COMMON-MODE ERRORS LIMIT LOW-TIMESTEP DEEP SPIKING Q-NETWORKS

Anonymous authors
Paper under double-blind review

ABSTRACT

Spiking neural networks (SNNs) offer sparse and event-driven computation, making them attractive for energy-constrained reinforcement learning (RL) on edge devices. In value-based RL, deep spiking Q-networks (DSQNs) combine such efficiency with action-value estimation for decision making. However, existing DSQNs often require multiple simulation timesteps for competitive performance, increasing computational and energy costs, whereas reducing the timesteps can cause substantial performance degradation. We investigate this degradation from the perspective of Q-value estimation errors. By decomposing errors across actions into common-mode and differential-mode components, we find that low-timestep DSQNs suffer disproportionately from common-mode errors shared across action values, which are particularly detrimental to temporal-difference learning through bootstrapped targets. Based on this finding, we propose Common-Mode Compensation Deep Spiking Q-Network (CMC-DSQN), which uses an auxiliary ANN to compensate for common-mode errors in the SNN outputs. At inference, greedy action selection can be performed directly from the SNN outputs, allowing the auxiliary ANN to be completely removed and preserving the energy efficiency of SNNs. Extensive experiments on Atari and MiniAtar environments demonstrate substantial performance improvements under low-timestep settings. CMC-DSQN outperforms state-of-the-art DSQN baselines by nearly 20% at $T = 2$ and further surpasses the ANN baseline at $T = 4$.

1 INTRODUCTION

Spiking neural networks (SNNs) process information through sparse binary spikes, providing an event-driven alternative to conventional artificial neural networks (ANNs) (Maass, 1997; Gerstner et al., 2014). When deployed on neuromorphic hardware, such sparse computation can substantially reduce the computational cost and inference latency (Merolla et al., 2014; Davies et al., 2018; DeBole et al., 2019). These properties are particularly attractive for reinforcement learning (RL), where agents repeatedly interact with the environment and continuously make decisions. Combining SNNs with RL therefore provides a promising route toward energy-efficient sequential decision making on resource-constrained edge devices (Yamazaki et al., 2022).

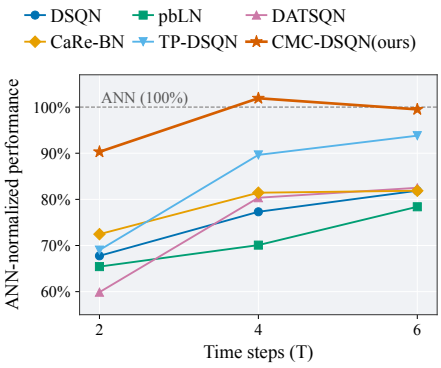


Figure 1: Comparison of existing directly trained DSQNs on the MiniAtar environments (Young & Tian, 2019), including DSQN (Liu et al., 2022), pbLN (Sun et al., 2022), DATSQN (Ghoreishee et al., 2026), CaRe-BN (Xu et al., 2026c), and TP-DSQN (Chowdhury et al., 2022). Performance is averaged across all environments and normalized by the ANN.

RL has achieved remarkable success in sequential decision-making tasks (Mnih et al., 2015; Schulman et al., 2017; Lillicrap et al., 2015; Haarnoja et al., 2018). Among different RL paradigms, value-based methods learn action values that estimate the expected long-term returns of candidate actions. Deep Q-Networks (DQNs) combine Q-learning with ANNs for function approximation (Mnih et al., 2013; 2015), achieving human-level

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

control on Atari games and establishing a fundamental framework for value-based RL (Mnih et al., 2015; Van Hasselt et al., 2016; Schaul et al., 2015; Hessel et al., 2018). However, conventional DQNs require dense ANN computation to estimate Q-values at every decision step. Replacing the ANN-based Q-function approximator with an SNN gives rise to Deep Spiking Q-Networks (DSQNs), which aim to improve the energy efficiency of value-based RL.

DSQNs have been developed through both ANN-to-SNN conversion (Patel et al., 2019; Tan et al., 2021; Kumar et al., 2025) and direct gradient-based training (Liu et al., 2022; Qin et al., 2022; 2025; Sun et al., 2022; Chen et al., 2022; Ghoreishee et al., 2026), achieving competitive performance or even surpassing ANNs. However, such performance often relies on a relatively large number of SNN simulation timesteps. For each RL decision, a DSQN must simulate its neuronal dynamics for T internal timesteps before outputting Q-values. A larger T therefore increases inference latency and computational cost. Although reducing T is highly desirable, we find that low-timestep DSQNs suffer from substantial performance degradation.

In this work, we investigate this degradation from the perspective of Q-value estimation errors. We decompose the errors across actions into *common-mode* and *differential-mode* components. While ANN Q-networks exhibit relatively balanced errors between the two modes, low-timestep DSQNs are dominated by common-mode errors. We further show theoretically that common-mode errors affect temporal-difference learning more directly than differential-mode errors. Experiments on the CliffWalking environment (Sutton & Barto, 2018) confirm their greater impact, showing that removing common-mode errors is more effective than removing differential-mode errors. Together, these results identify common-mode error as a major bottleneck of low-timestep DSQNs.

Motivated by these findings, we propose *Common-Mode Compensation Deep Spiking Q-Network* (CMC-DSQN). To suppress common-mode errors in the low-timestep SNN, CMC-DSQN replaces the action-wise mean of the SNN outputs with a common-mode estimate produced by an auxiliary ANN. The SNN and ANN are jointly optimized through temporal-difference learning, allowing the ANN to provide a more reliable estimate of the common-mode component. Since this common-mode correction does not change the greedy action, the auxiliary ANN is needed only during training and can be removed entirely at inference. CMC-DSQN therefore improves Q-value estimation without introducing additional computational overheads during deployment.

We evaluate CMC-DSQN on both Atari (Mnih et al., 2013) and MiniAtar (Young & Tian, 2019) environments. Under low-timestep settings, CMC-DSQN consistently achieves strong performance and substantially outperforms both directly trained DSQNs and ANN-to-SNN conversion baselines. As shown in Fig. 1, at $T = 2$, CMC-DSQN exceeds state-of-the-art DSQN baselines by nearly 20%. With $T = 4$, CMC-DSQN further surpasses the ANN baseline on both Atari and MiniAtar environments. These results demonstrate that mitigating common-mode estimation errors is particularly effective for closing the performance gap of low-timestep DSQNs.

2 Related Work

2.1 Spiking Neural Networks for Policy-Based Reinforcement Learning

SNNs have been applied to RL through biologically motivated synaptic plasticity (Florian, 2007; Frémaux & Gerstner, 2016; Gerstner et al., 2018; Frémaux et al., 2013; Yang et al., 2024) and gradient-based learning (Bellec et al., 2020). For continuous control, numerous studies have developed spiking policies through ANN-to-SNN conversion (Xu et al., 2026b) or actor–critic learning (Tang et al., 2020; 2021; Zhang et al., 2022; Chen et al., 2024; Ding et al., 2022; Van den Berghe et al., 2026; Xu et al., 2026a;c). These studies primarily focus on policy learning, whereas our work investigates value-based RL through DSQNs.

2.2 Spiking Neural Networks for Value-Based Reinforcement Learning

For value-based RL, SNNs have been integrated into Deep Q-Networks through both ANN-to-SNN conversion and direct training. Conversion-based methods transform pretrained DQNs into spiking networks (Patel et al., 2019; Tan et al., 2021; Kumar et al., 2025), whereas directly trained DSQNs use surrogate gradients such as STBP (Wu et al., 2018) to optimize spiking Q-functions end-to-end (Liu et al., 2022; Qin et al., 2022). Subsequent work has improved DSQNs through recurrent

mechanisms (Qin et al., 2025), normalization (Sun et al., 2022; Xu et al., 2026c), and enhanced spiking-neuron designs (Chen et al., 2022; Ghoreishee et al., 2026). However, Q-value estimation errors in low-timestep DSQNs remain insufficiently understood.

2.3 Value Decomposition in Deep Q-Learning

Value decomposition exploits structures shared across actions. Dueling architectures and their extensions decompose Q-values into a state-dependent shared component and action-dependent advantages (Wang et al., 2016; Hessel et al., 2018; Tavakoli et al., 2018), while VA-learning further studies value–advantage learning under bootstrapping and its relation to the behavior policy (Tang et al., 2023). In contrast to these methods that decompose the Q-function itself, we decompose its estimation errors into common-mode and differential-mode components, and show that common-mode errors dominate in low-timestep DSQNs. Beyond this distinction, CMC-DSQN adopts a hybrid architecture in which a training-only auxiliary ANN estimates the common-mode component without sharing the SNN encoder, differing from typical dueling-style shared-encoder architectures.

3 Preliminaries

3.1 Deep Q-Learning

Reinforcement learning (RL) considers a Markov decision process (MDP) $\langle \mathcal{S}, \mathcal{A}, P, R, \gamma \rangle$ (Sutton & Barto, 2018), where $\mathcal{S}$ and $\mathcal{A}$ denote the state and action spaces, $P(s'|s,a)$ is the transition probability, $R(s,a)$ is the reward function, and $\gamma \in [0,1)$ is the discount factor. At interaction step i, the agent observes state s_i, selects action a_i, receives reward r_i, and transitions to s_{i+1}. The discounted return G_i and the action-value function $Q^\pi(s,a)$ under policy π are defined as

$$G_i = \sum_{k=0}^{\infty} \gamma^k r_{i+k}, \tag{1}$$

$$Q^\pi(s,a) = \mathbb{E}_\pi \left[G_i \mid s_i = s, a_i = a \right]. \tag{2}$$

The objective is to learn the optimal action-value function Q^*, which satisfies the Bellman equation

$$Q^*(s_i,a_i) = \mathbb{E}\left[r_i + \gamma \max_{a'} Q^*(s_{i+1},a') \right]. \tag{3}$$

Deep Q-Networks (DQNs) (Mnih et al., 2013; 2015) approximate Q^* using a neural network $Q_\theta(s,a)$ parameterized by θ. Given a transition $(s_i, a_i, r_i, s_{i+1}, d_i)$, the network is optimized by minimizing the temporal-difference (TD) loss

$$\mathcal{L} = \mathbb{E}\left[\left(Q_\theta(s_i,a_i) - y_i \right)^2 \right], \tag{4}$$

$$y_i = r_i + \gamma(1-d_i) \max_{a'} Q_\theta(s_{i+1},a'), \tag{5}$$

where y_i is the bootstrapped TD target and $d_i \in \{0,1\}$ indicates whether the transition terminates the episode. During inference, the agent follows the greedy policy $a_i = \arg\max_{a \in \mathcal{A}} Q_\theta(s_i,a)$. Because the TD target depends on estimated Q-values at the next state, estimation errors can propagate through successive bootstrapped updates, which motivates our analysis in Sec. 4.

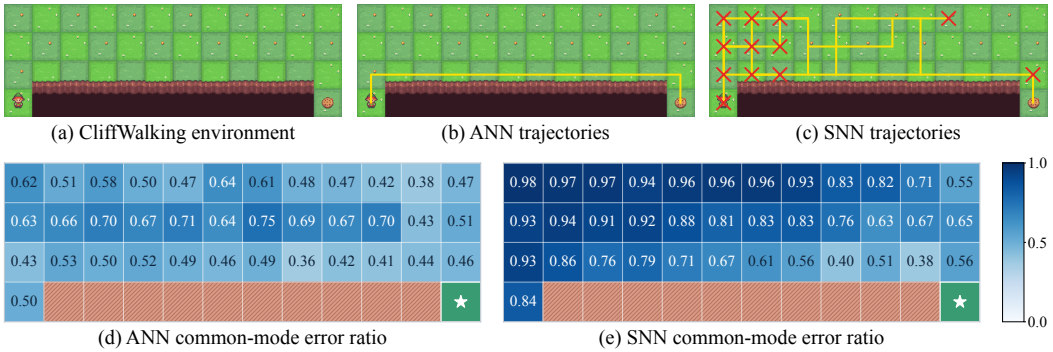
3.2 Spiking Neural Networks

Spiking neural networks (SNNs) process information through discrete spikes over multiple simulation timesteps. We adopt the commonly used leaky integrate-and-fire (LIF) neuron with hard reset (Gerstner & Kistler, 2002). At SNN timestep t, its dynamics are

$$\tilde{u}^t = \lambda u^{t-1} + I^t, \qquad o^t = H(\tilde{u}^t - v_{\mathrm{th}}), \qquad u^t = \tilde{u}^t(1-o^t), \tag{6}$$

where $\tilde{u}^t$ and u^t denote the membrane potentials before and after reset, λ is the membrane decay factor, I^t is the synaptic input, v_{th} is the firing threshold, and $H(\cdot)$ denotes the Heaviside function.

A deep spiking Q-network (DSQN) replaces the ANN-based Q-network with an SNN. For each RL observation, the SNN runs for T simulation timesteps to output the action values. A larger T generally improves value representation at higher computational cost, whereas a smaller T makes accurate Q-value estimation more challenging.



(a) CliffWalking environment (b) ANN trajectories (c) SNN trajectories

(d) ANN common-mode error ratio (e) SNN common-mode error ratio

Figure 2: Analysis on deterministic CliffWalking. (a) Environment. (b,c) Greedy trajectories of the ANN and low-timestep SNN, respectively, aggregated over 20 random seeds. Yellow segments denote trajectories, and red crosses mark states leading to infinite loops. (d,e) State-wise common-mode error ratios of the ANN and SNN with respect to the optimal action-value function Q^*.

4 Common-Mode Errors in Deep Spiking Q-Networks

We begin with a case study on the deterministic CliffWalking environment shown in Fig. 2(a), where all transitions are deterministic and entering a cliff cell terminates the episode. Its small state space allows both the optimal action-value function Q^* and the action-value function Q^π of any policy to be computed exactly from the Bellman equations (Bellman, 1966). This enables direct analysis of Q-value estimation errors and their effects on TD bootstrapping (Sutton, 1988).

4.1 Common- and Differential-Mode Error Decomposition

Consider a discrete action space $\mathcal{A}$ with $|\mathcal{A}| = K$. Given a reference action-value function Q^{ref}, we decompose the estimation error $e(s,a) = \hat{Q}(s,a) - Q^{\text{ref}}(s,a)$ across actions as

$$e_c(s) = \frac{1}{K} \sum_{a \in \mathcal{A}} e(s,a), \tag{7}$$

$$e_d(s,a) = e(s,a) - e_c(s), \tag{8}$$

where e_c is the *common-mode error*, representing the error shared across actions, and e_d is the *differential-mode error*, capturing errors in the relative differences among actions.

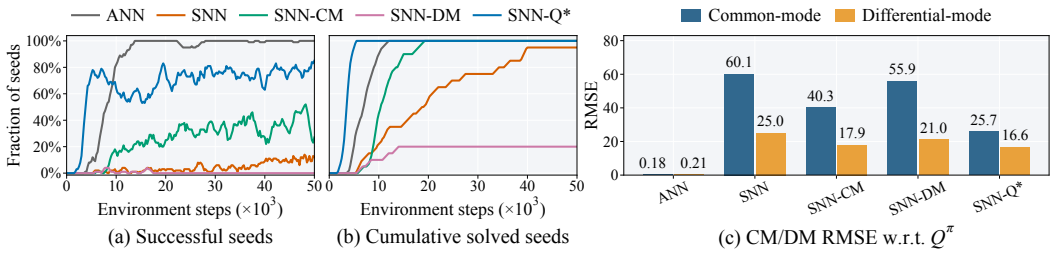
We further define the common-mode error ratio $\rho_c(s)$ to quantify the relative magnitude of the common-mode error compared with the two error components at each state:

$$\rho_c(s) = \frac{E_c(s)}{E_c(s) + E_d(s)}, \qquad E_c(s) = |e_c(s)|, \qquad E_d(s) = \sqrt{\frac{1}{K} \sum_a e_d(s,a)^2}. \tag{9}$$

4.2 Common-Mode Error Dominance in Low-Timestep SNNs

We first compare a conventional ANN with a low-timestep SNN ($T = 2$) under the same DQN framework over 20 random seeds. Because both the environment and the greedy evaluation policy are deterministic, each seed either succeeds or fails at each evaluation point. As shown in Figs. 3(a) and (b), the ANN rapidly learns a stable solution, whereas the SNN is substantially less stable. Although many SNN seeds eventually discover a successful policy, only a small fraction remain successful at any given evaluation point, indicating difficulty in maintaining stable value estimates.

Common-mode error dominance. To separate estimation error from policy suboptimality, Fig. 3(c) measures the common- and differential-mode RMSEs with respect to the exact Q^π of each learned greedy policy. The SNN exhibits substantially larger errors than the ANN in both modes. More importantly, its common-mode error is much larger than its differential-mode error, whereas the



(a) Successful seeds (b) Cumulative solved seeds (c) CM/DM RMSE w.r.t. Q^{π}

Figure 3: Learning dynamics and Q-value estimation errors on CliffWalking environment. (a) Fraction of successful seeds at each evaluation point. (b) Cumulative fraction of seeds that have solved the task at least once. Curves in (a) and (b) are smoothed for visual clarity. (c) Common- and differential-mode RMSEs with respect to the exact Q^{π} of each learned deterministic greedy policy.

two are relatively balanced in the ANN. Thus, low-timestep SNNs exhibit not only larger Q-value estimation errors, but also a common-mode dominance.

State-wise common-mode dominance. We further examine the spatial distribution of these errors using the optimal action-value function Q^* as the reference. Figures 2(d) and (e) show the state-wise common-mode error ratio $\rho_c(s)$ for the ANN and SNN, respectively. The ANN maintains a relatively balanced error composition, whereas the SNN is dominated by common-mode errors at nearly every state, particularly in the upper-left region. This region also coincides with the cyclic trajectories frequently observed in unsuccessful SNN policies (Fig. 2(c)), suggesting a close association between common-mode error dominance and unstable policy.

4.3 Error Propagation through Temporal-Difference Bootstrapping

To determine whether common-mode errors contribute to performance degradation, we next analyze how common-mode and differential-mode errors propagate through TD bootstrapping. Consider the one-step target in Eq. (5) for a non-terminal transition ($d_i = 0$). Given a reference Q-function Q^{ref}, let $\hat{Q}(s,a) = Q^{\mathrm{ref}}(s,a) + e_c(s) + e_d(s,a)$. Substituting this decomposition into the TD target gives

$$\begin{cases} \hat{y}_i = r_i + \gamma e_c(s_{i+1}) + \gamma \max_a \left[Q^{\mathrm{ref}}(s_{i+1},a) + e_d(s_{i+1},a) \right], \\ y_i^{\mathrm{ref}} = r_i + \gamma \max_a Q^{\mathrm{ref}}(s_{i+1},a). \end{cases} \tag{10}$$

Thus, the error propagated to the bootstrapped target is:

$$\hat{y}_i - y_i^{\mathrm{ref}} = \gamma e_c(s_{i+1}) + \gamma \Delta_d(s_{i+1}). \tag{11}$$

$$\Delta_d(s) = \max_a \left[Q^{\mathrm{ref}}(s,a) + e_d(s,a) \right] - \max_a Q^{\mathrm{ref}}(s,a). \tag{12}$$

The common-mode error shifts all action values equally and therefore enters the bootstrap target directly, whereas the differential-mode error affects it only through maximization, with $|\Delta_d(s)| \le \max_a |e_d(s,a)|$. Common-mode errors thus have a more direct pathway into successive TD updates.

Removing Bootstrap Errors. To isolate the effects of the two modes, we modify only the next-state Q-values used for bootstrapping:

$$\begin{cases} \hat{Q}_{\mathrm{CM}}(s,a) = \hat{Q}(s,a) - e_c^{\pi}(s), \\ \hat{Q}_{\mathrm{DM}}(s,a) = \hat{Q}(s,a) - e_d^{\pi}(s,a), \\ \hat{Q}_{Q^*}(s,a) = Q^*(s,a), \end{cases} \tag{13}$$

where e_c^{π} and e_d^{π} are computed with respect to the exact Q^{π} of the current learned greedy policy. These variants are denoted as SNN-CM, SNN-DM, and SNN-Q*, respectively.

Learning dynamics. As shown in Figs. 3(a) and (b), replacing the next-state estimate with exact Q^* substantially improves learning, confirming the importance of bootstrap Q-value accuracy.

Algorithm 1 CMC-DSQN

Training:
for each sampled transition (s_i, a_i, r_i, s_{i+1}) **do**
  Compute $Q^{\mathrm{CMC}}_{\phi,\theta}(s_i, \cdot)$ and $Q^{\mathrm{CMC}}_{\phi,\theta}(s_{i+1}, \cdot)$ using Eq. (16)
  Construct the TD target y_i using Eq. (5)
  Update θ and ϕ by minimizing $\mathcal{L} = (Q^{\mathrm{CMC}}_{\phi,\theta}(s_i, a_i) - y_i)^2$
end for
Inference:
Compute only $Q^{\mathrm{SNN}}_{\theta}(s_i, \cdot)$
Select $a_i = \arg\max_a Q^{\mathrm{SNN}}_{\theta}(s_i, a)$

Removing only the common-mode error also yields a clear improvement, whereas removing the differential-mode error does not improve learning[1]. This indicates that common-mode errors play a more critical role than differential-mode errors in destabilizing TD learning in low-timestep SNNs.

Q-value estimation errors. The Q-value estimation errors in Fig. 3 (c) further support this observation. Using the exact optimal Q-value Q^* achieves the lowest estimation errors. Removing either the common-mode or differential-mode error reduces both error components, but correcting the common mode leads to a relatively larger reduction. Together, these results show that common-mode errors have a substantially greater impact on TD learning than differential-mode errors.

5 Common-Mode Compensation for Deep Spiking Q-Networks

The analysis in Sec. 4 shows that common-mode estimation errors can directly bias bootstrapped TD targets and degrade learning in low-timestep DSQNs. In the simple CliffWalking environment, such errors can be explicitly removed because exact reference Q-functions are available. In practical RL tasks, however, neither Q^π nor Q^* is accessible during training.

A straightforward alternative is to introduce an auxiliary ANN to estimate and compensate for the entire Q-value estimation error. However, this requires the auxiliary network to model both common- and differential-mode components, introducing unnecessary complexity. As shown in Sec. 6.3, this design improves over the vanilla DSQN but underperforms CMC-DSQN.

5.1 CMC-DSQN

Motivated by the analysis in Sec. 4, we propose *Common-Mode Compensation Deep Spiking Q-Network* (CMC-DSQN), which introduces an auxiliary ANN $C^{\mathrm{ANN}}_{\phi}(s)$ parameterized by ϕ to estimate the common-mode component.

Let $Q^{\mathrm{SNN}}_{\theta}(s, a)$ denote the raw Q-value produced by the SNN with parameters θ. Its common-mode estimation error $e_c(s)$ with respect to a reference Q-function Q^{ref} is

$$e_c(s) = \frac{1}{K} \sum_{a \in \mathcal{A}} \left(Q^{\mathrm{SNN}}_{\theta}(s, a) - Q^{\mathrm{ref}}(s, a) \right). \tag{14}$$

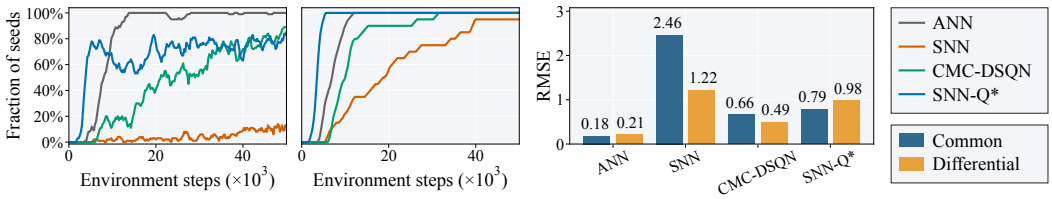
Directly predicting $e_c(s)$ is undesirable because the SNN evolves continuously during training, making the residual error itself a moving target. Instead, the auxiliary ANN directly estimates the corresponding common-mode Q-value component,

$$C^{\mathrm{ANN}}_{\phi}(s) \approx \frac{1}{K} \sum_{a \in \mathcal{A}} Q^{\mathrm{ref}}(s, a). \tag{15}$$

Note that Q^{ref} is unavailable during training. Thus, the auxiliary ANN is learned jointly with the SNN solely through TD learning by constructing the CMC Q-value as

$$Q^{\mathrm{CMC}}_{\phi,\theta}(s, a) = Q^{\mathrm{SNN}}_{\theta}(s, a) - \frac{1}{K} \sum_{a' \in \mathcal{A}} Q^{\mathrm{SNN}}_{\theta}(s, a') + C^{\mathrm{ANN}}_{\phi}(s). \tag{16}$$

[1] One possible explanation is that Δ_d can occasionally offset part of the common-mode bias; removing it may eliminate this incidental cancellation.



(a) Successful seeds (b) Cumulative solved seeds (c) CM/DM RMSE w.r.t. Q*

Figure 4: Learning dynamics and Q-value estimation errors on CliffWalking. (a) Fraction of successful seeds at each evaluation point. (b) Cumulative fraction of seeds that have solved the task at least once. Curves in (a) and (b) are smoothed for visual clarity. (c) Common- and differential-mode RMSEs with respect to the exact optimal Q-function Q^*. Unlike Fig. 3(c), which uses the exact Q^π of each learned policy as the reference, this panel evaluates all methods against Q^*.

Thus, CMC-DSQN removes the common component of the SNN outputs and replaces it with the estimate produced by the auxiliary ANN. The ANN and SNN process the observation independently without sharing an encoder, and are jointly optimized through the DQN objective in Eq. (4). The overall training and inference procedures of CMC-DSQN are summarized in Algorithm 1.

An important property of Eq. (16) is that the auxiliary ANN does not affect greedy action selection:

$$\arg\max_a Q^{\mathrm{CMC}}_{\phi,\theta}(s,a) = \arg\max_a Q^{\mathrm{SNN}}_{\theta}(s,a). \tag{17}$$

Therefore, the auxiliary ANN is required only during training and can be removed entirely at inference, where actions are selected directly from the original SNN outputs. CMC-DSQN therefore introduces no additional computational overhead and preserves the energy efficiency of SNNs.

5.2 Understanding Common-Mode Compensation

We revisit CliffWalking to examine whether CMC-DSQN alleviates the error characteristics identified in Sec. 4. As shown in Figs. 4(a) and (b), CMC-DSQN substantially improves learning stability over the vanilla SNN and enables nearly all seeds to solve the task. Figure 4(c) further shows that CMC-DSQN markedly reduces both common- and differential-mode Q-value errors.

Interestingly, CMC-DSQN achieves even lower estimation errors than SNN-Q*, despite the latter using the exact optimal Q-values for bootstrapping. SNN-Q* still requires a single low-timestep SNN to reduce both common- and differential-mode errors, whereas CMC-DSQN assigns the common-mode error reduction to the ANN and allows the SNN to focus on reducing differential-mode errors. This reduces the representational burden on the SNN and improves estimation of both error modes.

6 Experiments

6.1 Experimental Setup

We evaluate CMC-DSQN on two commonly used benchmarks: Atari (Mnih et al., 2013) and MiniAtar (Young & Tian, 2019). For Atari, we consider seven games evaluated in the original DQN study (Mnih et al., 2013): BeamRider, Breakout, Enduro, Pong, Q*bert, Seaquest, and SpaceInvaders. For MiniAtar, we evaluate on all five environments: Asterix, Breakout, Freeway, Seaquest, and SpaceInvaders. We use $T \in \{2,3,4\}$ for Atari and $T \in \{2,4,6\}$ for MiniAtar. All models are trained for 30M environment transitions on Atari and 5M transitions on MiniAtar. Results are averaged over three random seeds for Atari and five random seeds for MiniAtar. The main hyperparameters follow Mnih et al. (2015) and Young & Tian (2019), respectively. Detailed implementation settings are provided in the Appendix.

We compare CMC-DSQN with both ANN-to-SNN conversion methods and directly trained DSQNs. The conversion baselines include vanilla ANN-to-SNN conversion with integrate-and-fire (IF) neurons (DSQN-CVT), differential coding with multi-threshold neurons (DCGS) (Huang et al., 2025; 2024), and cross-step residual potential initialization (CRPI) (Xu et al., 2026b) combined with

Table 1: Average maximum return over three random seeds and average performance ratio (APR) on Atari environments after 30M environment transitions.

Timesteps	Method	BeamRider	Breakout	Enduro	Pong	Q*bert	Seaquest	SpaceInvaders	APR
–	Random	354.64	0.97	0.00	-20.26	162.08	60.20	127.83	0.00%
	DQN(ANN)	6842.99	364.51	295.79	19.86	9224.17	1960.44	1492.50	100.00%
$T = 2$	DSQN	**4279.90**	276.11	280.97	17.60	1300.21	2101.49	806.36	70.75%
	CMC-DSQN	4136.44	**382.90**	**316.87**	**18.34**	**4186.33**	**2358.57**	**1183.40**	**87.06%**
$T = 3$	DSQN	4705.58	292.41	273.83	17.23	4076.25	**2863.83**	935.96	83.32%
	CMC-DSQN	**5284.37**	**398.73**	**313.55**	**19.59**	**8782.83**	1766.93	**1401.10**	**95.57%**
$T = 4$	DSQN	4676.82	318.01	266.15	16.13	4306.67	**3589.00**	883.21	88.76%
	CMC-DSQN	**5904.69**	**411.70**	**311.73**	**19.98**	**8941.17**	2399.73	**1491.77**	**103.45%**

Table 2: Average maximum return over five random seeds and average performance ratio (APR) on MiniAtar environments after 5M environment transitions, where $\pm$ denotes one standard deviation.

Timesteps	methods	Asterix	Breakout	Freeway	Seaquest	SpaceInvaders	APR
—	Random	0.493 ± 0.902	0.366 ± 0.618	0.370 ± 0.586	0.116 ± 0.427	4.057 ± 3.199	0.00%
	DQN (ANN)	32.22 ± 4.94	27.76 ± 4.98	62.58 ± 0.34	49.70 ± 8.15	138.34 ± 3.87	100.00%
$T = 2$	DSQN-CVT	1.89 ± 1.14	4.11 ± 2.40	19.17 ± 17.72	1.36 ± 0.75	3.82 ± 3.77	10.12%
	DCGS	18.36 ± 3.55	11.64 ± 2.21	59.22 ± 0.77	26.17 ± 2.84	60.56 ± 20.89	57.33%
	CRPI	21.02 ± 2.36	14.92 ± 0.89	59.70 ± 0.64	28.56 ± 2.44	82.02 ± 21.38	65.72%
	DSQN	14.50 ± 3.04	23.90 ± 1.22	56.92 ± 1.06	22.84 ± 15.67	100.76 ± 11.51	67.76%
	pbLN	16.20 ± 1.40	26.43 ± 2.56	55.57 ± 0.35	13.87 ± 2.69	92.70 ± 1.87	65.43%
	DATSQN	14.67 ± 1.29	23.93 ± 3.37	58.47 ± 0.84	2.33 ± 0.21	99.17 ± 19.28	59.88%
	CaRe-BN	17.46 ± 1.99	21.38 ± 2.13	58.04 ± 0.29	34.76 ± 5.13	97.42 ± 4.23	72.46%
	TP-DSQN	19.06 ± 4.40	23.34 ± 2.54	58.38 ± 0.77	30.38 ± 10.84	69.02 ± 14.37	69.01%
	CMC-DSQN	27.30 ± 4.94	23.50 ± 5.09	56.74 ± 1.12	41.06 ± 14.90	150.92 ± 36.31	**90.30%**
$T = 4$	DSQN-CVT	5.56 ± 3.22	6.67 ± 2.07	50.97 ± 3.63	3.98 ± 2.01	14.29 ± 9.35	27.15%
	DCGS	21.65 ± 6.96	13.83 ± 3.96	60.77 ± 0.70	33.43 ± 6.41	78.50 ± 17.75	67.11%
	CRPI	26.42 ± 3.65	17.87 ± 2.36	61.28 ± 0.76	43.06 ± 8.84	102.76 ± 17.66	80.72%
	DSQN	22.04 ± 5.54	22.96 ± 4.56	57.04 ± 1.10	33.10 ± 6.95	109.50 ± 15.99	77.31%
	pbLN	15.70 ± 0.36	25.03 ± 2.23	55.57 ± 0.67	23.20 ± 2.46	107.67 ± 2.64	70.08%
	DATSQN	26.96 ± 3.70	25.06 ± 1.99	58.76 ± 3.13	35.96 ± 14.92	87.38 ± 11.38	80.35%
	CaRe-BN	19.82 ± 2.98	27.92 ± 3.20	58.56 ± 0.70	31.88 ± 5.39	122.38 ± 10.21	81.44%
	TP-DSQN	28.74 ± 4.34	26.66 ± 2.31	60.62 ± 0.51	45.04 ± 6.24	105.72 ± 14.27	89.64%
	CMC-DSQN	35.40 ± 2.36	26.92 ± 4.34	59.78 ± 0.37	55.78 ± 8.78	131.44 ± 13.50	**101.92%**
$T = 6$	DSQN-CVT	7.60 ± 2.82	9.15 ± 2.72	56.46 ± 2.09	8.27 ± 2.56	19.27 ± 12.49	34.48%
	DCGS	23.56 ± 4.84	15.68 ± 3.88	60.33 ± 1.14	38.57 ± 7.76	89.98 ± 18.49	73.30%
	CRPI	25.65 ± 3.72	18.41 ± 2.17	61.08 ± 0.95	47.63 ± 10.11	111.13 ± 16.11	83.66%
	DSQN	20.98 ± 1.98	28.50 ± 2.19	57.04 ± 0.82	33.98 ± 10.87	115.14 ± 11.39	81.88%
	pbLN	18.70 ± 3.42	29.23 ± 2.49	56.07 ± 0.25	33.00 ± 8.51	102.57 ± 11.42	78.39%
	DATSQN	29.20 ± 3.37	23.24 ± 2.93	57.00 ± 6.31	38.56 ± 12.73	98.04 ± 16.59	82.51%
	CaRe-BN	20.43 ± 1.63	26.33 ± 1.22	58.10 ± 0.53	32.47 ± 2.76	129.83 ± 17.06	81.87%
	TP-DSQN	33.62 ± 5.33	24.90 ± 2.35	61.20 ± 0.82	46.74 ± 13.35	115.82 ± 24.09	93.80%
	CMC-DSQN	35.44 ± 3.63	26.88 ± 2.75	59.38 ± 0.50	50.12 ± 10.59	131.48 ± 13.94	**99.51%**

DCGS. The directly trained baselines include vanilla DSQN (Liu et al., 2022), potential-based layer normalization (pbLN) (Sun et al., 2022), deep asymmetric ternary spiking Q-network (DATSQN) (Ghoreishee et al., 2026), confidence-adaptive and re-calibration batch normalization (CaReBN) (Xu et al., 2026c), and DSQN with temporal pruning (TP-DSQN) (Chowdhury et al., 2022).

In addition to raw scores, we report the average performance ratio (APR):

$$\mathrm{APR}_m = \frac{1}{|\mathcal{E}|} \sum_{e \in \mathcal{E}} \frac{S_{m,e} - S_{\mathrm{Random},e}}{S_{\mathrm{ANN},e} - S_{\mathrm{Random},e}} \times 100\%, \tag{18}$$

where $\mathcal{E}$ denotes the set of environments and $S_{m,e}$ is the evaluation score of method m on environment e. An APR of 100% corresponds to the ANN baseline (DQN) on average.

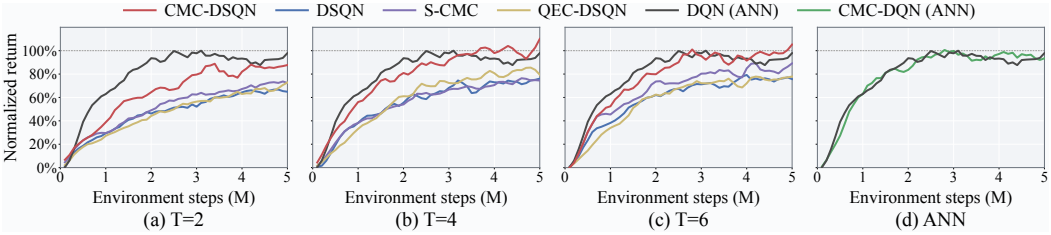


Figure 5: Average normalized return across MiniAtar environments. (a)–(c) Comparison of different DSQN variants at $T = 2, 4, 6$, respectively. (d) Comparison of the vanilla ANN DQN and its common-mode compensated variant. All curves are smoothed uniformly for visual clarity.

6.2 Main Results

As shown in Table 1, CMC-DSQN consistently improves the aggregated performance of low-timestep DSQNs across the seven Atari games. The improvement is particularly evident at $T = 2$, where CMC-DSQN increases the average performance ratio (APR) by more than 16 percentage points over the vanilla DSQN. The advantage persists as the simulation horizon increases.

Similar improvements are observed on MiniAtar, where we further compare CMC-DSQN with recent directly trained DSQNs and ANN-to-SNN conversion methods, as summarized in Table 2 and Fig. 1. CMC-DSQN consistently outperforms both categories of baselines across different simulation timesteps. At $T = 2$, CMC-DSQN reaches an APR of 90.30%, nearly 20 percentage points higher than CaRe-BN (Xu et al., 2026c), the strongest competing DSQN baseline.

Overall, CMC-DSQN achieves average normalized performance exceeding the ANN baseline at $T = 4$ on both Atari and MiniAtar. These results show that common-mode compensation effectively alleviates the performance degradation caused by reducing the number of SNN simulation timesteps.

6.3 Ablation Studies

Is the improvement simply due to additional parameters? We replace the auxiliary ANN with an SNN, denoted as S-CMC. Figures 5(a)–(c) show that S-CMC consistently underperforms CMC-DSQN across different T, indicating that the improvement is not due to introducing additional trainable parameters and highlighting the importance of reliable common-mode compensation.

Why compensate only the common mode? We further consider using the auxiliary ANN to compensate the Q-value error rather than only its common-mode component, denoted as QEC-DSQN. Figures 5(a)–(c) show that QEC-DSQN improves over the vanilla DSQN but consistently underperforms CMC-DSQN, validating the design choice of compensating only the common-mode errors.

Does common-mode compensation also benefit ANN-based DQNs? We apply the same compensation mechanism to an ANN-based DQN, denoted as CMC-DQN. Figure 5(d) shows that CMC-DQN closely matches the vanilla DQN with little additional benefit. This is consistent with Sec. 4, where common-mode error dominance is observed in low-timestep SNNs but not in ANNs. This suggests that the improvement of CMC-DSQN primarily addresses an SNN-specific estimation problem rather than arising from a generic improvement to the underlying RL algorithm.

7 Conclusion

We investigated the performance degradation of low-timestep DSQNs from the perspective of Q-value estimation errors and found that common-mode errors dominate and affect TD bootstrapping more strongly than differential-mode errors. Based on this finding, we proposed Common-Mode Compensation Deep Spiking Q-Network (CMC-DSQN), which replaces the SNN common-mode component with an estimate from an auxiliary ANN. Since the auxiliary ANN can be removed entirely at inference, CMC-DSQN preserves the energy-efficiency advantages of SNNs. Experiments on Atari and MiniAtar environments show that CMC-DSQN substantially outperforms state-of-the-art DSQN baselines in low-timestep settings and surpasses the ANN baseline at $T = 4$.

AI USE STATEMENT

We used generative AI tools only to assist with manuscript writing and code editing. For manuscript writing, generative AI was used to identify and correct spelling and grammatical errors and to improve expressions that were unclear, unnatural, or inconsistent with academic writing conventions. The research content, technical claims, and scientific conclusions were determined by the authors. For code development, the overall structure, algorithmic logic, and initial implementation of all experimental code were first designed and written by us. Generative AI tools were subsequently used to assist with code editing and programming. All AI-assisted code was manually inspected and cross-checked by two authors to ensure consistency with the intended algorithms and experimental settings.

All experimental data collection, processing, organization, and analysis were performed manually by us. Generative AI tools can not access or alter any experimental data or reported results. Research problem formulation, methodological development, theoretical analysis, experimental design, literature review, result interpretation, figure generation, and scientific conclusions were carried out by the authors **without** the use of generative AI.

We have reviewed all AI-assisted content and take full responsibility for the final content of this work, including its text, code, results, claims, and other artifacts.

ETHICS STATEMENT

This work studies learning algorithms for spiking neural networks using simulated reinforcement learning benchmarks. The experiments **do not** involve human participants, animal subjects, personally identifiable information, or private datasets. The proposed method is intended to improve the performance and computational efficiency of value-based spiking agents, with potential benefits for energy-efficient neuromorphic computing. We are not aware of any specific ethical concerns in this study.

REPRODUCIBILITY STATEMENT

We provide detailed information to facilitate reproduction of our results. The exact experimental configurations and evaluation protocols are also documented in the Appendix. Source code is provided in the supplementary materials to reproduce the reported experiments.

REFERENCES

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.

M. G. Bellemare, Y. Naddaf, J. Veness, and M. Bowling. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47:253–279, jun 2013.

Richard Bellman. Dynamic programming. *Science*, 153(3731):34–37, 1966.

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

Sayeed Shafayet Chowdhury, Nitin Rathi, and Kaushik Roy. Towards ultra low latency spiking neural networks for vision and sequential tasks using temporal pruning. In *European Conference on Computer Vision*, pp. 709–726. Springer, 2022.

Steven Dalton and iuri frosio. Accelerating reinforcement learning through gpu atari emulation. *Advances in Neural Information Processing Systems*, 33:19773–19782, 2020.

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.

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.

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.

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.

Wulfram Gerstner and Werner M Kistler. *Spiking neuron models: Single neurons, populations, plasticity*. Cambridge University Press, 2002.

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.

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.

Aref Ghoreishee, Abhishek Mishra, John MacLaren Walsh, Anup Das, and Nagarajan Kandasamy. Improving performance of spike-based deep q-learning using ternary neurons. In *Proceedings of the International Conference on Neuromorphic Systems*, pp. 44–51, 2026.

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

Matteo Hessel, Joseph Modayil, Hado Van Hasselt, Tom Schaul, Georg Ostrovski, Will Dabney, Dan Horgan, Bilal Piot, Mohammad Azar, and David Silver. Rainbow: Combining improvements in deep reinforcement learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

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*, pp. 10688–10697, 2024.

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.

Sotetsu Koyamada, Shinri Okano, Soichiro Nishimori, Yu Murata, Keigo Habara, Haruka Kita, and Shin Ishii. Pgx: Hardware-accelerated parallel game simulators for reinforcement learning. *Advances in Neural Information Processing Systems*, 36:45716–45743, 2023.

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.

Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

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.

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

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.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. 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.

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.

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

Lang Qin, Ziming Wang, Runhao Jiang, Rui Yan, and Huajin Tang. Grsn: Gated recurrent spiking neurons for pomdps and marl. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 1483–1491, 2025.

Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. Prioritized experience replay. *arXiv preprint arXiv:1511.05952*, 2015.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *ArXiv*, abs/1707.06347, 2017.

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.

Richard S Sutton. Learning to predict by the methods of temporal differences. *Machine Learning*, 3:9–44, 1988.

Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT Press, 2018.

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, pp. 9816–9824, 2021.

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*, pp. 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*, pp. 2016–2029. PMLR, 2021.

Yunhao Tang, Rémi Munos, Mark Rowland, and Michal Valko. Va-learning as a more efficient alternative to q-learning. In *International Conference on Machine Learning*, pp. 33739–33757. PMLR, 2023.

Arash Tavakoli, Fabio Pardo, and Petar Kormushev. Action branching architectures for deep reinforcement learning. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence and Thirtieth Innovative Applications of Artificial Intelligence Conference and Eighth AAAI Symposium on Educational Advances in Artificial Intelligence*, pp. 4131–4138, 2018.

Korneel Van den Berghe, Stein Stroobants, Vijay Janapa Reddi, and Guido De Croon. Adaptive surrogate gradients for sequential reinforcement learning in spiking neural networks. *Advances in Neural Information Processing Systems*, 38:147904–147926, 2026.

Hado Van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 30, 2016.

Ziyu Wang, Tom Schaul, Matteo Hessel, Hado Van Hasselt, Marc Lanctot, and Nando De Freitas. Dueling network architectures for deep reinforcement learning. In *Proceedings of the 33rd International Conference on Machine Learning*, pp. 1995–2003, 2016.

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.

Zijie Xu, Tong Bu, Zecheng Hao, Jianhao Ding, and Zhaofei Yu. Proxy target: Bridging the gap between discrete spiking neural networks and continuous control. *Advances in Neural Information Processing Systems*, 38:159158–159184, 2026a.

Zijie Xu, Zihan Huang, Yiting Dong, Kang Chen, Wenxuan Liu, and Zhaofei Yu. Error amplification limits ann-to-snn conversion in continuous control. *arXiv preprint arXiv:2601.21778*, 2026b.

Zijie Xu, Xinyu Shi, Yiting Dong, Zihan Huang, and Zhaofei Yu. CaRe-BN: Precise moving statistics for stabilizing spiking neural networks in reinforcement learning. In *The Fourteenth International Conference on Learning Representations*, 2026c.

Kashu Yamazaki, Viet-Khoa Vo-Ho, Darshan Bulsara, and Ngan Le. Spiking neural networks and their applications: A review. *Brain Sciences*, 12(7):863, 2022.

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.

Kenny Young and Tian Tian. Minatar: An atari-inspired testbed for thorough and reproducible reinforcement learning experiments. *arXiv preprint arXiv:1903.03176*, 2019.

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, pp. 59–67, 2022.

A Deep Spiking Q-Network Architectures

A.1 Encoding

For each RL observation, the DSQN is simulated for T internal timesteps. We adopt direct deterministic encoding. The same observation is presented to the SNN at every simulation timestep. Thus, for an input observation $\mathbf{x}$,

$$\mathbf{x}^t = \mathbf{x}, \qquad t = 1, \ldots, T. \tag{19}$$

This design avoids additional randomness introduced by Poisson sampling and is particularly suitable for the low-timestep setting considered in this work.

For MiniAtar, the observations are already binary and are therefore directly used as spike-compatible inputs. For CliffWalking, the discrete state index is first transformed into a one-hot vector.

A.2 Decoding

The last spiking hidden layer is connected to $K = |\mathcal{A}|$ non-firing, non-leaky readout neurons, with one neuron corresponding to each action. These neurons neither emit spikes nor undergo membrane leakage or reset, but instead integrate the incoming spikes throughout the T simulation timesteps:

$$\mathbf{u}_{\text{out}}^t = \mathbf{u}_{\text{out}}^{t-1} + \mathbf{W}_{\text{out}}\mathbf{s}^t + \mathbf{b}_{\text{out}}. \tag{20}$$

The final membrane potentials are directly used as the continuous Q-value estimates,

$$Q(s, a_k) = u_{\text{out},k}^T. \tag{21}$$

A.3 Training with Surrogate Gradient

The SNN in DSQN is trained using spatio-temporal backpropagation (STBP) (Wu et al., 2018). We employ a rectangular surrogate gradient during the backward pass:

$$\Theta'(x) = \begin{cases} \frac{1}{2\omega}, & -\omega \le x \le \omega, \\ 0, & \text{otherwise}, \end{cases} \tag{22}$$

where ω denotes the half-width of the surrogate window.

B Experiment Details

B.1 Environments

CliffWalking. We use the `CliffWalking-v0` environment from Gymnasium as a case study. The environment consists of a 4×12 grid with 48 discrete states and four actions. Each state is represented by a 48-dimensional one-hot vector before being fed into the networks. The transition dynamics are deterministic. Following the setting used in our analysis, reaching the goal terminates the episode, and stepping into a cliff cell receives a reward of -100 and is also treated as terminal. All other transitions receive a reward of -1.

Atari. We evaluate image-based control on Atari games (Bellemare et al., 2013), as illustrated in Fig. 6. We use CuLE (Dalton & frosio, 2020) for parallel environment simulation. Each observation consists of four consecutive grayscale frames resized to 84×84, resulting in an input of size $4 \times 84 \times 84$. Each selected action is repeated for four emulator frames. Pixel values are converted to floating-point values and normalized to $[0, 1]$ before being passed to the network. During training, rewards are clipped by their signs, and life loss is treated as terminal for TD bootstrapping; evaluation instead uses the original unclipped rewards and full-game episodes. It is worth noting that Breakout contains substantially sparser visual observations than the other Atari environments considered in our experiments. To increase the effective input magnitude under the low-timestep spiking representation, we multiply the normalized Breakout observations by a factor of 4. All other Atari environments use the original normalized observations.

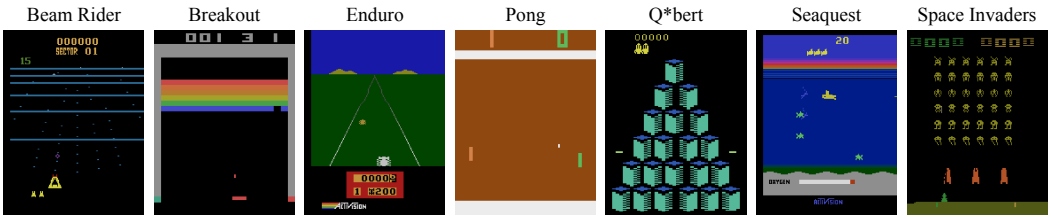


Figure 6: Atari environments (Bellemare et al., 2013) used in our experiments.

MiniAtar. We further evaluate CMC-DSQN on the five MiniAtar games (Young & Tian, 2019) Asterix, Breakout, Freeway, Seaquest, and Space Invaders, visualized in Fig. 7. MiniAtar represents each state using binary feature planes of spatial size 10×10, with the number of channels determined by the game. We directly use these native binary observations without frame stacking, resizing, or intensity normalization. Experiments are parallelized using PGX (Koyamada et al., 2023). We retain the default sticky-action probability of 0.1 and difficulty ramping, and no additional action repeat is applied. Native environment rewards are used without clipping or reward shaping.

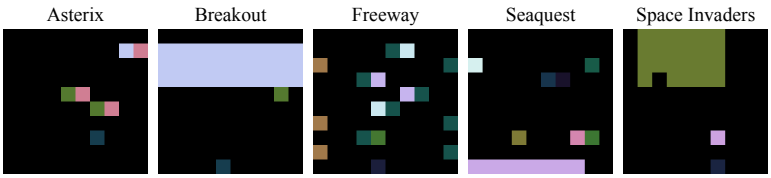


Figure 7: The five MiniAtar environments (Young & Tian, 2019) used in our experiments: Asterix, Breakout, Freeway, Seaquest, and Space Invaders. Observations are represented by native 10×10 binary feature planes.

B.2 Spiking Neuron Parameters

We use the same LIF neuron parameters across all environments unless otherwise specified. The membrane states are initialized to zero for every forward pass and are not propagated across RL interaction steps. The detailed parameters are summarized in Table 3.

Table 3: Parameters of LIF neurons.

Parameter	Value
Membrane leakage parameter λ	0.75
Threshold voltage V_{th}	0.5
Reset voltage V_{reset}	0
Surrogate gradient window size ω	0.5

B.3 Deep Q-Network Parameters

For notational simplicity, we omit the target-network parameters in Eq. (5); the implementation uses a periodically updated target network. For exposition, we present the squared TD loss; the experiments use the Huber loss according to Mnih et al. (2015); Young & Tian (2019).

B.3.1 CliffWalking

For CliffWalking, the ANN baseline consists of two fully connected hidden layers with 64 units and ReLU activations, followed by a four-dimensional linear output layer. The SNN uses the same hidden dimensions, replacing ReLU activations with LIF neurons (Gerstner & Kistler, 2002). The CMC auxiliary ANN has the same two 64-dimensional hidden layers and a scalar output.

All methods are trained for $50{,}000$ environment steps using Adam with a learning rate of 3×10^{-4}. The replay buffer contains $10{,}000$ transitions, with training starting after the first 500 interactions. Minibatches contain 64 transitions. We use $\gamma = 0.99$, synchronize the target network every 250 optimization steps, and clip the gradient norm to 10. The exploration rate is linearly decreased from 1.0 to 0.05 over the first $10{,}000$ training updates. Evaluation is performed every 250 environment steps using a deterministic greedy policy.

Table 4: DQN parameters for CliffWalking.

Parameter	Value
Hidden layers	64–64
Training steps	50,000
Replay capacity	10,000
Replay warm-up	500
Batch size	64
Discount factor γ	0.99
Optimizer	Adam
Learning rate	3×10^{-4}
Target update interval	250 updates
Initial / final ϵ	$1.0/0.05$
Exploration decay	10,000 updates
Gradient clipping	10
Evaluation interval	250 steps

B.3.2 ATARI

For Atari, we follow the convolutional architecture commonly used by DQN. The input is processed by three convolutional layers with 32 8×8 filters of stride 4, 64 4×4 filters of stride 2, and 64 3×3 filters of stride 1, respectively. The resulting 3136-dimensional representation is passed to a fully connected layer with 512 units and then to a linear output layer. The ANN baseline uses ReLU activations, whereas the DSQN replaces each hidden activation with an LIF neuron (Gerstner & Kistler, 2002). In CMC-DSQN, the auxiliary ANN uses an independent network of the same architecture with a scalar output.

We train each Atari agent for 30 million environment transitions using 256 parallel CuLE environments (Dalton & frosio, 2020). The replay buffer has a capacity of 10^6 transitions and training starts after $50{,}000$ transitions. We use minibatches of 32, $\gamma = 0.99$, and one optimization step per four collected transitions. The optimizer is DeepMind-style RMSProp with learning rate 2.5×10^{-4}, decay 0.95, and $\epsilon_{\text{opt}} = 0.01$. The target network is synchronized every $10{,}000$ optimization steps. Exploration is linearly annealed from $\epsilon = 1.0$ to 0.1 over 10^6 transitions after replay warm-up.

During evaluation, we use $\epsilon = 0.05$ and evaluate 30 episodes every $250{,}000$ transitions. Each episode is capped at 4500 action decisions, corresponding to at most $18{,}000$ emulator frames.

B.3.3 MINIATAR

For MiniAtar, each binary observation is first processed by a convolutional layer with 16 3×3 filters and stride 1. The resulting $16 \times 8 \times 8$ representation is flattened and passed through a 128-unit fully connected hidden layer, followed by the linear action-value output. As in Atari, ANN hidden layers use ReLU activations and SNN hidden layers use LIF neurons. The auxiliary CMC ANN uses the same backbone with a scalar output.

Agents are trained for 5 million transitions using 256 parallel PGX environments (Koyamada et al., 2023). The replay buffer contains $100{,}000$ transitions, with a warm-up period of 5000 transitions and minibatches of 32. We use $\gamma = 0.99$ and perform one optimizer update per collected transition. Training uses centered RMSProp with learning rate 2.5×10^{-4}, decay 0.95, and $\epsilon_{\text{opt}} = 0.01$. The target network is updated every 1000 optimization steps. Exploration is annealed from 1.0 to 0.1 over $100{,}000$ transitions after warm-up.

Table 5: DQN parameters for Atari.

Parameter	Value
Training transitions	30M
Parallel environments	256
Replay capacity	1M
Replay warm-up	50K
Batch size	32
Update frequency	every 4 transitions
Discount factor γ	0.99
TD loss	Huber
Optimizer	RMSProp
Learning rate	2.5×10^{-4}
RMSProp decay	0.95
RMSProp ϵ	0.01
Target update interval	10,000 updates
Initial / final ϵ	$1.0/0.1$
Exploration decay	1M transitions
Evaluation interval	250K transitions
Evaluation episodes	30
Evaluation ϵ	0.05
Maximum episode length	4500 decisions

Evaluation is performed every 100,000 transitions using a greedy policy. We evaluate 10 episodes per checkpoint, with each episode capped at 10,000 environment steps.

Table 6: DQN parameters for MiniAtar.

Parameter	Value
Training transitions	5M
Parallel environments	256
Replay capacity	100K
Replay warm-up	5K
Batch size	32
Update frequency	every transition
Discount factor γ	0.99
TD loss	Huber
Optimizer	centered RMSProp
Learning rate	2.5×10^{-4}
RMSProp decay	0.95
RMSProp ϵ	0.01
Target update interval	1000 updates
Initial / final ϵ	$1.0/0.1$
Exploration decay	100K transitions
Evaluation interval	100K transitions
Evaluation episodes	10
Evaluation policy	Greedy
Maximum episode length	10,000 steps

C Additional Experiemental Results

C.1 Learning Curves on MiniAtar

Figures 8–10 provide the complete learning curves of CMC-DSQN and the compared methods on all five MiniAtar environments under different simulation timesteps. These results complement the

aggregate performance reported in Sec. 6.2. Across different values of T, CMC-DSQN generally achieves faster learning and higher asymptotic performance, with particularly clear improvements on Asterix, Seaquest, and SpaceInvaders. The gains are most pronounced in the low-timestep setting, further supporting the effectiveness of common-mode compensation when the SNN simulation budget is limited.

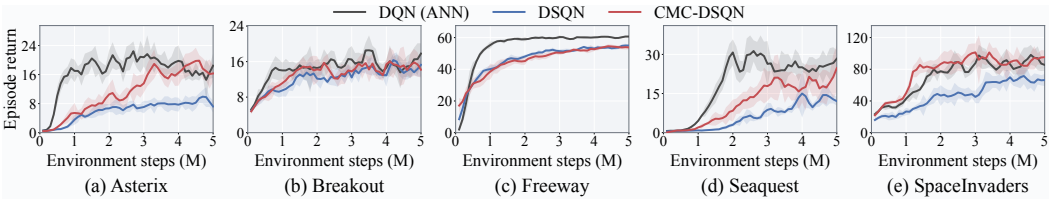


Figure 8: Learning curves on the five MiniAtar environments at $T = 2$. Shaded regions denote one half of the standard deviation over five random seeds. All curves are smoothed uniformly for visual clarity.

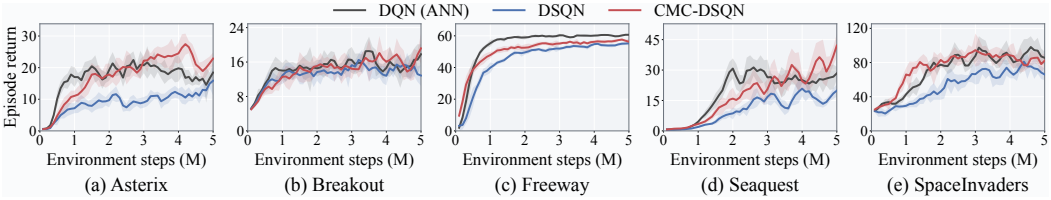


Figure 9: Learning curves on the five MiniAtar environments at $T = 4$. Shaded regions denote one half of the standard deviation over five random seeds. All curves are smoothed uniformly for visual clarity.

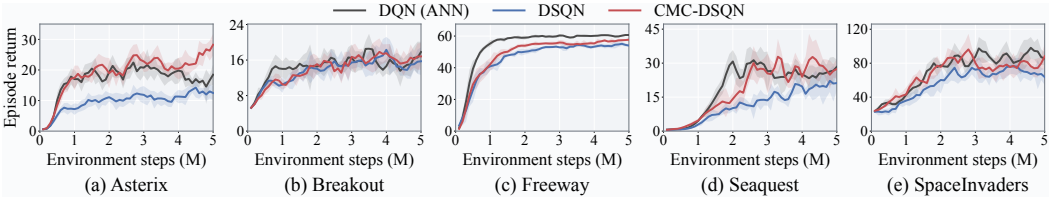


Figure 10: Learning curves on the five MiniAtar environments at $T = 6$. Shaded regions denote one half of the standard deviation over five random seeds. All curves are smoothed uniformly for visual clarity.

C.2 Ablation Studies

We provide complete per-environment results for the ablation variants introduced in Sec. 6.3.

Alternative designs for CMC-DSQN. We consider two alternative designs in addition to the vanilla DSQN. **S-CMC** replaces the auxiliary ANN in CMC-DSQN with an SNN, testing whether the improvement can be attributed merely to introducing an additional estimator. **QEC-DSQN** instead uses the auxiliary ANN to estimate the complete Q-value vector rather than only its common-mode component, testing whether targeted common-mode estimation is preferable to full Q-value estimation.

As shown in Table 7, CMC-DSQN achieves the highest APR at all three simulation timesteps. S-CMC improves over the vanilla DSQN but consistently remains below CMC-DSQN, supporting the importance of a reliable auxiliary estimator for the common-mode component. QEC-DSQN also consistently underperforms CMC-DSQN and can even degrade performance at $T = 6$, indicating that estimating the complete Q-value vector is less effective than explicitly isolating the com-

mon mode. These results support the two key design choices of CMC-DSQN: using an ANN for common-mode estimation and restricting the auxiliary estimator to the common-mode component.

Figures 11–13 further report the corresponding learning curves at $T = 2$, 4, and 6. The learning dynamics are consistent with the aggregate results in Table 7, with CMC-DSQN generally showing stronger learning performance across different simulation timesteps.

Table 7: Average maximum return over five random seeds and average performance ratio (APR) on MiniAtar environments after 5M environment transitions, where $\pm$ denotes one standard deviation.

Timesteps	Methods	Asterix	Breakout	Freeway	Seaquest	SpaceInvaders	APR
$T = 2$	DSQN	14.50 ± 3.04	23.90 ± 1.22	56.92 ± 1.06	22.84 ± 15.67	100.76 ± 11.51	67.76%
	QEC-DSQN	18.93 ± 3.72	25.00 ± 5.34	51.47 ± 2.28	18.52 ± 10.08	111.62 ± 8.78	69.48%
	S-CMC	22.23 ± 2.36	23.47 ± 2.78	56.07 ± 1.87	17.40 ± 2.87	132.53 ± 14.74	74.58%
	CMC-DSQN	27.30 ± 4.94	23.50 ± 5.09	56.74 ± 1.12	41.06 ± 14.90	150.92 ± 36.31	**90.30%**
$T = 4$	DSQN	22.04 ± 5.54	22.96 ± 4.56	57.04 ± 1.10	33.10 ± 6.95	109.50 ± 15.99	77.31%
	QEC-DSQN	29.50 ± 3.15	29.45 ± 6.07	56.70 ± 1.58	38.07 ± 6.56	104.28 ± 8.96	87.86%
	S-CMC	30.58 ± 4.79	22.98 ± 3.89	40.38 ± 30.18	29.72 ± 5.60	124.63 ± 20.16	78.24%
	CMC-DSQN	35.40 ± 2.36	26.92 ± 4.34	59.78 ± 0.37	55.78 ± 8.78	131.44 ± 13.50	**101.92%**
$T = 6$	DSQN	20.98 ± 1.98	28.50 ± 2.19	57.04 ± 0.82	33.98 ± 10.87	115.14 ± 11.39	81.88%
	QEC-DSQN	27.73 ± 2.07	22.12 ± 3.87	57.00 ± 1.26	33.87 ± 7.81	87.92 ± 7.77	77.36%
	S-CMC	33.38 ± 5.92	26.17 ± 3.30	59.43 ± 0.41	35.26 ± 6.34	105.84 ± 15.39	87.89%
	CMC-DSQN	35.44 ± 3.63	26.88 ± 2.75	59.38 ± 0.50	50.12 ± 10.59	131.48 ± 13.94	**99.51%**

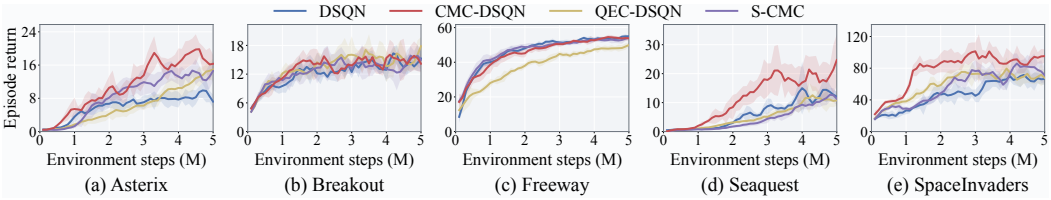


Figure 11: Learning curves of the SNN ablation variants on the five MiniAtar environments at $T = 2$. Shaded regions denote ± 0.5 standard deviations over five random seeds. All curves are smoothed uniformly for visual clarity.

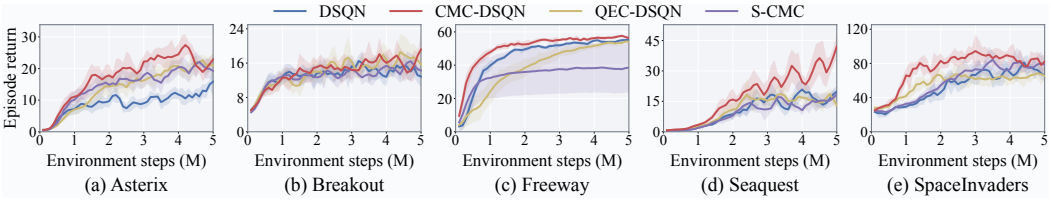


Figure 12: Learning curves of the SNN ablation variants on the five MiniAtar environments at $T = 4$. Shaded regions denote ± 0.5 standard deviations over five random seeds. All curves are smoothed uniformly for visual clarity.

Common-mode compensation for ANN-based DQN. We further apply the same common-mode compensation mechanism to an ANN-based DQN, denoted as CMC-DQN, to examine whether the proposed mechanism also benefits conventional Q-networks. As shown in Table 8 and Fig. 14, CMC-DQN performs comparably to the vanilla DQN and yields only marginal overall benefit. This observation is consistent with the analysis in Sec. 4, where strong common-mode error dominance is observed in low-timestep SNNs but not in ANNs. It therefore suggests that the improvement of CMC-DSQN primarily comes from addressing an SNN-specific estimation problem rather than providing a generic improvement to DQN.

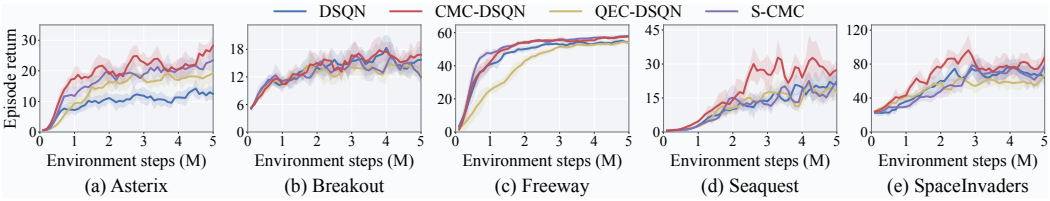


Figure 13: Learning curves of the SNN ablation variants on the five MiniAtar environments at $T = 6$. Shaded regions denote ± 0.5 standard deviations over five random seeds. All curves are smoothed uniformly for visual clarity.

Table 8: Average maximum return over five random seeds and average performance ratio (APR) on MiniAtar environments after 5M environment transitions, where $\pm$ denotes one standard deviation.

Methods	Asterix	Breakout	Freeway	Seaquest	SpaceInvaders	APR
DQN	32.22 ± 4.94	27.76 ± 4.98	62.58 ± 0.34	49.70 ± 8.15	138.34 ± 3.87	100.00%
CMC-DQN	30.18 ± 4.13	28.52 ± 3.28	61.96 ± 0.43	52.46 ± 9.67	150.50 ± 10.80	101.99%

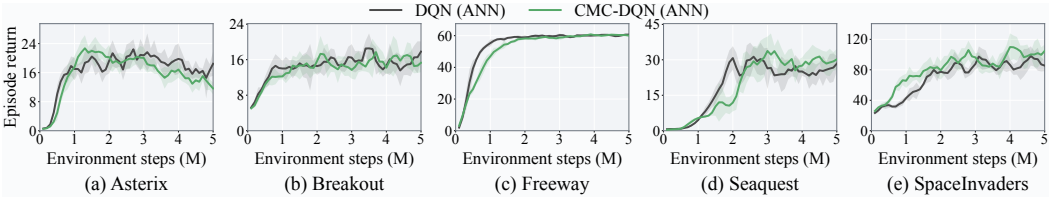


Figure 14: Learning curves of the ANN-based ablation variants on the five MiniAtar environments. Shaded regions denote ± 0.5 standard deviations over five random seeds. All curves are smoothed uniformly for visual clarity.