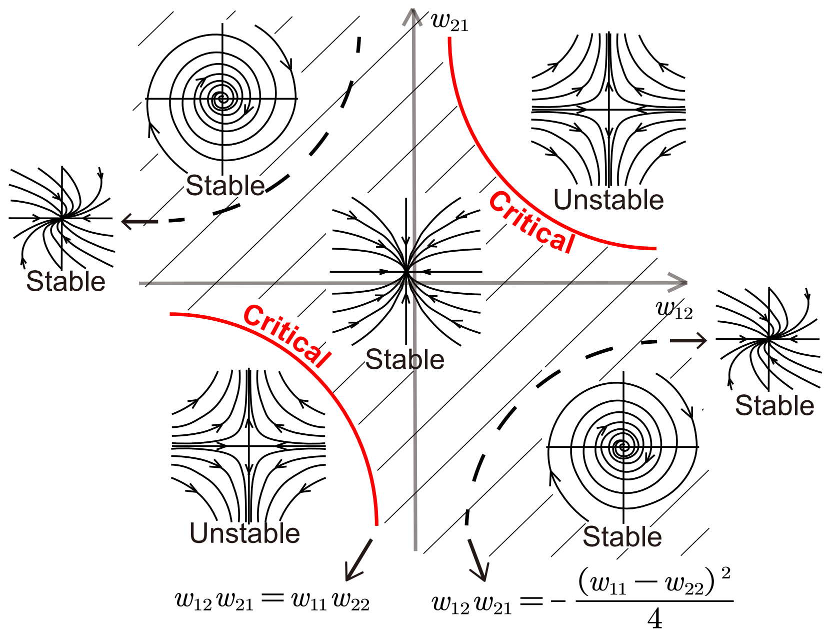


Figure 3: A schematic diagram of the general solution for all combinations of parameters. In the large coordinate system, the axes represent parameters w_{12} and w_{21}. The small coordinate systems show the trajectories of the general solutions. Each point in the large coordinate system corresponds to a 2 dimensional ODE, from which its general solution can be plotted as shown in the small coordinate system. We divide these into several regions based on the type of general solution. Systems whose general solutions eventually contract to the origin are stable, which are indicated by shaded areas, while those that do not are unstable. The critical state is the boundary between the stable and chaotic regions, indicated by the red line. Therefore, we conclude that **theorem** 2 is in fact both necessary and sufficient.

Here, $W_{0,ij} = 1,\ W_{1,ij} = 1,\ W_{2,ij} = 1,\ W_{3,ij} = 1$ denote no connection between i and j; a connection from i to j only; a connection from j to i only; and bidirectional connections between i and j, respectively. The counting algorithm for motifs 4, 5, 9 and 10 is

$$\begin{aligned} Num_4 &= L\left(W_3 \odot \left(W_1 W_0\right)\right) L^T, \\ Num_5 &= L\left(W_3 \odot \left(W_2 W_0\right)\right) L^T, \\ Num_9 &= L\left(W_3 \odot \left(W_1 W_2\right)\right) L^T, \\ Num_{10} &= L\left(W_3 \odot \left(W_2 W_1\right)\right) L^T. \end{aligned} \tag{17}$$

The correctness of these formulas is easy to prove. Taking Num_4 as an example, the usage of $W_1 W_0$ here is similar to the Dijkstra algorithm [48], where the ij element counts the number of 2 step paths between them satisfy the motif 4 condition $1-3-2$ (the number notations here correspond to those in Fig. 2). $W_3 \odot$ considers whether the one-step path between ij satisfies the condition $1-2$. So the ij element of $W_3 \odot (W_1 W_0)$ is the number of motif 4 in the entire network with base $i-j$ as $1-2$. L and L^T perform the summation. The criticality constraint for motifs 4, 5, 9 and 10 is

$$\begin{aligned} Cri_4 &= L\left(Q \odot W_3 \odot \left(W_1 W_0\right)\right) L^T, \\ Cri_5 &= L\left(Q \odot W_3 \odot \left(W_2 W_0\right)\right) L^T, \\ Cri_9 &= L\left(Q \odot W_3 \odot \left(W_1 W_2\right)\right) L^T, \\ Cri_{10} &= L\left(Q \odot W_3 \odot \left(W_2 W_1\right)\right) L^T, \end{aligned} \tag{18}$$

where $P = (I \odot w)\left(\mathbf{1}_{N\times N}\right)(I \odot w)$ and $Q = \left(w \odot w^T - P\right) \odot \left(w \odot w^T - P\right)$. All of them are tractable, so it is convenient to add them into loss. Equation 18 is derived from equation 17 and theorem 2. Theorem 2 specifies the conditions for a single special motif to reach a critical state, while Eq.18 locates all special motifs within the network and applies the constraints of Theorem 2 to each one.

Ultimately, Eq.17 is elevated in the loss function to increase the number of the four special motifs. Eq.18 is reduced in the loss function to constrain motifs into critical states.

5. Experiments

First, we evaluated the accelerated performance of our proposed method on the PPO architecture in four OpenAI-MuJoCo reinforcement learning benchmark tasks and one classic control benchmark. Then we tested our designed loss function and found that the proportion of specific motifs were indeed increased. We further found that most of these network motifs in neural network were driven into critical states. Finally, ablation experiments proved the usefulness of the proposed methods during network learning. Besides PPO, we also demonstrated the acceleration effect of the critical state on several other RL architectures: VPG, DDPG, TD3, and SAC.

5.1. Experimental Setup

In motifs, network connections are represented discretely as 0 or 1. To approximate this representation, we apply Eq.15 to the weight w_{ij} in weight matrix W, mapping its values into $(0, 1)$ and preserving differentiability. The sigmoid is defined with a threshold of $bias = 5e - 2$ and a steepness (gain) of $amp = 1e3$. In all experiments, we use an RNN with residual connections to simulate the dynamical network, with a hidden layer size (i.e., number of neurons) of 512. To ensure fairness in the experiment, the learning rate for all experiments was set to 3e-5. Each experiment was repeated 20 times, using random seeds from 1 to 20 to ensure the reliability of the experimental results. The memory buffer size for reinforcement learning is set to 2048, and only one epoch is trained for the same memory buffer. The maximum interaction time limit for the environment is set to 10,000, and interaction will not stop unless the environment actively terminates. The batch size for model training is set to 64. The PPO parameters are set as follows: discount factor γ = 0.98, GAE λ = 0.98, and PPO pruning threshold ε = 0.2. The simulation step size for each update is 2048. We configure DDPG with Polyak-averaged target networks (coefficient 0.995), discount factor γ = 0.99, and Gaussian action noise with standard deviation 0.1. For VPG, we use a discount factor γ = 0.98. For training rounds in different scenarios, we ensure that they are long enough to converge. Our experiments were run on 40G A100 x4.

5.2. Mujoco Benchmarks

The OpenAI MuJoCo physics engine was selected as RL benchmark tasks for continuous control. MuJoCo provides a diverse collection of high-fidelity simulation environments, such as Walker2d, Ant, HalfCheetah, Inverted Pendulum, and Inverted Double Pendulum, each characterized by continuous state and action spaces that closely emulate real-world robotic dynamics.

This richness in both kinematic complexity and contact modeling makes MuJoCo an ideal platform for evaluating the impact of criticality constraints on learning performance. All environments expose high-dimensional observation vectors (e.g., joint positions, velocities, body orientations) and continuous torque-based control signals, allowing us to rigorously assess how our motif- and criticality-based regularization schemes influence convergence speed, stability, and final task performance under a uniform algorithmic framework.

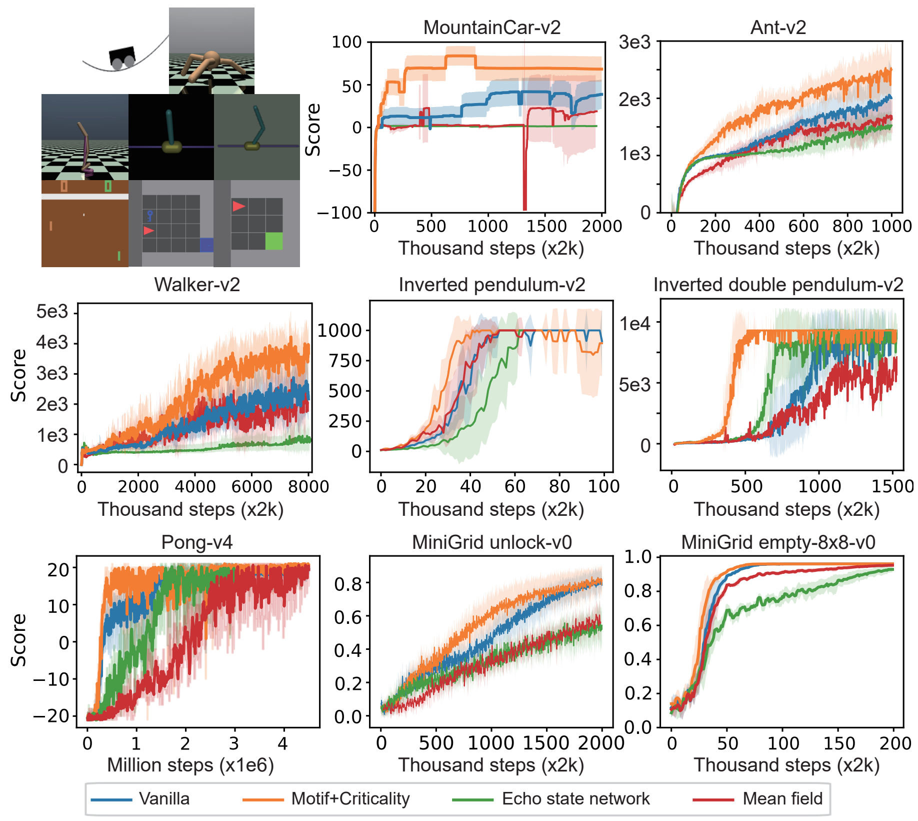


Figure 4: Diagram depicting learning curves in eight RL benchmarks. The vertical axis represents the reward scores, and the horizontal axis represents the training progress. Our critical state embedding method (orange line) accelerates PPO, which learns faster than vanilla PPO (blue line), ESN and mean-field method.

5.3. Criticality-accelerated learning

We conducted experiments in MuJoCo reinforcement learning environments. MuJoCo features continuous action and state spaces, which are better suited than discrete settings for testing criticality as a dynamical property. Our experiments compared standard PPO with PPO augmented by critical states. For each setup, we performed 20 parallel tests using random seeds of 1 to 20, and reported the average performance.

As shown in Fig. 4, in most of tasks, RNNs using PPO and critical-state constraints reached higher reward performance than RNNs using vanilla PPO, with mean 34.36% improvement (from 42.15 to 67.20, MountainCar-V2; from 1922.06 to 2451.86, Ant-V2; from 2071.18 to 3828.22, Walker-V2). In addition, we also found in all experimental results, the convergence rates (defined as the reward value that curves reached the predefined step values, i.e., 50% learning time) were also improved, with mean 121.55% improvement (from 25.12 to 71.90, MountainCar-V2; from 498.71 to 801.04, Inverted pendulum-V2; from 1273.94 to 1801.44, Ant-V2; from 1170.06 to 2371.77, Walker-V2; from 2844.33 to 9010.50, Inverted double pendulum-V2).

Specifically, in the inverted pendulum-v2 and inverted double pendulum-v2 environments, both methods converged to the same optimal value (1000 and 9010.50). However, criticality-constrained PPO reached the optimum more quickly. In the mountaincar-v2 environment, the constrained PPO not only learned faster, but also escaped local optima and converged to higher scores than the standard PPO.

Furthermore, we compared the learning time (real time, second) required to achieve the same score, where the score was taken as the lower of the two converged scores. As shown in Table 1, our method achieved the same score but with significantly less time.

Table 1: Comparison of the real simulation time required to achieve the same score, measured in seconds.

	PPO+Motif+Criticality	PPO
IP-v2	284.93 ± 25.94	383.18 ± 46.26
IDP-v2	2705.10 ± 244.24	5862.83 ± 536.53
Ant-v2	3033.74 ± 141.80	4355.24 ± 384.89
Walker-v2	7905.02 ± 1425.14	14053.49 ± 2533.34

5.4. Constraint proportions of four-type Motifs

We incorporate criticality into the network via two mechanisms: increasing the proportion of four special motifs, and constraining these motifs to enter the critical state. The experiments here verified whether the constraints we propose can increase the motif proportion to the set value. The target proportions for motifs 4, 5, 9, and 10 (critical motifs) were all set to 0.1, meaning that critical motifs will account for 40% of the three-point networks in the entire high-dimensional network. Table. 2 shows the proportions of motifs 4, 5, 9, and 10 before and after training for different tasks. The experiment was repeated 20 times, and the average results were recorded.

Overall, under different training tasks, the frequencies of motifs after training are relatively close to the set values, indicating that the motif frequency constraints we proposed are effective. The proportion of critical motifs before training is very low, but it shows a significant increase after training. Additionally, the variance of repeated experiments is small relative to the average value, demonstrating the stability of our method.

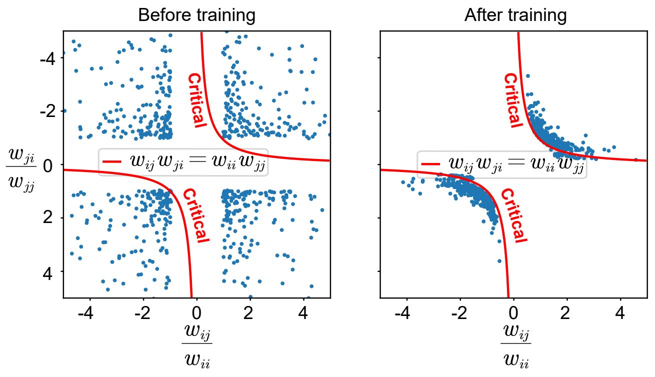


Figure 5: Diagram depicts the weight distribution of all four types of special network motifs in the RNN. Each blue dot represents the synaptic weights of a network motif (3-dimensional subnetwork). The red line represents the critical condition (see Methods for more details). The closer of a blue point to the line represents the closer to critical state of the network.

5.5. Constraint of network criticality

Then a large number of network motifs in the whole network is constrained to a critical state. We verify that the effect of the constraint is as expected. In the "Criticality in Motif" subsection, we derived the $w_{ij}w_{ji} = w_{ii}w_{jj}$ condition under which a single motif (a three-node subnetwork) reaches criticality. Fig. 5 shows the two curves that define this condition. In the experiment, we plot each four special types of network motif in the high-dimensional network according to its weight. Each motif

Table 2: The motif proportion constraint changes the proportion of critical motifs (4, 5, 9, 10) before/after training. Each task spans two rows (Before / After).

Task	Target proportion	Phase	Critical motifs			
			Motif 4	Motif 5	Motif 9	Motif 10
MountainCar-v2	0.1000	Before (±1e-4)	0.0126	0.0126	0.0010	0.0011
		After (±5e-4)	0.0844	0.0844	0.1023	0.1024
Inverted pend-v2	0.1000	Before (±1e-4)	0.0125	0.0125	0.0011	0.0011
		After (±5e-4)	0.0844	0.0845	0.1017	0.1019
Ant-v2	0.1000	Before (±1e-4)	0.0126	0.0125	0.0011	0.0010
		After (±5e-4)	0.0845	0.0845	0.1021	0.1019
Walker-v2	0.1000	Before (±1e-4)	0.0126	0.0126	0.0011	0.0011
		After (±5e-4)	0.0846	0.0846	0.1015	0.1016

corresponds to one point, and points closer to the curves satisfy the criticality condition more closely.

Before training, the points are scattered across all four quadrants and lie far from the curves. After training, the points originally in quadrants II and IV migrate to quadrants I and III, moving much closer to the curves. This shift indicates that, after training, the majority of special four types of network motifs in RNNs have indeed entered the critical state.

5.6. Spectral Radius Measures Critical States in High-Dimensional Networks

The intuition behind our proposed method is to approximate the critical state of the entire network using a large number of critical states on 3-point networks. Here, we employ an empirical criterion to verify that our method also approximates the critical state across the entire network. This criterion states that when the spectral radius approaches 1, the entire network is generally in a critical state [49].

Using our critical state loss for training, the spectral radius of the entire network decreased from an initial value of 1.238286 to 1.000021, exhibiting a trend of convergence toward 1 (Fig. 6). Thus, our method indeed approximates the critical state of the entire network using numerous local critical states.

5.7. Ablation study

Criticality control comprises two components: increasing the proportion of four types of network motifs and driving these motifs to a critical state.

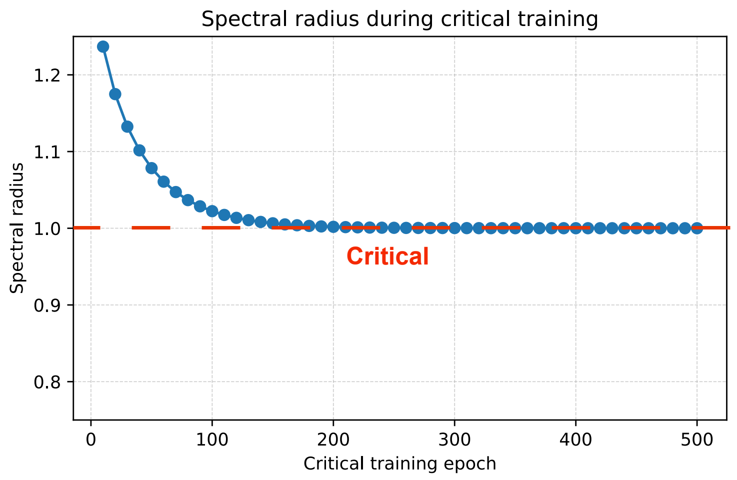


Figure 6: The spectral radius of the entire network converges to 1 as the critical state loss training proceeds. The horizontal axis represents the number of training epoch, while the vertical axis shows the spectral radius of the entire network.

We refer to the former as “Motif” and the latter as “Criticality”, and thus design 4 experimental conditions:“PPO”, “PPO+Motif”, “PPO+Criticality”, and “PPO+Motif+Criticality”. These ablation experiments are conducted in two reinforcement learning environments, ant-v2 and inverted pendulum-v2, with results averaged over 20 seeds. To enhance clarity, variance bands are omitted.

The results indicate that “PPO+Motif+Criticality” achieves the best performance, followed by “PPO+Criticality”. Since “PPO+Criticality” does not include an increased motif proportion, the number of critical motifs remains low and the acceleration effect of criticality regulation is moderate. Nevertheless, both augmented methods outperform “PPO”. These ablation experiments demonstrate that our two-step critical state control method (e.g., proportion of network motifs and critical state) works.

5.8. Extended to other RL benchmark architectures

Since our criticality control method is not limited to PPO architecture, more popular RL architectures were selected and compared to test the capability of our method, including VPG, DDPG, TD3, and SAC. We then evaluated them on the inverted pendulum-v2 task, showing the average performance over 20 runs [50]. As shown in Fig. 8, we found that our proposed

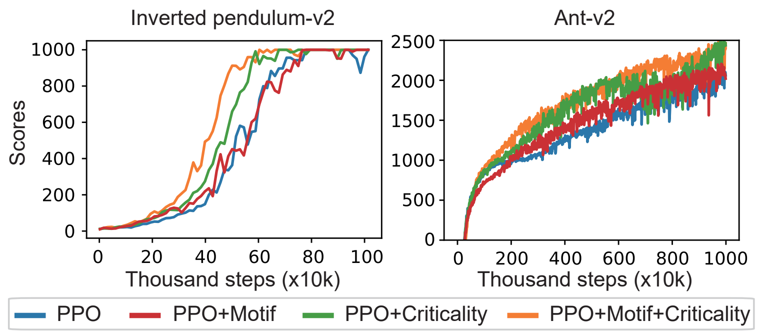


Figure 7: Ablation study of two constrains of motif proportions and critical states. Variance bands have been omitted for clarity. All experiments were repeat 20 times.

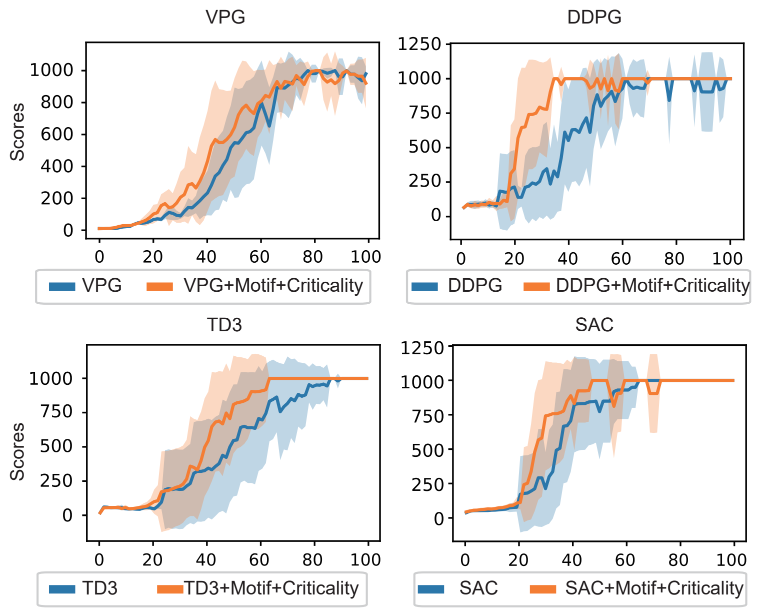


Figure 8: Diagram depicting learning curves of VPG, DDPG, TD3, and SAC after incorporating our critical state constraint. The critical state accelerates training and converges faster to the optimal solution.

criticality control technique accelerates learning in all tested frameworks, with higher convergence speed (At 50% training steps, VPG, DDPG, TD3, and SAC achieved improvements of 21.66%, 61.86%, 70.15%, and 14.93%, respectively.). This general acceleration effect reveals the potential of critical states in more benchmark RL architectures.

We note that the gains from motif+criticality are more modest for SAC/TD3 than for PPO. PPO is purely on-policy and thus more sensitive to how RNN dynamics balance exploration and exploitation, so driving the network toward a critical regime yields a larger boost, whereas SAC/TD3 already incorporate stabilizing and exploration-enhancing mechanisms, leaving less headroom.

6. Conclusion

Assessing the critical state in high-dimensional recurrent neural networks is computationally challenging. To address this, we decompose the high-dimensional network into a simpler, lower-dimensional structure—specifically, a three-node recurrent network motif represented in matrix form. This representation greatly simplifies the analysis of network dynamics. We then design a matrix-based motif loss function and use it to successfully train RNNs, guiding them toward topological configurations with enhanced criticality.

To evaluate our method, we conducted experiments on four OpenAI MuJoCo RL tasks and an additional continuous control task. The results show that RNNs trained with our approach achieve higher rewards and converge faster than standard RNNs that lack criticality constraints. An ablation study further confirms the contribution of each component to the overall effectiveness of our method.

The mathematical condition for criticality (Fig. 2), particularly the key relationship $W_{12}W_{21} = W_{11}W_{22}$, is central to guiding the network learning. A detailed mathematical demonstration of its role in controlling criticality is provided in the Appendix.

Although this study focuses on four specific network motifs, we believe that the other nine of the total 13 possible types warrant further investigation. Analyzing their stability and flexibility could be highly valuable for applying critical network learning to non-RL tasks.

We chose RL-based continuous control tasks for validation because they inherently require a balance between exploration and exploitation—a balance that networks operating near criticality are well-suited to provide, thus maximizing learning speed and cumulative reward. Furthermore, the diverse behavioral dynamics in these tasks align well with the rich neural dynamics exhibited by critical RNNs.

Appendix A. A special topological class of Motifs

The weight matrices of level 2 are

$$\begin{gathered}\begin{pmatrix} w_{11} & w_{12} & w_{13} \\ w_{21} & w_{22} & 0 \\ 0 & 0 & w_{33} \end{pmatrix}, \begin{pmatrix} w_{11} & w_{12} & 0 \\ w_{21} & w_{22} & 0 \\ w_{31} & 0 & w_{33} \end{pmatrix}, \\ \begin{pmatrix} w_{11} & w_{12} & 0 \\ w_{21} & w_{22} & 0 \\ w_{31} & w_{32} & w_{33} \end{pmatrix}, \begin{pmatrix} w_{11} & w_{12} & w_{13} \\ w_{21} & w_{22} & w_{23} \\ 0 & 0 & w_{33} \end{pmatrix}. \end{gathered} \tag{A.1}$$

They are all block matrices with similar stability conditions (see Appendix Subsection"Necessary and Sufficient Conditions for Theorem 2" for a detailed analysis). Since their stability conditions are more complex than those of level 1, they are classified as level 2.

Appendix B. Necessary and Sufficient Conditions for Theorem 2

Due to its complexity, we present the proof here that **theorem** 2 is in fact sufficient and necessary.

3 Under the assumption that $max\{w_{11}, w_{22}\} < 0$, $w_{12}w_{21} = w_{11}w_{22}$ is a sufficient condition for criticality of equations

$$\begin{pmatrix} \frac{dh_1}{dt} \\ \frac{dh_2}{dt} \end{pmatrix} = \begin{pmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \end{pmatrix} \begin{pmatrix} h_1 \\ h_2 \end{pmatrix}. \tag{B.1}$$

Proof. The necessary and sufficient conditions for criticality are all parameter combinations$(w_{12}, w_{21}, w_{11}, w_{22})$ that make the system(equation (B.1)) critical. Since criticality is defined as the boundary between chaos and stable, we first need to determine whether equation (B.1) is chaotic or stable for each parameter combination.

In the subsection "Criticality in Motifs" we define $\hat{w} = \begin{pmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \end{pmatrix}$. The characteristic equation of the matrix $\hat{w}$ is

$$\lambda^2 - (w_{11} + w_{22})\,\lambda + w_{11}w_{22} - w_{12}w_{21}. \tag{B.2}$$

According to the root formula for quadratic equations, we obtain that

$$\lambda_{1,2} = \frac{1}{2}\left(\operatorname{tr}\hat{w} \pm \sqrt{(\operatorname{tr}\hat{w})^2 - 4\det\hat{w}}\right). \tag{B.3}$$

For convenience, let $\Delta = (\operatorname{tr}\hat{w})^2 - 4\det\hat{w}$. According to Jordan's canonical theorem, matrix $\hat{w}$ can be decomposed into the form $P^{-1}JP$, where J is the Jordan canonical form of $\hat{w}$. In this situation, (B.1) can be similar diagonalized to a diagonal matrix of the following form

$$\begin{pmatrix} \frac{dh_1}{dt} \\ \frac{dh_2}{dt} \end{pmatrix} = P^{-1}JP\begin{pmatrix} h_1 \\ h_2 \end{pmatrix}. \tag{B.4}$$

Let $\left(\begin{array}{c} x_1 \\ x_2 \end{array}\right) = P\left(\begin{array}{c} h_1 \\ h_2 \end{array}\right)$, then (B.4) can be rewrite as

$$\left(\begin{array}{c} \frac{dx_1}{dt} \\ \frac{dx_2}{dt} \end{array}\right) = J\left(\begin{array}{c} x_1 \\ x_2 \end{array}\right). \tag{B.5}$$

Based on the stability invariance under similar transformations, the dynamical characteristics (chaotic or stable) of Eqs. B.1-B.5 are consistent. Therefore, we will analyze the stability of Eq. B.5 based on the 3 different forms of J.

1. For the first type of Jordan normal form $J = \begin{pmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{pmatrix}, \lambda_1 \neq \lambda_2, \Delta > 0$. The general solution is

$$x_1 = c_1 e^{\lambda_1 t}, x_2 = c_2 e^{\lambda_2 t}, \tag{B.6}$$

where c_1, c_2 are constants related to the initial value. $\operatorname{tr}\hat{w} < 0$ because $max\{w_{11}, w_{22}\} < 0$. So

$$\lambda_1 = \frac{1}{2}\left(\operatorname{tr}\hat{w} - \sqrt{(\operatorname{tr}\hat{w})^2 - 4\det\hat{w}}\right) < 0, \tag{B.7}$$

which means the system is always stable(converge to 0) in the x_1 direction. When $\det\hat{w} < 0$, $\sqrt{(\operatorname{tr}\hat{w})^2 - 4\det\hat{w}} > -\operatorname{tr}\hat{w}$. It follows that

$$\lambda_2 = \frac{1}{2}\left(\operatorname{tr}\hat{w} + \sqrt{(\operatorname{tr}\hat{w})^2 - 4\det\hat{w}}\right) > 0. \tag{B.8}$$

In this case the system is chaos(not converge to 0) in the x_2 direction. Conversely, when $\det \hat{w} > 0$, $\lambda_2 < 0$. At this point x_2 is converge to 0, the system is stable.
So $\det \hat{w} = 0$ is the boundary of stable and chaos. One of the sufficient condition for criticality is

$$w_{11}w_{22} = w_{12}w_{21}. \tag{B.9}$$

2. For the second type of Jordan normal form $\begin{pmatrix} \xi & \eta \\ -\eta & \xi \end{pmatrix}$, $\Delta < 0$. The general solution is

$$\begin{aligned} x_1(t) &= c_1 e^{\xi t}\cos(\eta t) + c_2 e^{\xi t}\sin(\eta t), \\ x_2(t) &= c_1 e^{\xi t}\sin(-\eta t) + c_2 e^{\xi t}\cos(\eta t). \end{aligned} \tag{B.10}$$

   Where η determines the rotation speed, and ξ determines convergence or divergence. ξ is the real part of the eigenvalue, so

$$\xi = \frac{1}{2}\operatorname{tr}\hat{w} < 0 \tag{B.11}$$

   Therefore, x_1 and x_2 both converge to 0. There are no critical states in this region.

3. The last type of Jordan normal form can only be $\begin{pmatrix} \lambda & 1 \\ 0 & \lambda \end{pmatrix}$ rather than $\begin{pmatrix} \lambda & 0 \\ 0 & \lambda \end{pmatrix}$. Both of them need $\Delta = 0$. At this time,

$$w_{12}w_{21} = -\frac{(w_{11} - w_{22})^2}{4}. \tag{B.12}$$

   Further calculations can yield

$$\lambda^2 - (w_{11} + w_{22})\lambda + w_{11}w_{22} - w_{12}w_{21} = 0. \tag{B.13}$$

   This means that the characteristic polynomial of $\hat{w}$ is 0. Therefore, the eigenvalue λ has only one eigenvector, and its Jordan canonical form must be $\begin{pmatrix} \lambda & 1 \\ 0 & \lambda \end{pmatrix}$. So the general solution is

$$\begin{aligned} x_1(t) &= c_1 e^{\lambda t} + c_2 e^{\lambda t} t, \\ x_2(t) &= c_2 e^{\lambda t}. \end{aligned} \tag{B.14}$$

The eigenvalue is

$$\lambda = \frac{1}{2}\operatorname{tr}\hat{w} < 0. \tag{B.15}$$

Therefore, x_1 and x_2 both converge to 0. There are no critical states in this region.

In summary, the necessary and sufficient condition for criticality is

$$w_{11}w_{22} = w_{12}w_{21}. \tag{B.16}$$

The above general solution and parameter conditions are consistent with the content shown in Figure 3. □

Appendix C. Training Pseudocode

The pseudocode for our critical embedding algorithm applied to the actor network of PPO is as follows.

Algorithm 1 PPO with Motif and Critical-State Regularization

```
1: Initialize policy and value parameters $\theta, \phi$
2: Set regularization weights $\lambda_{\text{motif}}, \lambda_{\text{crit}}$
3: for iteration $k = 1, 2, \ldots$ do
4:     Roll out policy $\pi_\theta$ to collect trajectories $\mathcal{D}$
5:     Compute returns and advantages on $\mathcal{D}$
6:     Compute standard PPO loss $\mathcal{L}_{\text{PPO}}(\theta, \phi)$
7:     Extract recurrent weight matrix $W$ from the RNN
8:     Compute motif-count vector $m(W)$ using Eq.(17) [? ]
9:     Compute motif loss
           $\mathcal{L}_{\text{motif}} = \|m(W) - m^\star\|_2^2$
10:    Compute critical-state loss $\mathcal{L}_{\text{crit}}$ using Eq.(18) # applies motif-level
       criticality condition $w_{12}w_{21} = w_{11}w_{22}$ to all motifs
11:    Total loss:
           $\mathcal{L} = \mathcal{L}_{\text{PPO}} + \lambda_{\text{motif}}\mathcal{L}_{\text{motif}} + \lambda_{\text{crit}}\mathcal{L}_{\text{crit}}$
12:    Update $\theta, \phi$ via gradient descent on $\mathcal{L}$
13: end for
```

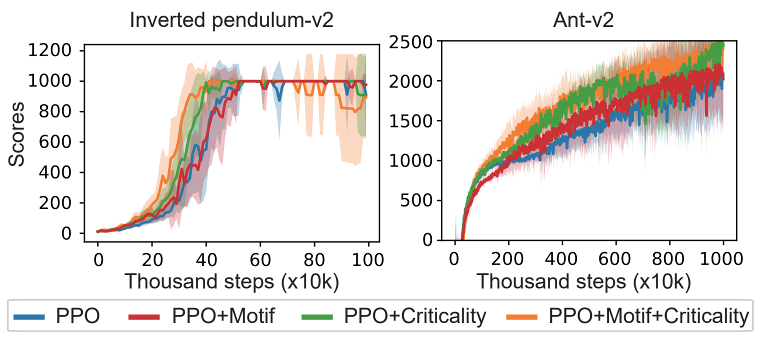


Figure D.9: Ablation study of two constrains of motif proportions and critical states. All experiments were repeat 20 times.

Appendix D. Ablation Experiment With Variance Band

As shown in Fig. D.9, Fig. 7 is provided with a variance band version. For detailed experimental setup, see Section 4.6.

References

[1] C. G. Langton, Computation at the edge of chaos: Phase transitions and emergent computation, Physica D: nonlinear phenomena 42 (1-3) (1990) 12–37.

[2] R. Legenstein, W. Maass, Edge of chaos and prediction of computational performance for neural circuit models, Neural networks 20 (3) (2007) 323–334.

[3] J. M. Beggs, D. Plenz, Neuronal avalanches in neocortical circuits, Journal of neuroscience 23 (35) (2003) 11167–11177.

[4] S. S. Schoenholz, J. Gilmer, S. Ganguli, J. Sohl-Dickstein, Deep information propagation, arXiv preprint arXiv:1611.01232 (2017).

[5] C. Haldeman, J. M. Beggs, Critical branching captures activity in living neural networks and maximizes the number of metastable states, Physical review letters 94 (5) (2005) 058101.

[6] R. Der, F. Hesse, G. Martius, Rocking stamper and jumping snakes from a dynamical systems approach to artificial life, Adaptive Behavior 14 (2) (2006) 105–115.

[7] O. Kinouchi, M. Copelli, Optimal dynamical range of excitable networks at criticality, Nature physics 2 (5) (2006) 348–351.

[8] D. R. Chialvo, Are our senses critical?, Nature physics 2 (5) (2006) 301–302.

[9] O. Arviv, A. Goldstein, O. Shriki, Neuronal avalanches and time-frequency representations in stimulus-evoked activity, Scientific reports 9 (1) (2019) 13319.

[10] P. Massobrio, V. Pasquale, S. Martinoia, Self-organized criticality in cortical assemblies occurs in concurrent scale-free and small-world networks, Scientific reports 5 (1) (2015) 10578.

[11] A. Levina, J. M. Herrmann, T. Geisel, Dynamical synapses causing self-organized criticality in neural networks, Nature physics 3 (12) (2007) 857–860.

[12] T. Matsuki, K. Shibata, Adaptive balancing of exploration and exploitation around the edge of chaos in internal-chaos-based learning, Neural Networks 132 (2020) 19–29.

[13] M. Lukoševičius, H. Jaeger, B. Schrauwen, Reservoir computing trends, KI-Künstliche Intelligenz 26 (2012) 365–371.

[14] C. Gallicchio, A. Micheli, L. Silvestri, Local lyapunov exponents of deep echo state networks, Neurocomputing 298 (2018) 34–45.

[15] N. Stepp, D. Plenz, N. Srinivasa, Synaptic plasticity enables adaptive self-tuning critical networks, PLoS computational biology 11 (1) (2015) e1004043.

[16] Y. Kawai, M. Asada, Spatiotemporal motor learning with reward-modulated hebbian plasticity in modular reservoir computing, Neurocomputing 558 (2023) 126740.

[17] G. A. D'Inverno, J. Dong, Comparison of reservoir computing topologies using the recurrent kernel approach, Neurocomputing 611 (2025) 128679.

[18] K. Shibata, Y. Sakashita, Reinforcement learning with internal-dynamics-based exploration using a chaotic neural network, in: 2015 International Joint Conference on Neural Networks (IJCNN), IEEE, 2015, pp. 1–8.

[19] D. Sussillo, L. F. Abbott, Generating coherent patterns of activity from chaotic neural networks, Neuron 63 (4) (2009) 544–557.

[20] R. Milo, S. Shen-Orr, S. Itzkovitz, N. Kashtan, D. Chklovskii, U. Alon, Network motifs: simple building blocks of complex networks, Science 298 (5594) (2002) 824–827.

[21] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, W. Zaremba, Openai gym, arXiv preprint arXiv:1606.01540 (2016).

[22] V. Mnih, Playing atari with deep reinforcement learning, arXiv preprint arXiv:1312.5602 (2013).

[23] M. Chevalier-Boisvert, B. Dai, M. Towers, R. Perez-Vicente, L. Willems, S. Lahlou, S. Pal, P. S. Castro, J. Terry, Minigrid & miniworld: Modular & customizable reinforcement learning environments for goal-oriented tasks, Advances in Neural Information Processing Systems 36 (2023) 73383–73394.

[24] D. E. Rumelhart, G. E. Hinton, R. J. Williams, Learning representations by back-propagating errors, nature 323 (6088) (1986) 533–536.

[25] A. Lerchner, W. Maass, Real-time computations at the edge of chaos in recurrent neural networks, in: Advances in Neural Information Processing Systems, Vol. 17, 2004, pp. 143–150.

[26] W. L. Shew, D. Plenz, The functional benefits of criticality in the cortex, The neuroscientist 19 (1) (2013) 88–100.

[27] H. Sompolinsky, A. Zippelius, Dynamic theory of the spin-glass phase, Physical Review Letters 47 (5) (1981) 359.

[28] L. Zhang, L. Feng, K. Chen, C. H. Lai, Edge of chaos as a guiding principle for modern neural network training, arXiv preprint arXiv:2107.09437 (2021).

[29] Y. Bahri, J. Kadmon, J. Pennington, S. S. Schoenholz, J. Sohl-Dickstein, S. Ganguli, Statistical mechanics of deep learning, Annual review of condensed matter physics 11 (1) (2020) 501–528.

[30] H. Sompolinsky, A. Zippelius, Relaxational dynamics of the edwards-anderson model and the mean-field theory of spin-glasses, Physical Review B 25 (11) (1982) 6860.

[31] G. Yang, S. S. Schoenholz, Mean field residual networks: On the edge of chaos, in: Advances in Neural Information Processing Systems, Vol. 30, 2017, pp. 7103–7112.

[32] J. Pennington, S. Schoenholz, S. Ganguli, Resurrecting the sigmoid in deep learning through dynamical isometry: theory and practice, Advances in neural information processing systems 30 (2017).

[33] H. Jaeger, The "echo state" approach to analysing and training recurrent neural networks-with an erratum note, Bonn, Germany: German national research center for information technology gmd technical report 148 (34) (2001) 13.

[34] H. Jaeger, Echo state network, scholarpedia 2 (9) (2007) 2330.

[35] B. Derrida, Dynamical phase transition in nonsymmetric spin glasses, Journal of Physics A: Mathematical and General 20 (11) (1987) L721.

[36] N. Bertschinger, T. Natschläger, R. Legenstein, At the edge of chaos: Real-time computations and self-organized criticality in recurrent neural networks, Advances in neural information processing systems 17 (2004).

[37] M. L. Puterman, Markov decision processes: discrete stochastic dynamic programming, John Wiley & Sons, 2014.

[38] C. J. Watkins, P. Dayan, Q-learning, Machine learning 8 (1992) 279–292.

[39] G. A. Rummery, M. Niranjan, On-line Q-learning using connectionist systems, Vol. 37, University of Cambridge, Department of Engineering Cambridge, UK, 1994.

[40] S. Fujimoto, H. Hoof, D. Meger, Addressing function approximation error in actor-critic methods, in: International conference on machine learning, PMLR, 2018, pp. 1587–1596.

[41] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, K. Kavukcuoglu, Asynchronous methods for deep reinforcement learning, in: International conference on machine learning, PmLR, 2016, pp. 1928–1937.

[42] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, D. Wierstra, Continuous control with deep reinforcement learning, arXiv preprint arXiv:1509.02971 (2015).

[43] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, O. Klimov, Proximal policy optimization algorithms, arXiv preprint arXiv:1707.06347 (2017).

[44] X. Zhao, D. Zhang, H. Liyuan, T. Zhang, B. Xu, Ode-based recurrent model-free reinforcement learning for pomdps, Advances in Neural Information Processing Systems 36 (2023) 65801–65817.

[45] R. S. Sutton, D. McAllester, S. Singh, Y. Mansour, Policy gradient methods for reinforcement learning with function approximation, Advances in neural information processing systems 12 (1999).

[46] J. Schulman, S. Levine, P. Abbeel, M. Jordan, P. Moritz, Trust region policy optimization, in: International conference on machine learning, PMLR, 2015, pp. 1889–1897.

[47] R. T. Chen, Y. Rubanova, J. Bettencourt, D. K. Duvenaud, Neural ordinary differential equations, Advances in neural information processing systems 31 (2018).

[48] E. W. Dijkstra, A note on two problems in connexion with graphs, in: Edsger Wybe Dijkstra: his life, work, and legacy, 2022, pp. 287–290.

[49] M. Buehner, P. Young, A tighter bound for the echo state property, IEEE transactions on neural networks 17 (3) (2006) 820–824.

[50] T. Haarnoja, A. Zhou, P. Abbeel, S. Levine, Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor, in: International conference on machine learning, Pmlr, 2018, pp. 1861–1870.