Error Amplification Limits ANN-to-SNN Conversion in Visual Continuous Control

Zijie Xu
Department of Computer Science
Peking University
Beijing, China 100871
zjxu25@stu.pku.edu.cn

Chunyu He
Department of Computer Science
Peking University
Beijing, China 100871
chunyuhe25@stu.pku.edu.cn

Peng Ma
Department of Computer Science
Peking University
Beijing, China 100871
pma25@stu.pku.edu.cn

Bingrui Guo
Department of Computer Science
Peking University
Beijing, China 100871
bguo9894@stu.pku.edu.cn

Ruiqi Li
Department of Computer Science
Peking University
Beijing, China 100871
rqli25@stu.pku.edu.cn

Abstract

Spiking Convolutional Neural Networks (SCNNs) offer a promising pathway for energy-efficient visual perception and control in embodied agents. By leveraging ANN-to-SNN conversion, we can transfer high-performance visual policies from pre-trained CNNs without expensive retraining. However, while effective in static image classification, this approach suffers severe degradation in vision-based continuous control tasks. In this work, we identify spatiotemporal error amplification as the primary cause: minor quantization errors in the spiking visual representation develop temporal correlations, triggering a compounding shift in the induced state trajectory. To mitigate this, we introduce Cross-Step Residual Potential Initialization (CRPI), a training-free strategy that ensures temporal consistency in spiking feature maps by propagating residual membrane potentials across decision steps. Our experiments on the DeepMind Control Suite with high-dimensional pixel inputs demonstrate that CRPI effectively suppresses error accumulation, enabling SCNNs to recover state-of-the-art performance in complex visual control scenarios.The code are available at https://github.com/hcypkumain/25Fall-CV-Finalwork.git.

1 Introduction

Spiking Neural Networks (SNNs) Maass [1997], Gerstner et al. [2014], inspired by the efficient information processing of biological visual systems, communicate through discrete spikes rather than continuous activations. This event-driven paradigm is inherently suitable for processing high-dimensional spatiotemporal visual data, substantially reducing energy consumption when deployed on neuromorphic hardware Merolla et al. [2014], Davies et al.

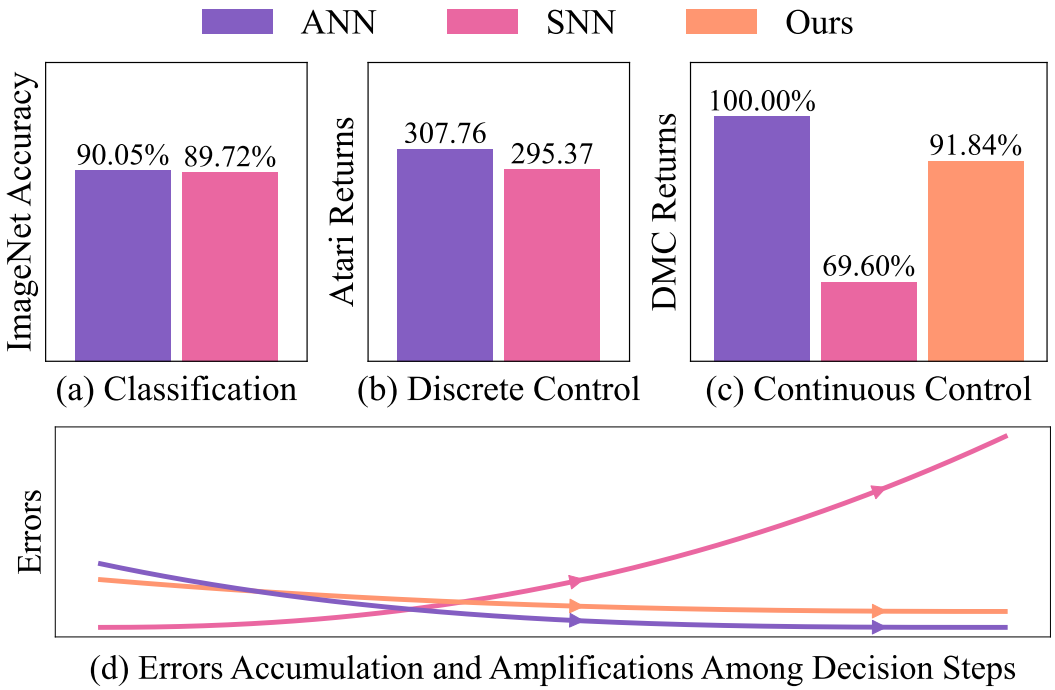


Figure 1: Challenges of ANN-to-SNN conversion across different task categories. (a) Classification accuracy on ImageNet Huang et al. [2025]. (b) Average returns in discrete control tasks on Atari Patel et al. [2019]. (c) Relative returns in continuous control tasks, averaged over six environments from the DeepMind Control Suite. Additional results in the experimental section confirm that the performance degradation is consistent across tasks. (d) Illustration of error accumulation and amplification, where trajectories generated by converted SNNs progressively diverge from those of the original ANN policies.

[2018], DeBole et al. [2019]. These properties make SNNs particularly attractive for vision-based Reinforcement Learning (RL), where agents must perceive and react to complex pixel inputs in real-time. This is crucial for embodied AI on resource-constrained edge devices, such as drones navigating via onboard cameras and IoT sensors monitoring visual scenes, where power efficiency is a critical concern.

ANN-to-SNN conversion constructs SNNs by transferring pretrained ANN weights and replacing nonlinear activations with spiking neurons, allowing SNNs to inherit strong ANN performance without additional training Cao et al. [2015], Han et al. [2020], Li et al. [2021], Deng and Gu [2021], Bu et al. [2022a, 2025]. This training-free paradigm is especially valuable in RL, where learning an agent typically requires extensive environment interaction that is costly, time-consuming, and potentially unsafe Jiang et al. [2023], Padalkar et al. [2024], Tang et al. [2025], Jayant and Bhatnagar [2022]. By reusing pretrained ANN policies, ANN-to-SNN conversion allows energy-efficient SNN agents to be deployed without further environment interaction.

Despite these advantages, the study of ANN-to-SNN conversion in RL remains limited in scope. Existing work has primarily focused on discrete control settings Patel et al. [2019], Tan et al. [2021], Kumar et al. [2025], Feng et al. [2024], while ANN-to-SNN conversion in continuous action spaces remains largely unexplored, despite its central role in real-world robotics and embodied AI systems [Kober et al., 2013, Gu et al., 2017, Brunke et al., 2022]. Figure 1(a)–(c) compares the performance of ANN-to-SNN conversion across classification, discrete control, and continuous control tasks. While existing conversion methods achieve competitive performance in classification and discrete control, they suffer substantially larger performance degradation in continuous control. This gap arises from the requirement for precise, high-dimensional vector-valued actions in continuous control, in contrast to the

categorical outputs in classification and discrete control tasks, making continuous control considerably more sensitive to conversion errors.

To understand this phenomenon, we conduct a detailed analysis of conversion errors in continuous control. We find that: (i) performance degradation in converted SNNs is primarily driven by deviations in induced state trajectories rather than instantaneous action errors; (ii) these state deviations grow progressively over decision steps along a trajectory; and (iii) action approximation errors exhibit positive temporal correlation across consecutive decision steps, amplifying even small conversion errors. As illustrated in Figure 1(d), trajectories generated by converted SNNs gradually diverge from those of the optimal ANN policies, whereas ANN policies themselves do not exhibit such progressive drift.

Guided by these findings, we propose Cross-Step Residual Potential Initialization (CRPI), a straightforward yet potent mechanism designed to curb the error amplification effect in RL-oriented ANN-to-SNN conversion. CRPI operates by utilizing the residual membrane potentials left over at the end of one decision step to inform the initialization of neuronal states at the next step. This cross-step memory serves to dampen the temporal correlation of action errors, thereby promoting the stability of the induced state trajectories, as conceptually depicted in Figure 1(d). A crucial advantage of CRPI is its training-free nature; it functions as a plug-and-play enhancement that can be effortlessly incorporated into any existing ANN-to-SNN conversion pipeline. We rigorously evaluate CRPI on a suite of continuous control benchmarks, including the vision-based tasks from the DeepMind Control (DMC) Suite. The results demonstrate that CRPI consistently boosts the performance of various leading conversion techniques and even surpasses the results achieved by SNNs trained from scratch in these demanding visual continuous control settings. Collectively, our work establishes continuous control as a critical and rigorous benchmark for evaluating ANN-to-SNN conversion strategies, where the long-horizon nature of the task can severely magnify conversion errors, with profound consequences for overall agent efficacy.

We evaluate CRPI on a range of continuous control benchmarks like vision-based environments from the DeepMind Control (DMC) Suite Tunyasuvunakool et al. [2020]. CRPI consistently improves the performance of multiple state-of-the-art ANN-to-SNN conversion methods and outperforms directly trained SNNs in challenging vision-based continuous control tasks. Our results highlight continuous control as a challenging benchmark for ANN-to-SNN conversion, where conversion errors can be strongly amplified and significantly impact long-horizon performance.

2 Related Works

2.1 ANN–SNN Conversion

ANN-to-SNN conversion typically maps ReLU activations in ANNs to the firing rates of Integrate-and-Fire neurons by accumulating spikes over time Cao et al. [2015]. However, the bounded firing rates in SNNs introduce significant errors, which is often mitigated through techniques such as weight normalization Rueckauer et al. [2017] and threshold balancing Han et al. [2020]. Temporal discretization introduces additional quantization errors, which have been addressed by methods like quantizing the source ANN activations Bu et al. [2023], Hu et al. [2023], using two-stage inference Hao et al. [2023a], improving membrane potential initialization Hao et al. [2023b], and extending neuron models with signed spikes Wang et al. [2022a], Li et al. [2022] or multiple thresholds Huang et al. [2024]. Other encoding schemes such as time-to-first-spike coding Rueckauer and Liu [2018], Zhang et al. [2019], Stanojevic et al. [2023], phase coding Kim et al. [2018], Wang et al. [2022b], burst coding Park et al. [2019], Li and Zeng [2022], Wang et al. [2025], and differential coding Huang et al. [2025], have also been explored to enhance both efficiency and expressiveness. Recent works have further extended conversion methods by allowing for approximations of general nonlinear layers Oh and Lee [2024], Jiang et al. [2024], Huang et al. [2024] and enabling conversion of Transformer architectures to SNNs Wang et al. [2023], You et al. [2024].

2.2 SNNs for Reinforcement Learning

Early works on SNNs for RL primarily relied on biologically inspired local learning rules, particularly reward-modulated spike-timing-dependent plasticity (R-STDP) and its variants Florian [2007], Frémaux and Gerstner [2016], Gerstner et al. [2018], Frémaux et al. [2013], Yang et al. [2024]. Later research introduced gradient-based optimization methods, such as spatio-temporal backpropagation (STBP) for deep spiking Q-networks Wu et al. [2018], Liu et al. [2022], Chen et al. [2022], Qin et al. [2022], Sun et al. [2022] and e-prop for policy gradient methods Bellec et al. [2020]. In continuous control, the hybrid actor-critic framework has been widely adopted, where a spiking actor network is co-trained with an ANN-based critic network Xu et al. [2025a], Tang et al. [2020, 2021], Zhang et al. [2022], Chen et al. [2024], Ding et al. [2022]. This approach has been further advanced with proxy-target mechanisms, allowing spiking actors to match or even surpass ANN policies Xu et al. [2025b].

2.3 ANN–SNN Conversion in Reinforcement Learning

ANN-to-SNN conversion in RL has also been explored in several studies. These works mainly focus on converting Deep Q-Networks (DQNs) Mnih [2013], Mnih et al. [2015] into spiking policies for Atari games Patel et al. [2019], Tan et al. [2021], as well as deploying converted agents in real-world robotic tasks, such as ball catching Feng et al. [2024] and path planning Kumar et al. [2025]. These studies report competitive performance, improved energy efficiency, and enhanced robustness of SNN-based agents. However, existing works have been limited to discrete control tasks, and ANN-to-SNN conversion in continuous control remains largely unexplored. This work shows that directly applying existing conversion techniques to continuous control leads to greater performance degradation, which forms the primary motivation for our work.

3 Preliminaries

3.1 Spiking Neural Networks

SNNs process information via discrete spike events and temporal membrane dynamics. For an Integrate-and-Fire (IF) neuron in layer l at discrete time step t, the neuronal dynamics are given by

$$\mathbf{I}^l[t] = \mathbf{W}^l \mathbf{x}^{l-1}[t] + \mathbf{b}^l, \tag{1}$$

$$\mathbf{m}^l[t] = \mathbf{v}^l[t-1] + \mathbf{I}^l[t], \tag{2}$$

$$\mathbf{o}^l[t] = H\big(\mathbf{m}^l[t] - \boldsymbol{\theta}^l\big), \tag{3}$$

$$\mathbf{x}^l[t] = \boldsymbol{\theta}^l \odot \mathbf{o}^l[t], \tag{4}$$

$$\mathbf{v}^l[t] = \mathbf{m}^l[t] - \mathbf{x}^l[t], \tag{5}$$

where $\mathbf{x}^l[t]$ denotes the post-synaptic potential, $\mathbf{m}^l[t]$ and $\mathbf{v}^l[t]$ are the pre-reset and post-reset membrane potentials respectively, $\mathbf{o}^l[t]$ is the binary spike output, and $\boldsymbol{\theta}^l$ is the firing threshold. The operator $H(\cdot)$ denotes the Heaviside step function, and $\odot$ indicates element-wise multiplication.

3.2 ANN-to-SNN Conversion

ANN-to-SNN conversion leverages the correspondence between ReLU activations in ANNs and averaged firing responses in rate-coded SNNs. In a standard feedforward ANN, the output of layer l is computed as

$$\mathbf{z}^l = \mathrm{ReLU}\big(\mathbf{W}^l_{\mathrm{ANN}} \mathbf{z}^{l-1} + \mathbf{b}^l_{\mathrm{ANN}}\big). \tag{6}$$

Starting from the discrete-time dynamics of IF neurons, the membrane potential update can be written as

$$\mathbf{v}^l[t] = \mathbf{v}^l[t-1] + \mathbf{W}^l \mathbf{x}^{l-1}[t] + \mathbf{b}^l - \mathbf{x}^l[t]. \tag{7}$$

Averaging this equation over time steps $t = 1$ to T yields

$$\frac{1}{T}\sum_{t=1}^{T}\mathbf{x}^l[t] = \mathbf{W}^l\frac{1}{T}\sum_{t=1}^{T}\mathbf{x}^{l-1}[t] + \mathbf{b}^l + \frac{\mathbf{v}^l[0] - \mathbf{v}^l[T]}{T}. \tag{8}$$

Under the standard assumption that the membrane potential of IF neurons remains bounded by the firing threshold, i.e., $\mathbf{v}^l[t] \in [0, \boldsymbol{\theta}^l)$, the residual term $\frac{\mathbf{v}^l[0]-\mathbf{v}^l[T]}{T}$ vanishes as T increases. By setting $\mathbf{W}^l = \mathbf{W}^l_{\text{ANN}}$ and $\mathbf{b}^l = \mathbf{b}^l_{\text{ANN}}$, and identifying ANN activations with the average post-synaptic potential $\mathbf{z}^{l-1} = \frac{1}{T}\sum_{t=1}^{T}\mathbf{x}^{l-1}[t]$, the time-averaged SNN response $\frac{1}{T}\sum_{t=1}^{T}\mathbf{x}^l[t]$ converges to the corresponding ANN activation $\mathbf{z}^l$.

3.3 Reinforcement Learning

RL studies the problem of an agent interacting with an environment, which is commonly formalized as a Markov Decision Process (MDP). At decision step k, the agent observes the environment state $\mathbf{s}_k \in \mathcal{S}$ and selects an action $\mathbf{a}_k \in \mathcal{A}$ according to a policy $\pi : \mathcal{S} \to \mathcal{A}$. The environment then transitions to a new state $\mathbf{s}_{k+1}$ and provides a reward $r_k = r(\mathbf{s}_k, \mathbf{a}_k)$. The objective of the agent is to maximize the expected cumulative return $R = \mathbb{E}\sum_k r_k$.

A key property of the MDP formulation is that the environment dynamics and reward depend only on the current state and action, rather than the full history of past interactions. Accordingly, at each decision step, the policy computes an action solely based on the current observation. In standard ANN-to-SNN conversion for RL, this is typically enforced by executing the SNN for a fixed internal simulation horizon of T time steps at each decision step, and initializing all neuronal states at the beginning of the next decision step. As a result, no internal neuronal states or membrane potentials are preserved across consecutive decision steps.

4 ANN-to-SNN Conversion

We implement IF neuron and three other baseline methods: Signed Neuron with Memory (SNM) Wang et al. [2022a], Multi Threshold (MT) Neuron Huang et al. [2024], and Differential coding (DC) based neuronHuang et al. [2025].

4.1 SNM neuron dynamics

SNM neuron can be regarded as an IF neuron with negative threshold and more strict spike emission condition on negative threshold in SNNs, let $\boldsymbol{m}^l(t)$ and $\boldsymbol{v}^l(t)$ denote the membrane potential of neurons in the l-th layer before and after firing spikes at time-step t, the neural dynamic can be formulated as follows:

$$\boldsymbol{m}^l(t) = \boldsymbol{v}^l(t-1) + \boldsymbol{W}^l\boldsymbol{x}^{l-1}(t), \tag{9}$$

$$\boldsymbol{s}^l(t) = H(\boldsymbol{m}^l(t) - \theta^l) - H(-\boldsymbol{m}^l(t) + \theta^l) \cdot H(\boldsymbol{c}^l(t) + \theta^l), \tag{10}$$

$$\boldsymbol{x}^l(t) = \theta^l\boldsymbol{s}^l(t), \tag{11}$$

$$\boldsymbol{v}^l(t) = \boldsymbol{m}^l(t) - \boldsymbol{x}^l(t). \tag{12}$$

$$\boldsymbol{c}^l[t] = \boldsymbol{c}^l[t-1] + \boldsymbol{x}^l[t], \tag{13}$$

where H is the Heaviside step function and θ^l is the neuron threshold in layer l. $\boldsymbol{s}^l(t)$ is the output spike of layer l. $\boldsymbol{x}^l(t)$ is the postsynaptic potential and theoretical output of layer l. $\boldsymbol{c}^l[t-1]$ represents an auxiliary cumulative variable used to support the ReLU-like behavior.

4.2 MT neuron dynamics

The MT neuron is characterized by several parameters, including the base threshold θ, and a total of $2n$ thresholds, with n positive and n negative thresholds. The threshold values of

the MT neuron are indexed by i, where λ_i^l represents the i-th threshold value in the layer l:

$$\begin{aligned}\lambda_1^l = \theta^l, \lambda_2^l = \frac{\theta^l}{2}, ..., \lambda_n^l = \frac{\theta^l}{2^{n-1}},\\ \lambda_{n+1}^l = -\theta^l, \lambda_{n+2}^l = -\frac{\theta^l}{2}, ..., \lambda_{2n}^l = -\frac{\theta^l}{2^{n-1}}.\end{aligned} \tag{14}$$

Let variables $\boldsymbol{I}^l[t]$, $\boldsymbol{W}^l$, $\boldsymbol{s}_i^l[t]$, $\boldsymbol{x}^l[t]$, $\boldsymbol{m}^l[t]$, and $\boldsymbol{v}^l[t]$ represent the input current, weight, the output spike of the i-th threshold, the total output signal, and the membrane potential before and after spikes in the l-th layer at the time-step t. It defines $\frac{4}{3}m^l[t] = (-1)^S 2^E(1+M)$ with 1 sign bit (S), 8 exponent bits (E), and 23 mantissa bits (M). Since the median of $\frac{1}{2^{k-1}}$ and $\frac{1}{2^k}$ is $\frac{3}{4}\frac{1}{2^{k-1}}$, we can easily select the correct threshold index i using E and S of $\frac{4}{3}m^l[t]$. The dynamics of the MT neurons are described by the following equations:

$$\boldsymbol{m}^l[t] = \boldsymbol{v}^l[t-1] + \boldsymbol{I}^l[t] = \boldsymbol{v}^l[t-1] + \boldsymbol{x}^{l-1}[t], \tag{15}$$

$$\boldsymbol{s}_i^l[t] = \text{MTH-R}_{\theta,n}(\boldsymbol{m}^l[t], i) \tag{16}$$

$$\boldsymbol{x}^l[t] = \sum_i \boldsymbol{s}_i^l[t]\boldsymbol{W}^l\lambda_i^l, \tag{17}$$

$$\boldsymbol{v}^l[t] = \boldsymbol{m}^l[t] - \boldsymbol{x}^l[t], \tag{18}$$

$$\text{MTH-R}_{\theta,n}(\boldsymbol{m}^l[t], i) = \begin{cases} 1, & \text{if } \begin{cases} i < n, & \text{S} = 0 \text{ and } i - 1 = -\text{E}, \\ i \geq n, & \text{S} = 1 \text{ and} \\ & i - n - 1 = \max(-\text{E}, -\text{E}_2) \end{cases} \\ 0, & \text{otherwise.} \end{cases} \tag{19}$$

$$\tag{20}$$

$$\boldsymbol{c}^l[t] = \boldsymbol{c}^l[t-1] + \boldsymbol{x}^l[t], \tag{21}$$

where $\boldsymbol{c}^l[t-1] = (-1)^{S_2}2^{E_2}(1+M_2)$ represents an auxiliary cumulative variable used to support the ReLU-like behavior.

4.3 DC based neuron dynamics

In rate coding, the output of the previous layer, $\boldsymbol{x}^{l-1}[t]$, is directly used as the input current for the current layer $\boldsymbol{I}^l[t] = \boldsymbol{x}^{l-1}[t]$. In differential coding, the input current $\boldsymbol{I}^l[t]$ can be adjusted as shown in Equation (22), which converts any spiking neuron into a differential spiking neuron:

$$\boldsymbol{I}^l[t] = \boldsymbol{m}_{\boldsymbol{r}}^l[t] + \boldsymbol{x}^{l-1}[t], \tag{22}$$

$$\boldsymbol{m}_{\boldsymbol{r}}^l[t+1] = \boldsymbol{m}_{\boldsymbol{r}}^l[t] + \frac{\boldsymbol{x}^{l-1}[t]}{t} - \frac{\boldsymbol{x}^l[t]}{t}, \tag{23}$$

where $\boldsymbol{m}_{\boldsymbol{r}}^l[0]$ is $\boldsymbol{b}^{l-1}$ if the previous layer has bias else 0. This work employs differential coding methods based on MT neurons.

For linear layers, including fully connected and convolutional layers that can be represented by Equation (24),

$$\boldsymbol{x}^l = \boldsymbol{W}^l\boldsymbol{x}^{l-1} + \boldsymbol{b}^l, \tag{24}$$

where $\boldsymbol{W}^l$ and $\boldsymbol{b}^l$ is the weight and bias of layer l. Under differential coding in SNNs, this is equivalent to eliminating the bias term $\boldsymbol{b}^l$ and initializing the membrane potential of the subsequent layer with the bias value as Equation (25):

$$\boldsymbol{x}^l = \boldsymbol{W}^l\boldsymbol{x}^{l-1}. \tag{25}$$

5 Analyzing the Conversion Errors

This section investigates error propagation in ANN-to-S 7yzNN conversion for continuous control and identifies a phenomenon of error amplification. Section 5.1 decomposes the

performance degradation of converted SNN policies into instantaneous action errors and the resulting state distribution shift, showing that the latter dominates the return loss. Section 5.2 demonstrates that small approximation errors are amplified across decision steps, leading to great state distribution shift. Section 5.3 identifies positive temporal correlations in action errors across consecutive decisions as the underlying cause of this amplification.

5.1 State-Dominated Performance Degradation

Given that ANN-to-SNN conversion exhibits greater performance degradation in continuous control than in widely-studied image classification tasks, we pose a central question: **Is this error solely due to the conversion process, or is it also amplified by the dynamics of RL environments?**

We begin by formalizing the expected return of a policy π:

$$R^{\pi} = \mathbb{E}_{s\sim P_{\pi}(s), a=\pi(s)}\left[r(s,a)\right], \tag{26}$$

where $P_{\pi}(s)$ denotes the marginal state distribution induced by executing policy π in the environment. Accordingly, the expected returns of the original ANN policy and its converted SNN counterpart are given by

$$R^{\mathrm{ANN}} = \mathbb{E}_{s\sim P_{\pi}^{\mathrm{ANN}}(s), a=\pi^{\mathrm{ANN}}(s)}[r(s,a)], \tag{27}$$

$$R^{\mathrm{SNN}} = \mathbb{E}_{s\sim P_{\pi}^{\mathrm{SNN}}(s), a=\pi^{\mathrm{SNN}}(s)}[r(s,a)]. \tag{28}$$

The discrepancy between R^{ANN} and R^{SNN} arises from two sources: (i) differences in action selection induced by the converted policy, i.e., π^{ANN} versus π^{SNN}, and (ii) divergence in the state visitation distributions, i.e., P_{π}^{ANN} versus P_{π}^{SNN}. To disentangle these effects, we define two auxiliary returns:

$$R^{\mathrm{SNN|ANN}} = \mathbb{E}_{s\sim P_{\pi}^{\mathrm{ANN}}(s),\, a=\pi^{\mathrm{SNN}}(s)}[r(s,a)], \tag{29}$$

$$R^{\mathrm{ANN|SNN}} = \mathbb{E}_{s\sim P_{\pi}^{\mathrm{SNN}}(s),\, a=\pi^{\mathrm{ANN}}(s)}[r(s,a)]. \tag{30}$$

The **SNN-action-only return** $R^{\mathrm{SNN|ANN}}$ evaluates the effect of replacing the ANN policy with the converted SNN policy while keeping the ANN-induced state distribution fixed. Conversely, the **SNN-state-only return** $R^{\mathrm{ANN|SNN}}$ isolates the impact of state distribution shift induced by the converted SNN while preserving the original ANN policy.

Figure 2(a) reports the expected returns of the ANN, SNN, SNN-action-only, and SNN-state-only settings. Replacing π^{ANN} with π^{SNN} while maintaining the ANN-induced state distribution results in only negligible performance degradation (less than 0.5%). In contrast, executing either policy under the SNN-induced state distribution leads to a substantial reduction in return. Comprehensive experiments results in Appendix **??** also demonstrates same pattern exists across diverse RL environments and SNN settings. This indicates that the performance degradation is overwhelmingly dominated by deviations in the induced state trajectories rather than instantaneous action mismatches. Furthermore, Fig. 2(b) visualizes the divergence between state trajectories generated by ANN and SNN policies. Despite their close per-step action outputs, the resulting trajectories diverge greatly, highlighting the sensitivity of continuous control systems to small perturbations.

5.2 Error Accumulation and Amplification

Having identified state distribution shift as the primary source of performance degradation, a natural question arises: **how does this shift evolve along a trajectory? Is it uniformly distributed, or does it grow over time?**

In a Markov Decision Process, each action affects the subsequent state, which in turn influences all future transitions. Consequently, small action errors introduced by ANN-to-SNN conversion can propagate across decision steps, causing progressive state deviations and accumulating performance loss.

Figure 3 (a) illustrates how state trajectories induced by ANN and converted SNN policies diverge over time. The discrepancy is small at early stages but grows progressively as

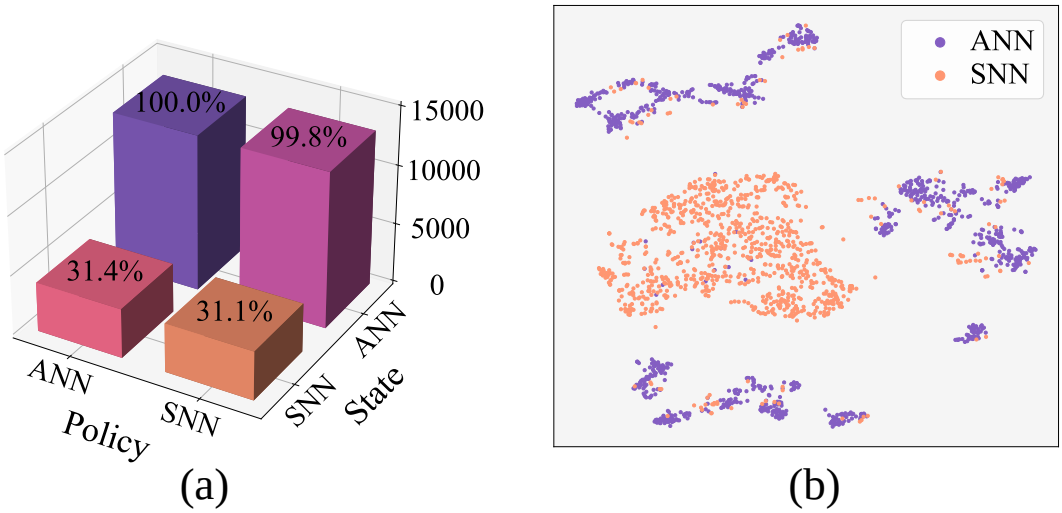


Figure 2: Analysis of performance degradation in ANN-to-SNN conversion in the HalfCheetah-v4 environment. The ANN policy is trained with TD3 for 3 million environment steps and converted using IF neurons with 8 simulation steps. (a) Expected returns under different combinations of policies and state distributions. (b) t-SNE visualization of state trajectories induced by ANN and converted SNN policies, revealing significant distribution divergence.

interaction proceeds, demonstrating that state deviations accumulate and are amplified by the environments. Figure 3 (b) then shows the corresponding impact on returns. While ANN and SNN policies achieve similar rewards at the start of an episode, the gap steadily widens over the decision horizon, reflecting the cumulative effect of state divergence on long-horizon performance.

5.3 Positive Cross-Step Correlation

The analysis in Section 5.2 shows that conversion errors accumulate and amplify over decision steps. A natural question arises: **why do these errors persist instead of being corrected by subsequent actions?**

We analyze the correlation of action approximation errors across consecutive steps. At step k with state s_k, let the actions produced by the ANN and the converted SNN be a_k^{ANN} and a_k^{SNN}, and define the instantaneous action error as $\delta a_k = a_k^{\mathrm{SNN}} - a_k^{\mathrm{ANN}}$. Executing these actions leads to next states s_{k+1}^{ANN} and s_{k+1}^{SNN}. To separate the effect of policy response from state shift, we define a counterfactual action $a_{k+1}^{\mathrm{cf}} = \pi^{\mathrm{ANN}}(s_{k+1}^{\mathrm{SNN}})$, which applies the ANN policy to the SNN-induced next state.

Using these definitions, we compute three cosine similarity metrics that characterize cross-step error propagation:

ANN Correction measures whether the ANN policy compensates for the previous-step action error under the shifted state:

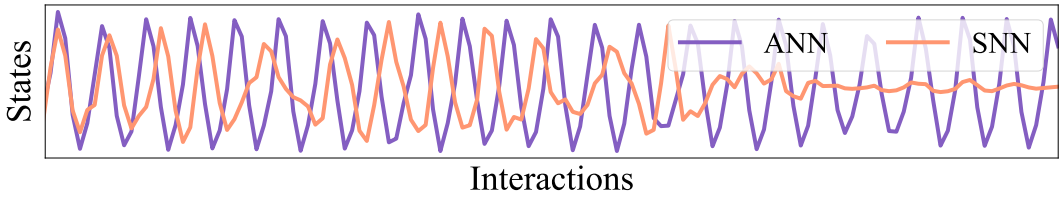
$$\text{ANN Correction} = \cos\big(\delta a_k,\ a_{k+1}^{\mathrm{cf}} - a_{k+1}^{\mathrm{ANN}}\big)\,. \tag{31}$$

SNN Consistency captures whether the SNN exhibits similar action deviations across consecutive steps under the same shifted state:

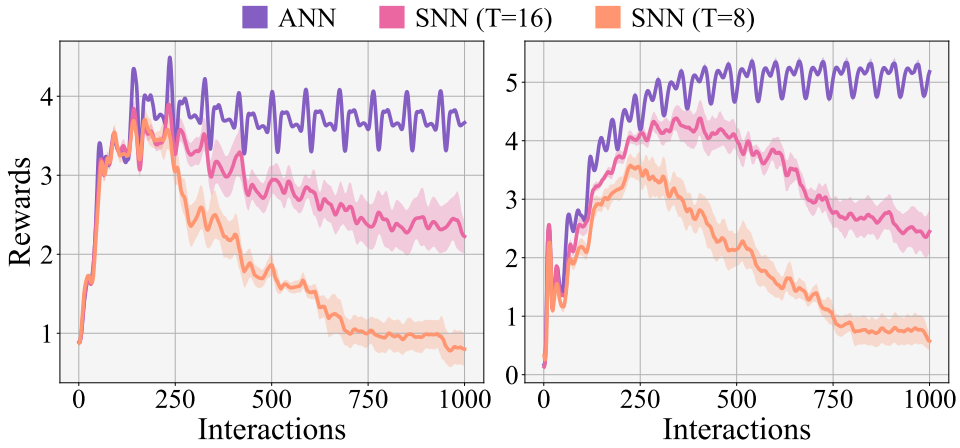
$$\text{SNN Consistency} = \cos\big(\delta a_k,\ a_{k+1}^{\mathrm{SNN}} - a_{k+1}^{\mathrm{cf}}\big)\,. \tag{32}$$

SNN Drift directly quantifies the temporal correlation of action errors between ANN and SNN trajectories:

$$\text{SNN Drift} = \cos\big(\delta a_k,\ a_{k+1}^{\mathrm{SNN}} - a_{k+1}^{\mathrm{ANN}}\big)\,. \tag{33}$$



(a) State evolution over decision steps.



(b) Average reward per decision step.

Figure 3: (a) One-dimensional visualization of state evolution over decision steps for ANN and converted SNN policies in the HalfCheetah-v4 environment, obtained by projecting paired trajectories onto the first principal component via PCA. (b) Average reward per decision step for ANN and converted SNN policies in Hopper-v4 (left) and Walker2d-v4 (right). Shaded regions denote half a standard deviation. All curves are uniformly smoothed for clarity. The ANN was trained with TD3 for 3 million environment steps, and the SNN uses IF neurons.

Table 1: Cosine similarity of action errors across consecutive decision steps in different environments. The ANN is trained with TD3 for 3 million environment steps and the SNN uses IF neurons with 16 simulation steps.

Environment	ANN Correction	SNN Consistency	SNN Drift
Ant-v4	-0.188	0.276	0.030
HalfCheetah-v4	-0.070	0.101	0.043
Hopper-v4	-0.256	0.481	0.129
Walker2d-v4	-0.185	0.462	0.180

Intuitively, negative ANN Correction indicates that the ANN policy actively compensates for previous errors, whereas positive SNN Consistency shows that errors persist across steps. Table 1 confirms this: ANN policies display negative temporal correlations, demonstrating inherent error-correcting behavior, while converted SNNs exhibit positive correlations with respect to the counterfactual ANN action (SNN Consistency), leading to consistent drift (SNN Drift). These results suggest that temporally correlated action errors are the key mechanism behind error accumulation and amplification in ANN-to-SNN conversion.

6 Reducing the Compounding Errors

The analysis in Section 5 shows that the performance degradation of ANN-to-SNN conversion in continuous control is primarily driven by positively correlated action approximation errors across consecutive decision steps. Once such temporal correlations arise, even small conversion errors are repeatedly reinforced by the environment dynamics, inducing progressive state drift and ultimately leading to amplified performance loss. Our objective is therefore to explicitly suppress this cross-step error correlation.

6.1 Deriving the Methods

Motivated by the prior analyses of rate-based ANN-to-SNN conversion Bu et al. [2022b], we assume that the dominant source of action approximation error arises from residual membrane potentials at the end of each decision step k in ANN-to-SNN conversion:

$$\varepsilon_k^l = \frac{\mathbf{v}_k^l[T] - \mathbf{v}_k^l[0]}{T}. \tag{34}$$

We use the temporal correlation of residual membrane potential errors as a tractable proxy for action-level error correlation.

Empirical results in Section 5.3 indicate that these residual errors exhibit positive temporal correlation, i.e., $\mathbb{E}\big[\cos\big(\varepsilon_{k+1}^l, \varepsilon_k^l\big)\big] > 0$, which directly leads to systematic error accumulation across decision steps. Rather than minimizing the magnitude of ε_k^l independently at each step, we instead aim to suppress its temporal correlation. Formally, our goal is to enforce

$$\mathbb{E}\big[\cos\big(\tilde{\varepsilon}_{k+1}^l, \varepsilon_k^l\big)\big] \le 0, \tag{35}$$

where $\tilde{\varepsilon}_{k+1}^l$ denotes the modified residual error after applying a cross-step correction.

To this end, we consider a first-order approximation of residual error dynamics across consecutive decision steps:

$$\tilde{\varepsilon}_{k+1}^l = \varepsilon_{k+1}^l - \alpha\, \varepsilon_k^l, \tag{36}$$

where $\alpha > 0$ captures the empirically observed positive alignment between successive residual errors (as demonstrated in Table 1). Increasing α reduces the expected cosine similarity in Eq. (35) and can drive it below zero, thereby suppressing error accumulation.

Substituting the definition of ε_k^l into Eq. (36) yields

$$\tilde{\varepsilon}_{k+1}^l = \frac{\mathbf{v}_{k+1}^l[T] - \mathbf{v}_{k+1}^l[0]}{T} - \alpha \frac{\mathbf{v}_k^l[T] - \mathbf{v}_k^l[0]}{T} \tag{37}$$

$$= \frac{\mathbf{v}_{k+1}^l[T] - \big(\mathbf{v}_{k+1}^l[0] - \alpha\, \big(\mathbf{v}_k^l[T] - \mathbf{v}_k^l[0]\big)\big)}{T}. \tag{38}$$

Under the standard assumption that the final membrane potential is approximately uniformly distributed in $(0, \theta^l)$ Bu et al. [2022a] and weakly dependent on its initialization, the expected residual error can be reduced by adjusting the initial membrane potential as

$$\mathbf{v}_{k+1}^l[0] \leftarrow \mathbf{v}_{k+1}^l[0] + \alpha\, \big(\mathbf{v}_k^l[T] - \mathbf{v}_k^l[0]\big)\,. \tag{39}$$

6.2 Cross-Step Residual Potential Initialization

We refer the membrane potential initialization mechanism of Equation (39) as Cross-Step Residual Potential Initialization (CRPI). In practice, Eq. (34) assumes non-negative activations induced by ReLU. Therefore, we clip $\mathbf{v}_k^l[T] - \mathbf{v}_0^l[T]$ in Eq. (39) to a minimum of $-\sum_{t=1}^{T} \mathbf{x}_k^l[t]$ to remain consistent with Equitation (6). Moreover, to prevent excessively large residuals from inducing abnormally high initial membrane potentials that may cause persistent bursting across subsequent decision steps, we further clip $\mathbf{v}_{k+1}^l[0]$ to the valid membrane potential range. The complete procedure is summarized in Algorithm 1.

CRPI introduces no additional training or architectural modification and operates solely through membrane potential initialization. It is lightweight and can be readily integrated with existing ANN-to-SNN conversion techniques such as normalization, quantization, and neuron model extensions.

Algorithm 1 Inference with CRPI

Initialize membrane potentials for all layers $\mathbf{v}_0^l[0] \leftarrow \frac{1}{2}\theta^l$
Observe initial environment state s_0
Run SNN for T steps and execute action $a_0 = \pi^{\text{SNN}}(s_0)$
for $k = 1$ to K **do**
  Observe next state s_k
  Compute residual potential from previous step:

$$\Delta\mathbf{v}_k^l \leftarrow \mathbf{v}_{k-1}^l[T] - \mathbf{v}_{k-1}^l[0]$$

  Clip residual to ensure valid firing range:

$$\Delta\mathbf{v}_k^l \leftarrow \max\big(\Delta\mathbf{v}_k^l,\ -\Sigma_{t=1}^{T}\mathbf{x}_{k-1}^l[t]\big)$$

  Initialize membrane potentials with cross-step residual:

$$\mathbf{v}_k^l[0] \leftarrow \text{clip}\big(\tfrac{1}{2}\theta^l + \alpha\Delta\mathbf{v}_k^l, 0,\ \theta^l\big)$$

  Run SNN for T steps and execute $a_k = \pi^{\text{SNN}}(s_k)$
end for

7 Experiments

7.1 Experimental Setup

Environments. We evaluate CRPI on a convincing set of continuous control benchmarks with vision-based observations, for which we evaluate six tasks from the DeepMind Control (DMC) Suite Tunyasuvunakool et al. [2020]: `Cartpole_Swingup`, `Finger_Spin`, `Reacher_Easy`, `Cheetah_Run`, `Acrobot_Swingup`, and `Quadruped_Walk`,which are widely used for benchmarking reinforcement learning algorithms.

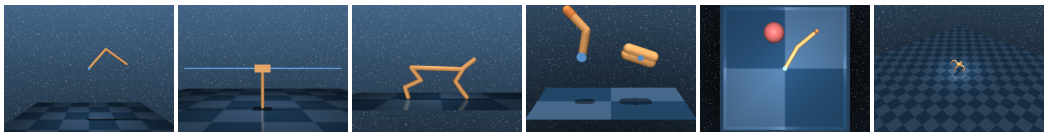


Figure 4: Representative DeepMind Control Suite tasks used in our experiments. From left to right: acrobot_swingup, cartpole_swingup, cheetah_run, finger_spin, reacher_easy, and quadruped_walk.

Table 2: Observation and action space specifications for DeepMind Control Suite environments.

Domain Name	Task Name	Observation	Action Dimension
Acrobot	Swingup	$84 \times 84 \times 3$	1
Cartpole	Swingup	$84 \times 84 \times 3$	1
Finger	Spin	$84 \times 84 \times 3$	2
Reacher	Easy	$84 \times 84 \times 3$	2
Cheetath	Run	$84 \times 84 \times 3$	6
Quadruped	Walk	$84 \times 84 \times 3$	12

DMC tasks rely on pixel observations; in all cases, the action space is continuous.Also they are used with their default parameters provided by the respective simulators. Rewards are not rescaled or normalized during either training or evaluation. For evaluation, each episode is capped at a maximum horizon of 1000 environment interactions, unless terminated earlier by environment-specific conditions. These settings are kept consistent across ANN baselines and converted SNN policies to ensure fair comparison.

RL Algorithms. For image-based continuous control tasks in the DeepMind Control Suite, we adopt Data-Regularized Q-v2 (DrQ-v2) Yarats et al. [2021a,b], a sample-efficient off-policy algorithm designed for high-dimensional visual observations. DrQ-v2 employs a convolutional encoder followed by an actor-critic architecture. Specifically, the visual encoder consists of four convolutional layers with 3×3 kernels and 32 channels, followed by a fully connected layer that produces a compact latent representation of 50 dimensions. Both the actor and critic networks operate on this latent feature and are implemented as two-layer MLPs with 1024 hidden units.

All RL agents are trained using standard hyperparameters and default environment settings. After training, the actor networks are converted to SNNs using different ANN-to-SNN conversion methods. During evaluation, only the converted SNN policies interact with the environment, and no further learning or fine-tuning is performed.

ANN-to-SNN Conversion Methods. We integrate CRPI with multiple ANN-to-SNN conversion techniques. As a baseline, we apply CRPI to standard IF neurons. To assess compatibility with more expressive neuron models, we further combine CRPI with Signed Neuron Models (SNM) Wang et al. [2022a], Multi-Threshold Neurons (MT) Huang et al. [2024], and Differential Coding (DC) Huang et al. [2025]. All MT neurons use four firing thresholds. To avoid additional hyperparameter tuning, the firing threshold θ is set to the maximum ReLU activation channelly.

Evaluation Protocol. All results are averaged over five random seeds. For each seed, we evaluate the policy over ten rollout episodes of up to $1,000$ interaction steps (terminated earlier if the episode ends), yielding a total of $50,000$ environment steps per method. The hyperparameter α is selected via a coarse grid search over $\{0, 0.1, 0.2, \ldots, 0.9, 1.0\}$ and is fixed across all seeds.

7.2 Reducing Error Correlation

To evaluate the effectiveness of CRPI in mitigating the temporally correlated conversion errors identified in Section 5, we first examine how CRPI influences cross-step error correlation. Specifically, we analyze the cosine similarity of residual membrane potential errors across adjacent decision steps, together with the *SNN Consistency* and *SNN Drift* metrics introduced in Section 5.3, which quantify temporal correlation at the action level.

7.3 Enhancing Performance

We next examine how this reduction in temporal error correlation translates into policy performance. Figure 5 illustrates the relationship between the correlation coefficient α and relative performance across tasks. Starting from $\alpha = 0$ (standard ANN-to-SNN conversion), increasing α leads to a steady improvement in performance, demonstrating that incorporating cross-step residual information effectively mitigates temporally correlated conversion errors. However, when α becomes too large, performance begins to degrade. That is because excessive values of α overcompensate residual errors from previous steps, resulting in unstable membrane potential initialization and greater output error. This behavior reveals a clear trade-off between error decorrelation and overcorrection.

Overall, these results show that CRPI prevents small approximation errors from being repeatedly reinforced by closed-loop system dynamics. By suppressing temporal error correlation, CRPI alleviates the error amplification phenomenon analyzed in Section 5.2 and stabilizes long-horizon behavior in continuous control tasks, which we identified as the dominant factor governing performance degradation in Section 5.1.

We conduct comprehensive evaluations on DeepMind Control Suite (DMC) benchmark to assess the effectiveness of CRPI under a wide range of settings, including different neuron models, SNN simulation steps, environments, and underlying RL algorithms. Tables 3 report the performance of the converted SNN policies on DMC tasks. For each configuration, we report the Average Performance Ratio (APR), defined as the mean ratio of SNN performance to the corresponding ANN performance, expressed in percentage and averaged across all evaluated environments.

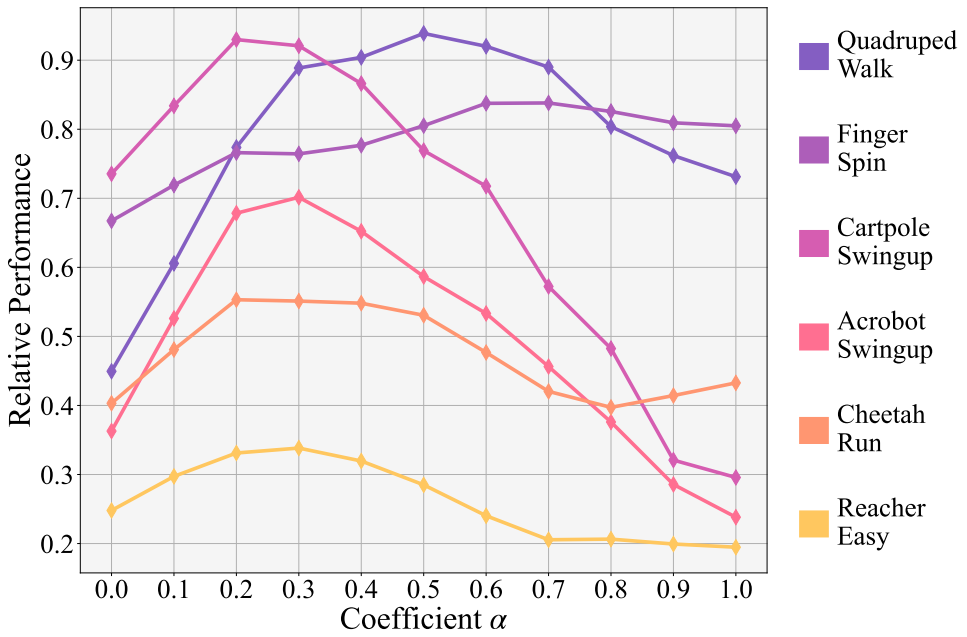


Figure 5: Relative performance on DeepMind Control tasks under different correlation parameter α, using IF neurons with 32 simulation steps. Performance is normalized by the corresponding ANN return. Curves are uniformly smoothed for visualization.

Table 3: Performance comparison on DeepMind Control Suite tasks with visual observations, where ± denotes half an standard deviations.

Module	T	CRPI	Acrobot Swingup	Cartpole Swingup	Cheetah Run	Finger Spin	Quadruped Walk	Reacher Easy	APR
ANN	–	–	225±17	880±0	733±4	976±1	751±9	929±19	100.00%
LIF	8	–	104.0	774.1	515.5	416.2	259.7	403.4	54.19%
TC-LIF	8	–	106.5	667.7	517.8	657.8	302.2	613.2	61.25%
Spiking-WM	8	–	113.7	791.0	577.2	682.0	350.7	701.3	68.54%
IF	32	×	57±9	563±99	240±34	606±39	295±25	197±59	40.77%
		√	167±21	823±8	411±48	831±11	752±22	334±58	74.19%
	64	×	93±12	843±16	494±42	896±12	643±26	331±24	69.60%
		√	223±24	867±1	642±20	940±5	774±21	617±29	91.84%
SNM	8	×	216±19	853±4	612±30	913±9	772±24	551±33	88.72%
		√	248±25	853±4	637±23	933±5	774±18	787±35	96.27%
	16	×	217±21	879±0	714±6	963±6	758±22	907±17	98.50%
		√	237±26	879±0	728±9	972±1	781±6	933±19	101.41%
MT	2	×	210±17	878±0	558±43	952±5	752±29	947±17	94.85%
		√	244±28	878±0	701±7	954±1	783±8	947±17	101.30%
DC	2	×	215±20	877±0	620±5	969±1	749±20	899±17	95.97%
		√	248±29	877±0	660±25	969±1	768±32	939±20	100.43%

Across all evaluated configurations, CRPI consistently improves the performance ratio compared to the corresponding baseline ANN-to-SNN conversion methods. These improvements are observed consistently across different neuron models, simulation lengths, and RL algorithms, indicating that CRPI is robust to both architectural and algorithmic variations.

Table 3 additionally includes comparisons with state-of-the-art directly trained SNNs on visual-based DMC tasks, including approaches incorporating a world model Hafner et al. [2019]. Specifically, we compare against vanilla Leaky Integrate-and-Fire (LIF) neurons, the two-component spiking neuron model (TC-LIF) Zhang et al. [2024], and the spiking world model (Spiking-WM) Sun et al. [2025]. CRPI consistently outperforms these directly trained

Table 4: Average inference-time energy consumption of ANNs and converted SNNs in DeepMind Control suite.

	ANN	IF (T=32)	MT (T=2)
FLOPs	4.53×10^7	–	–
SOPs	–	2.12×10^8	2.71×10^7
Consumptions	566.84 μJ	16.35 μJ	2.09 μJ

SNN approaches, highlighting the advantage of leveraging well-trained ANN policies while effectively mitigating error accumulation and amplification during the conversion process.

7.4 Energy Efficiency

We further analyze the inference-time energy consumption of the converted SNN models. Following the widely adopted estimation framework in Merolla et al. [2014], we approximate energy expenditure by assigning 12.5 pJ per floating-point operation (FLOP) and 77 fJ per synaptic operation (SOP) Qiao et al. [2015], Hu et al. [2021].

As shown in Table 4, ANN baselines incur substantially higher energy consumption per inference compared to their converted SNN counterparts. Despite requiring multiple simulation steps, SNNs achieve significant energy savings due to their sparse, event-driven computation. With advanced multi-threshold neurons, both the number of operations and the energy consumption are further reduced. This result highlights the energy efficiency of SNNs and further supports the suitability of CRPI-converted SNNs for low-power and resource-constrained deployment scenarios.

8 Conclusion

In this work, we conduct a comprehensive investigation into ANN-to-SNN conversion within the context of continuous control, uncovering a fundamental challenge that is largely inconsequential in classification or discrete control: during extended interactions, minute approximation errors develop persistent temporal correlations. This phenomenon acts as a catalyst for a cascading effect, driving a progressive drift in the agent's state distribution and culminating in severe performance deterioration. To counteract this, we have introduced Cross-Step Residual Potential Initialization (CRPI), a training-free inference protocol that explicitly targets and disrupts this cross-step error correlation. Our experiments confirm that CRPI is broadly compatible with a variety of neuron models and conversion methodologies, delivering consistent performance gains on the DeepMind Control benchmark suite without compromising the inherent energy efficiency of SNNs. These results firmly establish continuous control as an essential and demanding benchmark for the future development and evaluation of ANN-to-SNN conversion techniques, as it uniquely exposes how trivial approximation errors can be catastrophically amplified over time, leading to significant functional failure.

References

Wolfgang Maass. Networks of spiking neurons: the third generation of neural network models. *Neural Networks*, 10(9):1659–1671, 1997.

Wulfram Gerstner, Werner M Kistler, Richard Naud, and Liam Paninski. *Neuronal dynamics: From single neurons to networks and models of cognition*. Cambridge University Press, 2014.

Paul A Merolla, John V Arthur, Rodrigo Alvarez-Icaza, Andrew S Cassidy, Jun Sawada, Filipp Akopyan, Bryan L Jackson, Nabil Imam, Chen Guo, Yutaka Nakamura, et al. A million spiking-neuron integrated circuit with a scalable communication network and interface. *Science*, 345(6197):668–673, 2014.

Mike Davies, Narayan Srinivasa, Tsung-Han Lin, Gautham Chinya, Yongqiang Cao, Sri Harsha Choday, Georgios Dimou, Prasad Joshi, Nabil Imam, Shweta Jain, et al. Loihi: A neuromorphic manycore processor with on-chip learning. *IEEE Micro*, 2018.

Michael V DeBole, Brian Taba, Arnon Amir, Filipp Akopyan, Alexander Andreopoulos, William P Risk, Jeff Kusnitz, Carlos Ortega Otero, Tapan K Nayak, Rathinakumar Appuswamy, et al. TrueNorth: Accelerating from zero to 64 million neurons in 10 years. *Computer*, 2019.

Yongqiang Cao, Yang Chen, and Deepak Khosla. Spiking deep convolutional neural networks for energy-efficient object recognition. *International Journal of Computer Vision*, 2015.

Bing Han, Gopalakrishnan Srinivasan, and Kaushik Roy. Rmp-snn: Residual membrane potential neuron for enabling deeper high-accuracy and low-latency spiking neural network. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 13558–13567, 2020.

Yuhang Li, Shikuang Deng, Xin Dong, Ruihao Gong, and Shi Gu. A free lunch from ann: Towards efficient, accurate spiking neural networks calibration. In *International conference on machine learning*, pages 6316–6325. PMLR, 2021.

Shikuang Deng and Shi Gu. Optimal conversion of conventional artificial neural networks to spiking neural networks. *arXiv preprint arXiv:2103.00476*, 2021.

Tong Bu, Jianhao Ding, Zhaofei Yu, and Tiejun Huang. Optimized potential initialization for low-latency spiking neural networks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 36, 2022a.

Tong Bu, Maohua Li, and Zhaofei Yu. Inference-scale complexity in ANN-SNN conversion for high-performance and low-power applications. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 24387–24397, 2025.

Minqi Jiang, Tim Rocktäschel, and Edward Grefenstette. General intelligence requires rethinking exploration. *Royal Society Open Science*, 10(6):230539, 2023.

Abhishek Padalkar, Gabriel Quere, Antonin Raffin, João Silvério, and Freek Stulp. Guiding real-world reinforcement learning for in-contact manipulation tasks with shared control templates. *Autonomous Robots*, 48(4):12, 2024.

Chen Tang, Ben Abbatematteo, Jiaheng Hu, Rohan Chandra, Roberto Martín-Martín, and Peter Stone. Deep reinforcement learning for robotics: A survey of real-world successes. *Annual Review of Control, Robotics, and Autonomous Systems*, 8(1):153–188, 2025.

Ashish K Jayant and Shalabh Bhatnagar. Model-based safe deep reinforcement learning via a constrained proximal policy optimization algorithm. *Advances in Neural Information Processing Systems*, 35:24432–24445, 2022.

Zihan Huang, Wei Fang, Tong Bu, Peng Xue, Zecheng Hao, Wenxuan Liu, Yuanhong Tang, Zhaofei Yu, and Tiejun Huang. Differential coding for training-free ann-to-snn conversion. *arXiv preprint arXiv:2503.00301*, 2025.

Devdhar Patel, Hananel Hazan, Daniel J Saunders, Hava T Siegelmann, and Robert Kozma. Improved robustness of reinforcement learning policies upon conversion to spiking neuronal network platforms applied to atari breakout game. *Neural Networks*, 120:108–115, 2019.

Weihao Tan, Devdhar Patel, and Robert Kozma. Strategy and benchmark for converting deep q-networks to event-driven spiking neural networks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pages 9816–9824, 2021.

Aakash Kumar, Lei Zhang, Hazrat Bilal, Shifeng Wang, Ali Muhammad Shaikh, Lu Bo, Avinash Rohra, and Alisha Khalid. Dsqn: Robust path planning of mobile robot based on deep spiking q-network. *Neurocomputing*, 634:129916, 2025.

Shuo Feng, Jian Cao, Zehong Ou, Guang Chen, Yi Zhong, Zilin Wang, Juntong Yan, Jue Chen, Bingsen Wang, Chenglong Zou, et al. Brainqn: Enhancing the robustness of deep reinforcement learning with spiking neural networks. *Advanced Intelligent Systems*, 6(9): 2400075, 2024.

Jens Kober, J Andrew Bagnell, and Jan Peters. Reinforcement learning in robotics: A survey. *The International Journal of Robotics Research*, 32(11):1238–1274, 2013.

Shixiang Gu, Ethan Holly, Timothy Lillicrap, and Sergey Levine. Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates. In *2017 IEEE International Conference on Robotics and Automation*, pages 3389–3396. IEEE, 2017.

Lukas Brunke, Melissa Greeff, Adam W Hall, Zhaocong Yuan, Siqi Zhou, Jacopo Panerati, and Angela P Schoellig. Safe learning in robotics: From learning-based control to safe reinforcement learning. *Annual Review of Control, Robotics, and Autonomous Systems*, 5 (1):411–444, 2022.

Saran Tunyasuvunakool, Alistair Muldal, Yotam Doron, Siqi Liu, Steven Bohez, Josh Merel, Tom Erez, Timothy Lillicrap, Nicolas Heess, and Yuval Tassa. dm_control: Software and tasks for continuous control. *Software Impacts*, 6:100022, 2020.

Bodo Rueckauer, Iulia-Alexandra Lungu, Yuhuang Hu, Michael Pfeiffer, and Shih-Chii Liu. Conversion of continuous-valued deep networks to efficient event-driven networks for image classification. *Frontiers in neuroscience*, 11:682, 2017.

Tong Bu, Wei Fang, Jianhao Ding, PengLin Dai, Zhaofei Yu, and Tiejun Huang. Optimal ann-snn conversion for high-accuracy and ultra-low-latency spiking neural networks. *arXiv preprint arXiv:2303.04347*, 2023.

Yangfan Hu, Qian Zheng, Xudong Jiang, and Gang Pan. Fast-snn: Fast spiking neural network by converting quantized ann. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(12):14546–14562, 2023.

Zecheng Hao, Tong Bu, Jianhao Ding, Tiejun Huang, and Zhaofei Yu. Reducing ann-snn conversion error through residual membrane potential. In *Proceedings of the AAAI conference on artificial intelligence*, volume 37, pages 11–21, 2023a.

Zecheng Hao, Jianhao Ding, Tong Bu, Tiejun Huang, and Zhaofei Yu. Bridging the gap between anns and snns by calibrating offset spikes. *arXiv preprint arXiv:2302.10685*, 2023b.

Yuchen Wang, Malu Zhang, Yi Chen, and Hong Qu. Signed neuron with memory: Towards simple, accurate and high-efficient ann-snn conversion. In *IJCAI*, pages 2501–2508, 2022a.

Chen Li, Lei Ma, and Steve Furber. Quantization framework for fast spiking neural networks. *Frontiers in Neuroscience*, 16:918793, 2022.

Zihan Huang, Xinyu Shi, Zecheng Hao, Tong Bu, Jianhao Ding, Zhaofei Yu, and Tiejun Huang. Towards high-performance spiking transformers from ann to snn conversion. In *Proceedings of the 32nd ACM international conference on multimedia*, pages 10688–10697, 2024.

Bodo Rueckauer and Shih-Chii Liu. Conversion of analog to spiking neural networks using sparse temporal coding. In *2018 IEEE international symposium on circuits and systems (ISCAS)*, pages 1–5. IEEE, 2018.

Lei Zhang, Shengyuan Zhou, Tian Zhi, Zidong Du, and Yunji Chen. Tdsnn: From deep neural networks to deep spike neural networks with temporal-coding. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 1319–1326, 2019.

Ana Stanojevic, Stanisław Woźniak, Guillaume Bellec, Giovanni Cherubini, Angeliki Pantazi, and Wulfram Gerstner. An exact mapping from relu networks to spiking neural networks. *Neural Networks*, 168:74–88, 2023.

Jaehyun Kim, Heesu Kim, Subin Huh, Jinho Lee, and Kiyoung Choi. Deep neural networks with weighted spikes. *Neurocomputing*, 311:373–386, 2018.

Zhehui Wang, Xiaozhe Gu, Rick Siow Mong Goh, Joey Tianyi Zhou, and Tao Luo. Efficient spiking neural networks with radix encoding. *IEEE Transactions on Neural Networks and Learning Systems*, 35(3):3689–3701, 2022b.

Seongsik Park, Seijoon Kim, Hyeokjun Choe, and Sungroh Yoon. Fast and efficient information transmission with burst spikes in deep spiking neural networks. In *Proceedings of the 56th Annual Design Automation Conference 2019*, pages 1–6, 2019.

Yang Li and Yi Zeng. Efficient and accurate conversion of spiking neural network with burst spikes. *arXiv preprint arXiv:2204.13271*, 2022.

Ziqing Wang, Yuetong Fang, Jiahang Cao, Hongwei Ren, and Renjing Xu. Adaptive calibration: A unified conversion framework of spiking neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 1583–1591, 2025.

Hyunseok Oh and Youngki Lee. Sign gradient descent-based neuronal dynamics: Ann-to-snn conversion beyond relu network. *arXiv preprint arXiv:2407.01645*, 2024.

Yizhou Jiang, Kunlin Hu, Tianren Zhang, Haichuan Gao, Yuqian Liu, Ying Fang, and Feng Chen. Spatio-temporal approximation: A training-free snn conversion for transformers. In *The twelfth international conference on learning representations*, 2024.

Ziqing Wang, Yuetong Fang, Jiahang Cao, Qiang Zhang, Zhongrui Wang, and Renjing Xu. Masked spiking transformer. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 1761–1771, 2023.

Kang You, Zekai Xu, Chen Nie, Zhijie Deng, Qinghai Guo, Xiang Wang, and Zhezhi He. Spikezip-tf: Conversion is all you need for transformer-based snn. *arXiv preprint arXiv:2406.03470*, 2024.

Răzvan V Florian. Reinforcement learning through modulation of spike-timing-dependent synaptic plasticity. *Neural Computation*, 19(6):1468–1502, 2007.

Nicolas Frémaux and Wulfram Gerstner. Neuromodulated spike-timing-dependent plasticity, and theory of three-factor learning rules. *Frontiers in Neural Circuits*, 9:85, 2016.

Wulfram Gerstner, Marco Lehmann, Vasiliki Liakoni, Dane Corneil, and Johanni Brea. Eligibility traces and plasticity on behavioral time scales: experimental support of neohebbian three-factor learning rules. *Frontiers in Neural Circuits*, 12:53, 2018.

Nicolas Frémaux, Henning Sprekeler, and Wulfram Gerstner. Reinforcement learning using a continuous time actor-critic framework with spiking neurons. *PLoS Computational Biology*, 9(4):e1003024, 2013.

Zhile Yang, Shangqi Guo, Ying Fang, Zhaofei Yu, and Jian K Liu. Spiking variational policy gradient for brain inspired reinforcement learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024.

Yujie Wu, Lei Deng, Guoqi Li, Jun Zhu, and Luping Shi. Spatio-temporal backpropagation for training high-performance spiking neural networks. *Frontiers in Neuroscience*, 12:331, 2018.

Guisong Liu, Wenjie Deng, Xiurui Xie, Li Huang, and Huajin Tang. Human-level control through directly trained deep spiking q-networks. *IEEE Transactions on Cybernetics*, 53 (11):7187–7198, 2022.

Ding Chen, Peixi Peng, Tiejun Huang, and Yonghong Tian. Deep reinforcement learning with spiking q-learning. *arXiv preprint arXiv:2201.09754*, 2022.

Lang Qin, Rui Yan, and Huajin Tang. A low latency adaptive coding spiking framework for deep reinforcement learning. *arXiv preprint arXiv:2211.11760*, 2022.

Yinqian Sun, Yi Zeng, and Yang Li. Solving the spike feature information vanishing problem in spiking deep q network with potential based normalization. *Frontiers in Neuroscience*, 16:953368, 2022.

Guillaume Bellec, Franz Scherr, Anand Subramoney, Elias Hajek, Darjan Salaj, Robert Legenstein, and Wolfgang Maass. A solution to the learning dilemma for recurrent networks of spiking neurons. *Nature Communications*, 11(1):3625, 2020.

Zijie Xu, Xinyu Shi, Yiting Dong, Zihan Huang, and Zhaofei Yu. Care-bn: Precise moving statistics for stabilizing spiking neural networks in reinforcement learning. *arXiv preprint arXiv:2509.23791*, 2025a.

Guangzhi Tang, Neelesh Kumar, and Konstantinos P Michmizos. Reinforcement co-learning of deep and spiking neural networks for energy-efficient mapless navigation with neuromorphic hardware. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 6090–6097. IEEE, 2020.

Guangzhi Tang, Neelesh Kumar, Raymond Yoo, and Konstantinos Michmizos. Deep reinforcement learning with population-coded spiking neural network for continuous control. In *Conference on Robot Learning*, pages 2016–2029. PMLR, 2021.

Duzhen Zhang, Tielin Zhang, Shuncheng Jia, and Bo Xu. Multi-sacle dynamic coding improved spiking actor network for reinforcement learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 36, pages 59–67, 2022.

Ding Chen, Peixi Peng, Tiejun Huang, and Yonghong Tian. Fully spiking actor network with intralayer connections for reinforcement learning. *IEEE Transactions on Neural Networks and Learning Systems*, 36(2):2881–2893, 2024.

Jianchuan Ding, Bo Dong, Felix Heide, Yufei Ding, Yunduo Zhou, Baocai Yin, and Xin Yang. Biologically inspired dynamic thresholds for spiking neural networks. *Advances in Neural Information Processing Systems*, 35:6090–6103, 2022.

Zijie Xu, Tong Bu, Zecheng Hao, Jianhao Ding, and Zhaofei Yu. Proxy target: Bridging the gap between discrete spiking neural networks and continuous control. *arXiv preprint arXiv:2505.24161*, 2025b.

Volodymyr Mnih. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.

Tong Bu, Wei Fang, Jianhao Ding, PengLin Dai, Zhaofei Yu, and Tiejun Huang. Optimal ANN-SNN conversion for high-accuracy and ultra-low-latency spiking neural networks. In *International Conference on Learning Representations*, 2022b.

Denis Yarats, Ilya Kostrikov, and Rob Fergus. Image augmentation is all you need: Regularizing deep reinforcement learning from pixels. In *International Conference on Learning Representations*, 2021a. URL `https://openreview.net/forum?id=GY6-6sTvGaf`.

Denis Yarats, Rob Fergus, Alessandro Lazaric, and Lerrel Pinto. Mastering visual continuous control: Improved data-augmented reinforcement learning. *arXiv preprint arXiv:2107.09645*, 2021b.

Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to control: Learning behaviors by latent imagination. *arXiv preprint arXiv:1912.01603*, 2019.

Shimin Zhang, Qu Yang, Chenxiang Ma, Jibin Wu, Haizhou Li, and Kay Chen Tan. Tc-lif: A two-compartment spiking neuron model for long-term sequential modelling. In *Proceedings of the AAAI conference on artificial intelligence*, volume 38, pages 16838–16847, 2024.

Yinqian Sun, Feifei Zhao, Mingyang Lyu, and Yi Zeng. Spiking world model with multicompartment neurons for model-based reinforcement learning. *Proceedings of the National Academy of Sciences*, 122(50):e2513319122, 2025.

Ning Qiao, Hesham Mostafa, Federico Corradi, Marc Osswald, Fabio Stefanini, Dora Sumislawska, and Giacomo Indiveri. A reconfigurable on-line learning spiking neuromorphic processor comprising 256 neurons and 128K synapses. *Frontiers in Neuroscience*, 9:141, 2015.

Yangfan Hu, Huajin Tang, and Gang Pan. Spiking deep residual networks. *IEEE Transactions on Neural Networks and Learning Systems*, 34(8):5200–5205, 2021.