

Article

Latent Mamba-DQN: Improving Temporal Dependency Modeling in Deep Q-Learning via Selective State Summarization

HanYul Ryu ¹, Chae-Bong Sohn ^{1,*} and Dae-Yeol Kim ^{2,*}¹ Department of Defense Acquisition Program, Kwangwoon University, Seoul 01897, Republic of Korea; hanyul9808@gmail.com² Department of Artificial Intelligence, Kyungnam University, Changwon 51767, Republic of Korea

* Correspondence: cbsohn@kw.ac.kr (C.-B.S.); daeyeol@kyungnam.ac.kr (D.-Y.K.)

Abstract

This study proposes a novel framework, Mamba-DQN, which integrates the state space-based time-series encoder Mamba-SSM into the Deep Q-Network (DQN) architecture to improve reinforcement learning performance in dynamic environments. Conventional reinforcement learning models primarily rely on instantaneous state information, limiting their ability to effectively capture temporal dependencies. To address this limitation, the proposed Mamba-DQN generates latent representations that summarize temporal information from state sequences and utilizes them for both Q-value estimation and Prioritized Experience Replay (PER), thereby enhancing the adaptability of policy learning and improving sample efficiency. The Mamba-SSM offers linear computational complexity and is optimized for parallel processing, enabling real-time learning and policy updates even in environments characterized by high state transition rates. The effectiveness of the proposed framework was validated through experiments conducted in environments with strong temporal dependencies and sparse rewards. Experimental results demonstrate that Mamba-DQN achieves superior stability and efficiency in policy learning compared to conventional DQN, LSTM-DQN, and Transformer-DQN models.

Keywords: reinforcement learning; Deep Q-Network (DQN); state space model (SSM); Mamba-SSM; temporal representation learning; prioritized experience replay (PER)



Academic Editor: Pedro Couto

Received: 4 July 2025

Revised: 11 August 2025

Accepted: 12 August 2025

Published: 14 August 2025

Citation: Ryu, H.; Sohn, C.-B.; Kim, D.-Y. Latent Mamba-DQN: Improving Temporal Dependency Modeling in Deep Q-Learning via Selective State Summarization. *Appl. Sci.* **2025**, *15*, 8956. <https://doi.org/10.3390/app15168956>

Copyright: © 2025 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

Rapidly changing environments such as autonomous driving and real-time robotic control demand fast inference and precise decision-making. These requirements have highlighted the potential of reinforcement learning (RL) as a viable solution, which has been successfully applied in fields like gaming, robotics, and industrial automation [1–6].

In particular, deep learning-based off-policy methods such as the Deep Q-Network (DQN) [7] are considered practical due to their ability to reuse previously collected data during training, thereby achieving high sample efficiency even under limited interactions with the environment [8–11].

However, most existing RL models rely only on instantaneous observations, limiting their ability to capture temporal continuity and contextual cues related to state transitions.

To address these challenges, time-series encoders based on recurrent neural networks (RNN), long short-term memory (LSTM) networks, and Transformer architectures have been incorporated into RL models to improve their ability to capture temporal dependencies [12–17]. Nevertheless, these models still exhibit structural and computational

limitations, especially in environments characterized by high state transition rates or strict real-time processing requirements.

While prior works have incorporated temporal encoders—such as recurrent, attention-based, and latent-state models—to improve decision-making under partial observability, many rely on storing and reusing compressed latent representations in the replay buffer. Such reuse can lead to temporal misalignment between the encoder and the value estimator, as stored features may no longer reflect the agent’s current perception of the environment. Moreover, although some methods compute Q-values over entire sequences, action selection typically depends only on the most recent latent vector. This reliance on compressed latent vectors may cause information degradation and reduce robustness, especially in dynamic environments with frequent state transitions, where the stored latent representations become stale and inconsistent with the current policy.

To overcome these limitations, we propose an architecture that integrates the state space model (Mamba-SSM) into the DQN framework. Additionally, we introduce a training strategy called full state sequence replay (FSSR), in which entire observation sequences are re-encoded at each training step using the most recent network parameters. By discarding outdated vectors and reconstructing them from raw inputs, this method ensures temporal alignment between the encoder and the Q-value estimator. Unlike conventional approaches that emphasize efficiency through latent vector reuse, our strategy prioritizes structural consistency and temporal alignment, aiming to preserve the fidelity of learned representations in dynamic environments.

The main contributions of this study are as follows:

- We propose an architecture that integrates the Mamba-SSM into the DQN framework to enhance temporal representation.
- We introduce a learning strategy for Mamba-DQN, based on FSSR, which re-encodes state sequences during training to preserve temporal alignment between the encoder and the value function.
- We empirically demonstrate the effectiveness of our approach through comparative evaluations and policy performance improvements across multiple dynamic environments.
- We formulate a unified design philosophy that prioritizes temporal consistency and structural alignment over latent reuse, providing a higher-performing alternative to existing methods for incorporating temporal dependencies.

2. Related Work

2.1. Deep Recurrent Q-Network

In Partially Observable Markov Decision Process (POMDP) environments, reinforcement learning models that rely solely on single-step observations suffer from reduced state estimation accuracy due to the lack of temporal information. The Deep Recurrent Q-Network (DRQN) [18] was proposed to address this limitation by integrating LSTM [19] into the conventional DQN architecture.

In the DRQN structure, the first fully connected layer of the original DQN is replaced with an LSTM module, enabling the model to retain temporally continuous observation information in memory and utilize this information for more accurate state estimation.

In the same study, a POMDP environment was simulated by applying probabilistic frame masking, referred to as flickering conditions, to Atari game environments. The experimental results demonstrated that the DRQN effectively integrates temporal information, enabling stable policy learning even under observation loss conditions. Furthermore, DRQN achieved more generalized representation learning capability compared to approaches based on frame stacking and demonstrated improved policy performance relative to conventional DQN models.

2.2. Deep Transformer Q-Network

The Deep Transformer Q-Network (DTQN) [20] applies a self-attention-based Transformer architecture to encode temporal information in partially observable environments. To address the inherent limitations of recurrent architectures, such as LSTM and gated recurrent unit (GRU), including restricted parallelism and instability during training, the DTQN adopts an architecture that integrates a Transformer decoder with positional encoding [21].

This design enables the model to encode the entire sequence of observations and estimate Q-values for each time step. In the proposed architecture, the action is selected based on the output at the most recent time step; however, during training, an intermediate Q-value prediction strategy is applied, in which Q-values for all time steps are computed to improve the learning efficiency.

The performance of the DTQN was evaluated in various environments, including Gridverse, Hallway, Memory Cards, and Car Flag. The experimental results demonstrated that the DTQN outperforms conventional methods such as the DRQN, DQN, and Attention-Q in terms of both learning speed and final policy performance. These findings confirm that the Transformer architecture is highly effective in capturing the global characteristics of temporal information in reinforcement learning tasks.

2.3. Mamba-DQN Adaptively Tunes Visual SLAM Parameters Based on Historical Observation DQN

An architecture was proposed that incorporates a Mamba-SSM into the DQN framework to optimize the hyperparameters of visual simultaneous localization and mapping (Visual SLAM) systems using reinforcement learning [22]. In this approach, the action space is defined based on parameters of ORB-SLAM3, and temporal observation information is integrated through the Mamba block, enabling Q-value estimation that reflects the temporal context.

The proposed method demonstrated superior performance compared to conventional SLAM techniques in terms of both localization accuracy and computational efficiency, as validated through experiments conducted on the EuRoC Micro Aerial Vehicle (EuRoC MAV) and Technical University of Munich Visual-Inertial (TUM-VI) datasets.

Building upon this architectural foundation, the present study applies the temporal modeling capability of Mamba-based state space models to policy learning in general reinforcement learning environments. In particular, the impact of effectively embedding temporal information on policy stability and learning efficiency is experimentally evaluated in representative dynamic environments characterized by high state transition rates.

3. Materials and Methods

3.1. Proposed Architecture

This section describes the overall architecture of the proposed Mamba-DQN framework and the computational relationships among its four primary components.

Figure 1 illustrates the architecture in which data generated through agent–environment interaction are sequentially processed by the Mamba-SSM, the DQN, and the experience replay buffer.

Each module performs a distinct role: encoding full state sequences to capture temporal dependencies, estimating action values, learning the policy, and reusing past experiences. The data flow among these components forms a cyclic process that enables continuous learning.

The environment provides the agent with observable state sequences and returns rewards and next states according to the agent's selected actions. The system receives a state sequence of length determined by a tunable hyperparameter and feeds it into the

time-series encoder, the Mamba-SSM-based model. The Mamba-SSM selectively updates its internal states to capture temporal dependencies, producing a sequence representation z_t that is passed to the DQN to estimate Q-values for each action and select the final action via the epsilon-greedy policy.

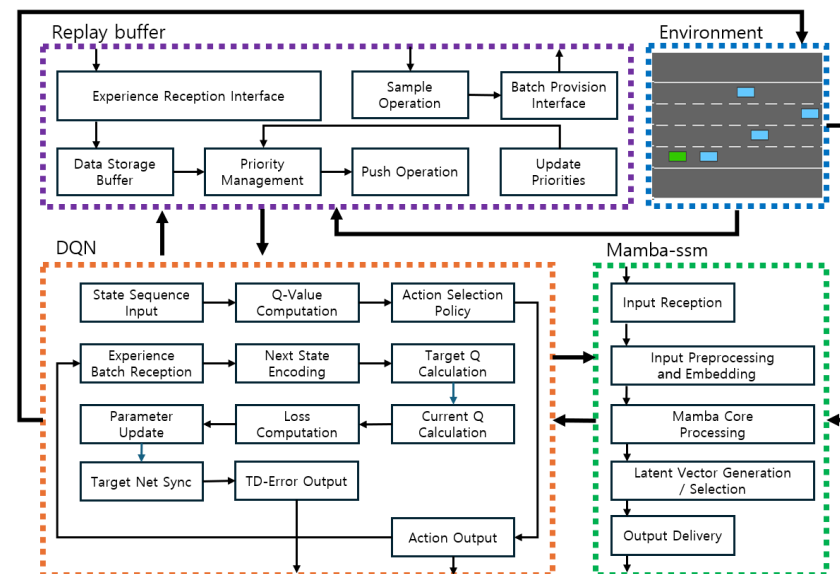


Figure 1. Overall architecture of the Mamba-DQN framework integrating a Mamba-SSM encoder, temporally aligned Q-value estimation, and FSSR with TD-priority sampling.

The selected action is transmitted to the environment, which executes the action and returns the corresponding reward, the next state, and termination information. The agent then constructs the next state sequence by appending the new state to its temporal window. This process forms a single transition experience (containing the full raw state sequences for the current and next steps), which is stored in the experience replay buffer.

For training, the replay buffer samples a minibatch of stored experiences with priority based on the temporal-difference (TD) error. The DQN re-processes these raw sequences to compute the target and current Q-values, and the loss is calculated from the resulting TD error. Finally, the newly calculated TD errors are fed back to the replay buffer to update each experience's sampling priority.

The primary functions of each module are summarized as follows:

- **Environment:** Provides state information and rewards.
- **Mamba-SSM:** Encodes the entire state sequence to generate a representation that captures temporal dependencies for action selection.
- **DQN:** Selects actions based on the input sequence representation. For learning, it re-encodes the entire state sequences sampled from the replay buffer using the FSSR strategy with the most recent network parameters to ensure temporal alignment between the encoder and the value estimator. It then updates its parameters to minimize the loss, with periodic synchronization of the target network.
- **Replay buffer:** Stores transition experiences and improves sample efficiency through TD error-based probabilistic sampling.

The proposed framework is designed to maintain temporal alignment and structural consistency by re-encoding full state sequences during training. This process integrates temporal information into policy learning and improves both representational capacity and sample efficiency compared to conventional DQN architectures.

3.2. Reinforcement Learning Procedure

Figure 2 illustrates the training pipeline of the proposed Mamba-DQN framework.

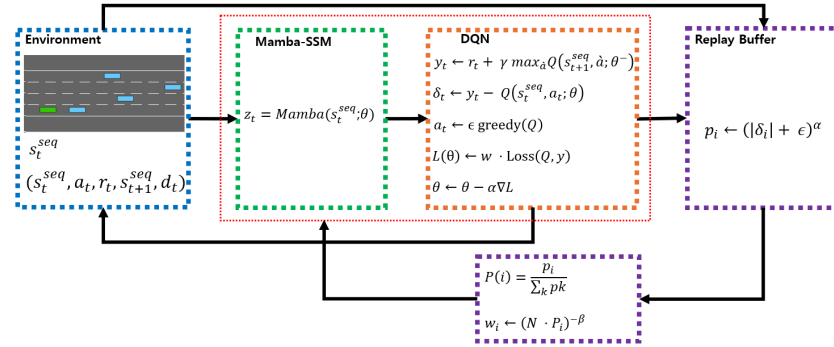


Figure 2. Equation-based learning procedure and module interaction structure of the Mamba-DQN framework.

The learning process consists of four main components: temporal encoding using the Mamba-SSM, action selection and value estimation via the Q-network, experience storage and replay, and parameter updates based on TD errors and prioritized sampling.

At each time step t , the Mamba-SSM encodes the current state sequence $\mathbf{s}_t$ into a latent vector $\mathbf{z}_t$, which is then used for action selection:

$$\mathbf{z}_t = \text{Mamba}(\mathbf{s}_t; \theta). \quad (1)$$

The agent's experience, consisting of the full state sequence and transition information, is stored in the replay buffer $\mathcal{D}$. Unlike a latent-replay strategy, our approach stores the raw state sequences to ensure learning stability.

$$\mathcal{D} := \{(\mathbf{s}_t, a_t, r_t, \mathbf{s}_{t+1}, d_t)\} \quad (2)$$

During training, a batch is sampled from $\mathcal{D}$.

To ensure that the Q-value estimations are always based on the most up-to-date network parameters, our method re-processes the full state sequences for both the current and target Q-value calculations. The TD target y_t is calculated by feeding the next-state sequence $\mathbf{s}_{t+1}$ into the target network $Q(\cdot, \cdot; \theta^-)$. Similarly, the current Q-value is determined by feeding the current state sequence $\mathbf{s}_t$ through the main network $Q(\cdot, \cdot; \theta)$.

This process defines the TD error δ_t :

$$y_t = r_t + \gamma \max_{a'} Q(\mathbf{s}_{t+1}, a'; \theta^-), \quad (3)$$

$$\delta_t = y_t - Q(\mathbf{s}_t, a_t; \theta). \quad (4)$$

Each sample's priority p_i is then updated based on the absolute TD error, ensuring that more informative transitions are sampled more frequently:

$$p_i = (|\delta_i| + \epsilon)^\alpha. \quad (5)$$

This FSSR approach, while computationally more intensive, ensures that the gradient updates for both the Mamba encoder and the Q-value head are always consistent with the current network state. This design choice prioritizes faster convergence and higher performance over the potential memory efficiency gains from reusing stale latent vectors.

3.3. Experimental Environment and Settings

To comprehensively evaluate the temporal encoding performance and policy learning effectiveness of the proposed Mamba-DQN architecture, three RL environments with fundamentally different characteristics were employed as experimental platforms. The environments differ in observation structure, reward density, and policy response requirements. They were selected to verify the generalization performance of the proposed model across diverse temporal dynamics.

The first environment is Highway-fast-v0, a highway driving simulation environment [23]. It imposes complex learning conditions characterized by frequent state transitions, multiple heterogeneous dynamic factors, and strict real-time decision-making requirements. The agent receives a 25-dimensional numerical state vector containing information about both the ego vehicle and surrounding vehicles. This vector includes speed, position, lane index, and relative distances and velocities. The input is provided as structured numerical values rather than visual images. The action space consists of five discrete actions: maintaining speed, accelerating, decelerating, changing to the left lane, and changing to the right lane.

The second environment is LunarLander-v3, characterized by sparse rewards and challenging policy convergence [24]. This environment involves a two-dimensional landing control problem under gravity, where the state is represented as an 8-dimensional continuous vector containing position, velocity, and landing angle information. The action space is defined by four discrete actions corresponding to the main thruster and left/right directional thrusters. Due to sparse rewards and highly dynamic interactions, this environment serves as a meaningful benchmark for evaluating temporal information representation and policy stability.

The third environment is the Atari 2600 Pong environment from the Arcade Learning Environment [25]. Unlike the previous two tasks, Pong provides high-dimensional visual observations in the form of 210×160 RGB pixel frames, which are converted to grayscale and downsampled before being processed by a convolutional encoder. This configuration enables the evaluation of the generalization capability of the proposed architecture in convolutional feature extraction contexts. Owing to the visual nature of the input and the sparse reward signal—occurring only when a point is scored—this environment serves as a challenging benchmark for verifying whether the proposed Mamba-DQN framework, when combined with a convolutional front-end, can maintain temporal consistency and policy stability across non-vector observation modalities.

In all experiments, Mamba-DQN employed a fixed-length state sequence as the encoder input, where the sequence length T was defined as a tunable hyperparameter rather than a fixed constant. This design enables the agent to incorporate the most recent T states, including the current time step, into the decision-making process. Formally, the input sequence at time step t is defined as

$$\mathbf{s}_t^{\text{seq}} = \{\mathbf{s}_{t-T+1}, \dots, \mathbf{s}_t\} \in \mathbb{R}^{T \times \text{input_dim}}. \quad (6)$$

The Mamba encoder compresses the input sequence into a state sequence encoding that encapsulates temporal dependencies for action selection, while raw state sequences are stored in the replay buffer and re-encoded at training time using the current network parameters to ensure temporal consistency during learning.

Figure 3 illustrates how the Mamba encoder generates state sequence encodings from recent observations for online decision-making, while learning is performed using freshly encoded sequences retrieved from the buffer.

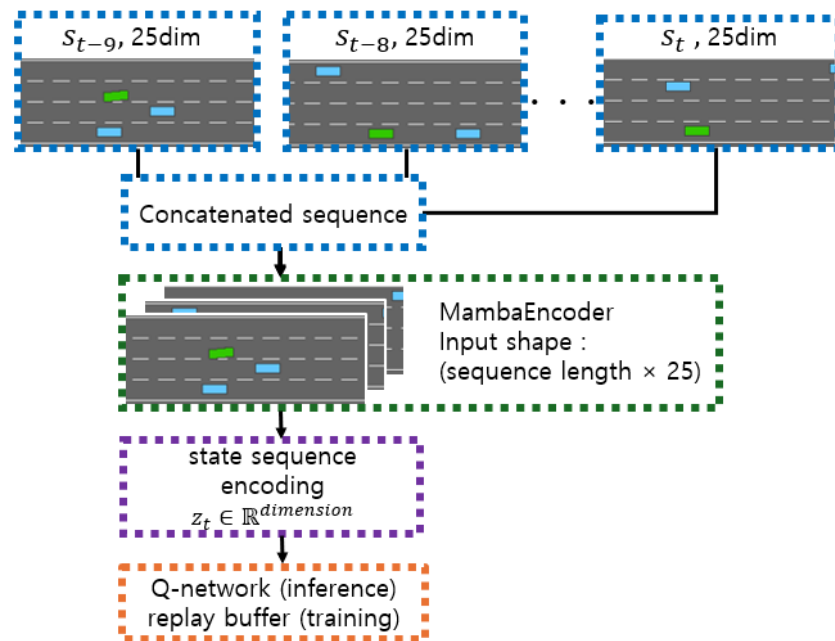


Figure 3. State sequence encoding process using the Mamba encoder, which transforms recent observations into temporally aligned representations for action selection and learning.

The following section details the hyperparameter configurations used in all experiments to ensure fair and consistent comparisons across models.

3.4. Hyperparameter Configuration

All experiments were conducted under identical environmental conditions and learning protocols to ensure fair comparisons. The four model architectures compared in this study are DQN, LSTM-DQN, Transformer-DQN, and Mamba-DQN. The state representation, Q-network structure, and reward function were kept consistent across all models, with only the time-series encoder varied. Table 1 summarizes the key hyperparameters and input configurations for all models.

Table 1. Key hyperparameter settings and training environment configurations for all comparative models. The sequence length parameter T , a tunable hyperparameter, was fixed at 10 for the Highway-fast-v0 and LunarLander-v3 environments and 4 for the Atari 2600 Pong environment.

Parameter	Value
DQN sequence length (T)	-
LSTM-DQN sequence length (T)	10/4
Transformer-DQN sequence length (T)	10/4
Mamba-DQN sequence length (T)	10/4
Gradient clipping threshold	0.1/0.5/1.0
Learning rate	5×10^{-4}

All models used the same learning rate, discount factor, batch size, replay buffer size, and other relevant hyperparameters to ensure strict control of experimental conditions. By isolating the time-series encoder as the only architectural difference, this experimental design enables a quantitative evaluation of how temporal sequence modeling affects policy learning performance.

4. Results

4.1. Experimental Setup

To comprehensively evaluate the performance of Mamba-DQN, experiments were conducted in three distinct environments: Highway-fast-v0, LunarLander-v3, and Atari 2600 Pong. The Highway-fast-v0 environment, featuring frequent state transitions in autonomous driving simulations, served as the primary benchmark for structural and hyperparameter analyses. In contrast, the LunarLander-v3 environment, characterized by sparse rewards and high control complexity, was employed to assess the model's generalization capability.

Smoothed total reward and TD loss were used as the primary evaluation metrics, where smoothing was applied for visualization purposes. For Highway-fast-v0, the performance was averaged over steps 200 to 20,000 to exclude the initial exploration phase, while for LunarLander-v3, hyperparameters derived from the Highway-fast-v0 environment were applied without modification, and the performance was averaged over steps 10,000 to 200,000.

To further evaluate the generalization capability of Mamba-DQN in a visually rich and dynamic environment, experiments were conducted in the Atari 2600 Pong environment. Pong is characterized by dense rewards and fast-paced two-dimensional spatiotemporal dynamics. All models were trained for 1,500,000 steps and were used to assess the generalization performance in a pixel-based visual input environment.

All models were trained five times independently under identical hyperparameter settings, and the average results were reported to ensure statistical reliability. Experiments were conducted on an Ubuntu 22.04 system equipped with an NVIDIA RTX 3050 Laptop GPU (MSI Sword 15 A12UC, MSI, Taipei, Taiwan), Mamba-SSM version 2.2.5, Causal-Conv1D version 1.5.2, and PyTorch version 1.12. The random seed was fixed to 42 to ensure reproducibility.

4.2. Comparative Evaluation with Baseline Models

In this section, the policy learning performance of Mamba-DQN is quantitatively compared to conventional DQN-based models. For each model, gradient clipping thresholds of 0.1, 0.5, and 1.0 were tested, and five independent training runs were conducted under identical environmental conditions. The average smoothed reward and TD loss were computed for each setting, and the clipping coefficient that yielded the best performance was adopted as the optimal value for each model. Based on these optimal conditions, the policy learning efficiency and stability of Mamba-DQN were compared with those of the baseline models.

Table 2 summarizes the performance results for each model in the Highway-fast-v0 environment. Mamba-DQN outperformed all baseline models in terms of both average smoothed reward and TD loss. Specifically, with a gradient clipping coefficient of 0.5, Mamba-DQN achieved an average reward of 20.9864, showing a 26.1% improvement over DQN, and a TD loss of 0.0207, indicating an 84.9% reduction relative to DQN.

These results suggest that the structural design of Mamba-DQN, which effectively captures temporal dependencies and selectively integrates important experiences into learning, contributes positively to both convergence speed and performance.

Figure 4 presents the smoothed reward convergence curves for each model. The results show that Mamba-DQN achieves faster reward improvement during the early training stages and maintains the highest average reward throughout the training process.

Figure 5 presents the TD loss trends for each model in the Highway-fast-v0 environment. Mamba-DQN maintained lower TD errors from the early training phase and

consistently exhibited reduced TD loss values throughout training, reflecting improved temporal representation and policy updates enabled by full state sequence re-encoding.

Table 2. Performance comparison in Highway-fast-v0 under fixed hyperparameter settings. *t*-test results are computed against Mamba-DQN with Clipping = 0.5 as the reference.

Model	Clipping	Max	Min	Mean $\pm$ Std	<i>t</i> -Stat	<i>p</i> -Value	Cohen's <i>d</i>
Mamba-DQN	1.0	19.92	16.28	17.52 $\pm$ 1.45	−2.808	0.0275	−1.776
Mamba-DQN	0.5	24.08	18.20	20.99 $\pm$ 2.34	—	—	—
Mamba-DQN	0.1	22.36	17.19	20.06 $\pm$ 2.51	−0.606	0.5612	−0.383
LSTM-DQN	1.0	15.88	11.00	12.90 $\pm$ 1.95	−5.938	0.00039	−3.756
LSTM-DQN	0.5	13.66	9.76	12.29 $\pm$ 1.51	−6.973	0.00024	−4.410
LSTM-DQN	0.1	14.49	10.98	12.64 $\pm$ 1.31	−6.954	0.00036	−4.398
Transformer-DQN	1.0	20.46	14.27	16.63 $\pm$ 2.42	−2.890	0.0202	−1.828
Transformer-DQN	0.5	17.80	13.68	15.82 $\pm$ 1.50	−4.150	0.00455	−2.625
Transformer-DQN	0.1	19.16	14.96	16.66 $\pm$ 1.63	−3.393	0.0112	−2.146
DQN	1.0	19.87	14.22	16.47 $\pm$ 2.15	−3.179	0.01315	−2.010
DQN	0.5	18.72	14.55	16.64 $\pm$ 1.63	−3.410	0.01098	−2.157
DQN	0.1	17.50	13.70	15.87 $\pm$ 1.50	−4.114	0.00476	−2.602

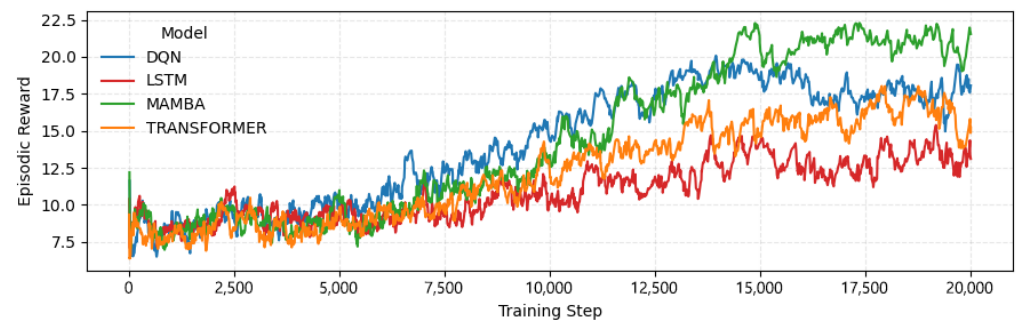


Figure 4. Comparison of smoothed reward curves for each model in the Highway-fast-v0 environment. All models were trained five times under identical settings, and a smoothing factor of 0.9 was applied. Mamba-DQN consistently achieved the highest reward across all training steps, indicating both faster convergence and reduced variance in reward.

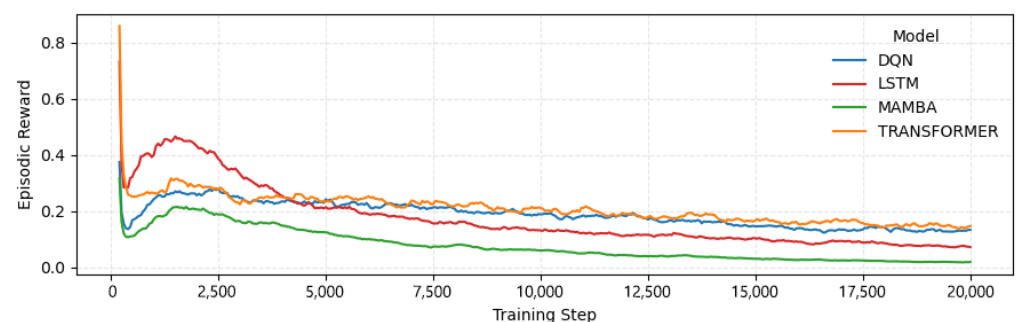


Figure 5. Comparison of TD loss trends for each model in the Highway-fast-v0 environment. Mamba-DQN consistently achieved the lowest loss across all training steps, with particularly rapid convergence observed between 5000 and 15,000 steps. This result reflects the structural advantages of the proposed model in enhancing value estimation accuracy and overall training stability.

To further evaluate the generalization capability and environment-agnostic performance of the proposed architecture, additional experiments were conducted in the LunarLander-v3 environment using the same unmodified hyperparameter settings. This environment is characterized by sparse rewards and complex physical dynamics, making it suitable for assessing temporal representation capability and learning stability.

Table 3 summarizes the performance of all models in the LunarLander-v3 environment under fixed hyperparameter settings. Mamba-DQN (Clip 1.0) achieved the highest maximum reward (236.56), indicating a significantly higher performance ceiling than the baseline models. However, this peak performance came at the cost of stability; a single outlier run with a large negative reward heavily skewed the mean, resulting in a performance distribution with high variance (Std: 109.78).

Table 3. Performance comparison in LunarLander-v3 under fixed hyperparameter settings. *t*-test results are computed against Mamba-DQN with Clipping = 1.0 as the reference.

Model	Clipping	Max	Min	Mean $\pm$ Std	<i>t</i> -Stat	<i>p</i> -Value	Cohen's <i>d</i>
Mamba-DQN	1.0	236.56	−15.33	139.46 $\pm$ 109.78	—	—	—
Mamba-DQN	0.5	125.15	−168.14	9.65 $\pm$ 108.24	−1.883	0.0965	−1.191
Mamba-DQN	0.1	110.55	0.28	65.05 $\pm$ 41.81	−1.416	0.2144	−0.896
LSTM-DQN	1.0	116.64	60.07	99.60 $\pm$ 23.34	−0.794	0.4681	−0.502
LSTM-DQN	0.5	197.92	24.66	111.37 $\pm$ 78.18	−0.466	0.6549	−0.295
LSTM-DQN	0.1	191.52	79.87	145.05 $\pm$ 42.76	0.106	0.9195	0.067
Transformer-DQN	1.0	179.30	14.7283	79.31 $\pm$ 60.75	−0.2739	0.7927	−0.1732
Transformer-DQN	0.5	144.14	61.75	92.11 $\pm$ 35.28	−0.918	0.4021	−0.581
Transformer-DQN	0.1	166.61	−20.06	54.72 $\pm$ 90.02	−1.335	0.2201	−0.844
DQN	1.0	66.74	−3.71	27.40 $\pm$ 26.57	−2.219	0.0837	−1.403
DQN	0.5	2.68	−56.02	−12.02 $\pm$ 24.95	−3.009	0.0349	−1.903
DQN	0.1	—	—	—	—	—	—

Figure 6 presents the smoothed reward convergence curves for each model in the LunarLander-v3 environment. The results indicate that Mamba-DQN achieves faster reward improvement during the early training phase and reaches convergence after approximately 175,000 steps.

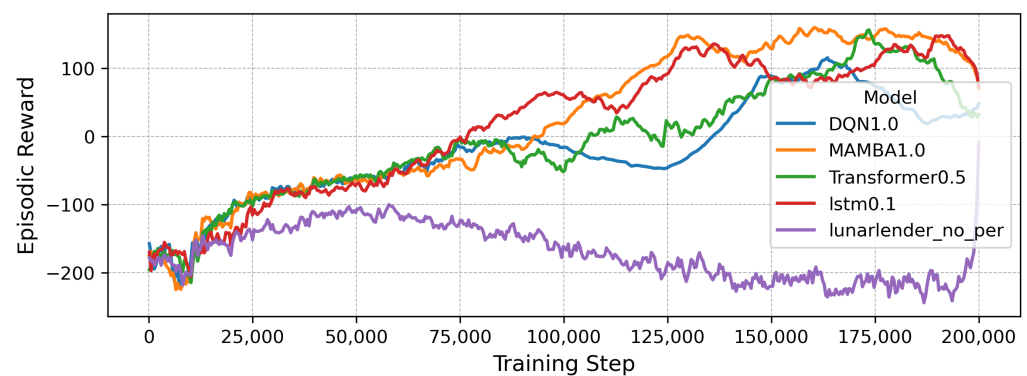


Figure 6. Smoothed reward convergence curves in the LunarLander-v3 environment. Mamba-DQN achieves faster reward improvement and converges to a reward of 200 after approximately 175,000 steps compared to baseline models.

Figure 7 shows the TD loss trends for each model. The results show that Mamba-DQN maintains lower TD loss after convergence.

Previous studies have reported that in the LunarLander-v3 environment, DQN-based models typically require at least 500,000 training steps to achieve convergence, especially when the initial policy performance is low [26,27].

In comparison, Mamba-DQN achieved policy convergence and competitive reward levels within approximately 200,000 training steps in the same environment. These results indicate that the proposed time-series encoding architecture improves state representation

and enables effective priority-based sample selection during the early stages of training, thereby facilitating more efficient policy learning with reduced sample requirements.

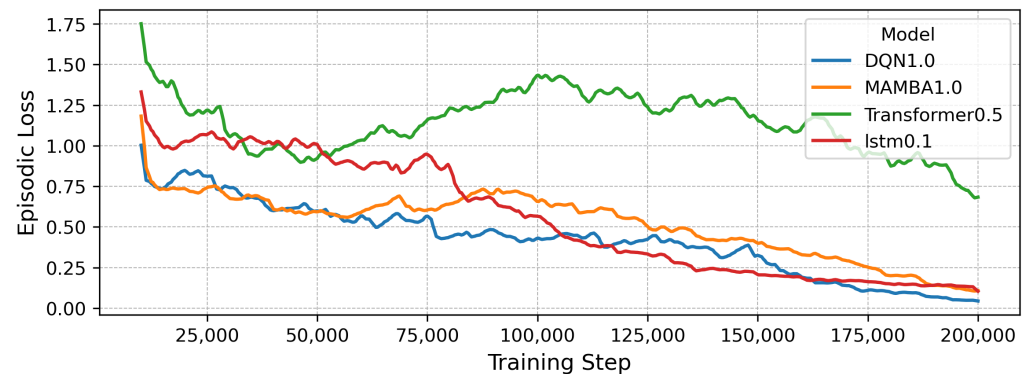


Figure 7. TD loss trends in the LunarLander-v3 environment. Mamba-DQN maintains low TD loss after convergence, indicating improved training efficiency and robustness compared to the baseline models.

To further assess the generalization capability of Mamba-DQN in a high-dimensional visual domain, additional experiments were conducted in the Atari 2600 Pong environment. Atari 2600 Pong is characterized by dense rewards and fast-paced two-dimensional spatiotemporal dynamics, requiring both effective temporal modeling and visual feature extraction. For all models, a CNN-based preprocessing module was integrated to process raw pixel observations, while all other hyperparameter settings remained identical to those used in the previous experiments.

For the pixel-based Atari 2600 Pong environment, the gradient clipping coefficient was fixed at 1.0. This decision was guided by results from the other two environments, where a coefficient of 1.0 consistently yielded training and competitive performance. As the Pong experiment was intended to evaluate the proposed architecture’s generalization to high-dimensional visual inputs, we selected a representative and well-performing setting rather than conducting exhaustive hyperparameter tuning.

Although this setting was not individually optimized for each model, the results demonstrate that Mamba-DQN retains its performance advantages in a visually rich and dynamic environment. Each model was trained for 1,500,000 steps, and evaluation was based on smoothed average reward and TD loss. These results are presented in Table 4.

Table 4. Performance comparison in Atari 2600 Pong under fixed hyperparameter settings. *t*-test results are computed against Mamba-DQN as the reference. A dash (–) indicates that the model either failed to converge (average reward below zero across all runs) or exhibited no variation in performance, making statistical comparison invalid.

Model	Max	Min	Mean $\pm$ Std	<i>t</i> -Stat	<i>p</i> -Value	Cohen’s <i>d</i>
Mamba-DQN	11.63	9.27	10.63 $\pm$ 1.22	–	–	–
LSTM-DQN	10.08	7.57	9.01 $\pm$ 1.29	–1.574	0.1909	–1.285
DQN	–	–	–	–	–	–
Transformer-DQN	–	–	–	–	–	–

Figure 8 presents the smoothed reward convergence curves for each model in the Atari 2600 Pong environment. The results indicate that Mamba-DQN achieves faster reward improvement during the early training phase and reaches convergence after approximately 900,000 steps.

Figure 9 shows the TD loss trends for each model in the Atari 2600 Pong environment. The results demonstrate that Mamba-DQN maintains lower TD loss after convergence.

Prior studies have reported that in the Atari 2600 Pong environment, DQN-based models typically require up to 50 million training steps [7] to achieve convergence. In contrast, Mamba-DQN achieved faster policy convergence and competitive reward levels within approximately 1.5 million training steps in the same environment. These results indicate that the proposed architecture improves state representation and effectively facilitates policy learning.

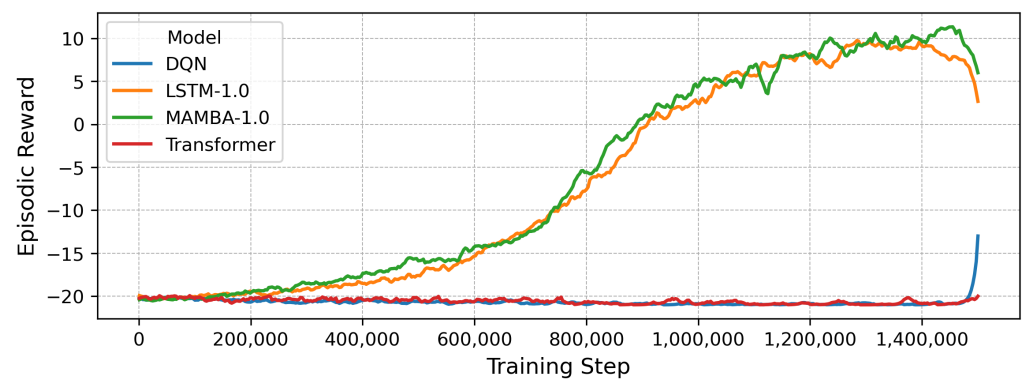


Figure 8. Smoothed reward convergence curves in the Atari 2600 Pong environment. Mamba-DQN achieves faster reward improvement and reaches convergence after approximately 900,000 steps.

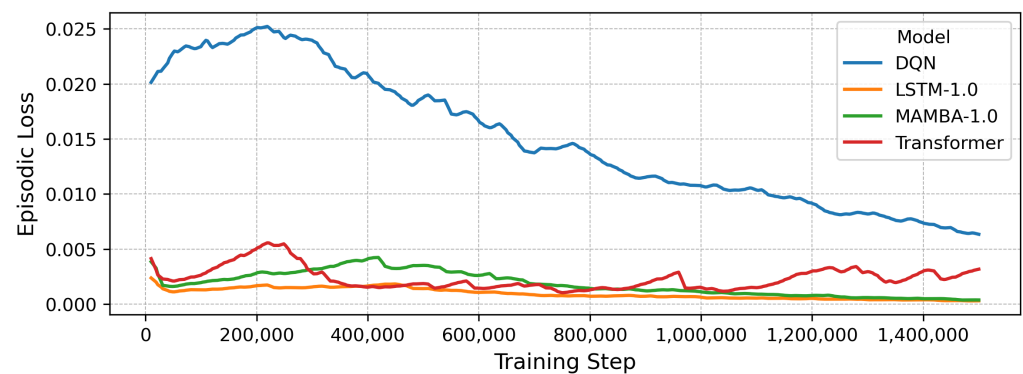


Figure 9. TD loss trends in the Atari 2600 Pong environment. Mamba-DQN maintains low TD loss after convergence, indicating improved training efficiency and performance compared to the baseline models.

The experiments in each proposed environment were designed to evaluate the generalization capability and environment-agnostic applicability of the proposed architecture. Each environment, characterized by sparse reward signals and high control complexity, was employed to assess the ability of the proposed architecture and learning method to achieve policy convergence under such conditions.

In summary, the proposed Mamba-DQN demonstrated consistent performance across environments with different reward densities and state transition dynamics. These results confirm that the proposed time-series encoding-based learning strategy facilitates faster policy convergence and enhances the generalization capability of reinforcement learning policies.

4.3. Ablation Study on the Structural Components of Mamba-DQN

The ablation experiments were conducted with different component removal settings for each environment.

For the 1D signal-based environments, namely LunarLander-v3 and Highway-fast-v0, two ablated variants were evaluated: (i) a model without PER [28] and (ii) a model without the Mamba-SSM encoder. The performance of these variants was compared against the full Mamba-DQN containing all components.

For the 2D image-based Atari 2600 Pong environment, two different ablations were performed: (i) a model without the Mamba-SSM encoder and (ii) a model in which the replay buffer was modified to store only the last state of each sequence while still providing sequence inputs to the network. These variants were compared with the full Mamba-DQN to assess the impact of each component.

Table 5 shows that removing the Mamba-SSM encoder leads to complete failure in learning, as indicated by rewards consistently near -20 . Modifying the replay buffer to store only the last state of each sequence results in a significant performance drop ($p < 0.05$) and a large negative effect size (Cohen's $d = -3.70$), confirming the importance of full sequence context in image-based environments.

Table 5. Ablation study results for Mamba-DQN in Atari 2600 Pong. t -test results are computed against the Full Model as the reference. A dash (–) indicates convergence failure or inapplicable statistical comparison.

Variant	Max	Min	Mean $\pm$ Std	t -Stat	p -Value	Cohen's d
Full Model	11.63	9.27	10.63 ± 1.22	–	–	–
Last-State Buffer	1.27	-10.20	-4.78 ± 5.76	-4.531	0.0386	-3.700
Encoder Removed	-19.82	-20.59	-20.10 ± 0.42	-41.120	0.00014	-33.574

As shown in Table 6, removing PER causes a substantial degradation in performance, with the average reward dropping to -185.40 and $p < 0.01$ compared to the full model.

Table 6. Ablation study results for Mamba-DQN in LunarLander-v3. t -test results are computed against the full model as the reference. A dash (–) indicates convergence failure or an inapplicable statistical comparison.

Variant	Max	Min	Mean $\pm$ Std	t -Stat	p -Value	Cohen's d
Full Model	236.56	-15.33	139.46 ± 109.78	–	–	–
No PER	-138.92	-288.45	-185.40 ± 60.09	-5.804	0.00102	-3.671
Encoder Removed	2.68	-56.02	-12.02 ± 24.95	-3.009	0.0349	-1.903

The removal of the Mamba-SSM encoder also reduces the performance significantly ($p < 0.05$), highlighting that both prioritized sampling and the encoder contribute to 1D signal-based tasks.

Table 7 shows that the removal of the PER results in only a minor statistically insignificant change ($p = 0.374$), indicating that the PER has less influence in this environment. However, removing the Mamba-SSM encoder produces a significant decrease in performance ($p < 0.05$), confirming the encoder's role in achieving higher rewards in high-speed navigation tasks.

Table 7. Ablation study results for Mamba-DQN in Highway-fast-v0. t -test results are computed against the full model as the reference. A dash (–) indicates convergence failure or an inapplicable statistical comparison.

Variant	Max	Min	Mean $\pm$ Std	t -Stat	p -Value	Cohen's d
Full Model	24.08	18.20	20.99 ± 2.34	–	–	–
No PER	21.35	16.22	19.68 ± 2.03	-0.943	0.3736	-0.597
Encoder Removed	18.72	14.55	16.64 ± 1.63	-3.410	0.0110	-2.157

Across all environments, the Mamba-SSM encoder emerges as the most critical component, with its removal consistently resulting in statistically significant performance degradation and large negative effect sizes.

While the PER contributes substantially in environments with sparse or high-variance rewards such as LunarLander-v3, its impact is less pronounced in highway-driving scenarios.

In the image-based Atari 2600 Pong task, both the encoder and the use of a full sequence replay buffer are essential for successful learning, with their removal leading to severe performance drops or complete training failure.

These findings validate that both architectural design and sampling strategy are environment-dependent and that the proposed full Mamba-DQN configuration offers the most robust performance across diverse task types.

4.4. Summary and Implications

Based on the comprehensive experimental evaluation, including both comparative and ablation studies, the results demonstrate that Mamba-DQN achieves strong performance, often outperforming baseline models across diverse environments. Although the performance gains were not universally statistically significant, Mamba-DQN exhibited clear advantages, with most configurations showing statistically significant improvements ($p < 0.05$) in both smoothed average reward and TD loss.

These performance gains stem from the architectural characteristics of Mamba-DQN. The Mamba-based time-series encoder efficiently compresses temporal dependencies, even with short input sequences, while the use of full state sequence re-encoding at each training step ensures temporal representations remain consistent with the current network parameters. This design promotes coherent Q-value estimation and mitigates representational drift during training.

Consequently, Mamba-DQN shows strong applicability to dynamic environments with frequent state transitions, where temporal alignment and accurate value estimation are critical for policy convergence. In summary, the proposed architecture integrates temporal abstraction, structural consistency, and training efficiency, offering strong potential for effective and transferable reinforcement learning performance.

5. Discussion

The empirical results demonstrate the potential of Mamba-DQN to improve policy learning efficiency, accelerate convergence, and enhance overall performance across reinforcement learning tasks with diverse temporal dynamics and reward structures.

The proposed architecture often outperformed baseline models, achieving faster convergence and lower TD loss in most environments without requiring additional hyperparameter tuning.

These findings indicate that Mamba-DQN possesses strong robustness, generalization capability, and adaptability to a broad range of environments, including those demanding rapid decision-making or long-term credit assignment.

Analysis of Architectural Trade-Offs in Temporal Modeling

A primary advantage of Mamba-DQN lies in the linear-time complexity of its Mamba-SSM encoder [29]. This contrasts sharply with conventional temporal models used in reinforcement learning.

For a sequence of length L and hidden dimension d , LSTM-based models have a complexity of $O(Ld^2)$ due to their sequential operations, which limit parallelism. Transformer-

based models exhibit a complexity of $O(L^2d)$ owing to the quadratic cost of global self-attention.

In contrast, Mamba-SSM achieves linear-time complexity $O(Ld)$ via a selective scan mechanism that permits parallel computation akin to convolutions, while preserving temporal recurrence.

This efficiency, summarized in Table 8, makes Mamba-DQN particularly suitable for real-time environments and long-sequence reinforcement learning tasks.

Table 8. Comparative time complexity of temporal encoders.

Model Architecture	Time Complexity	Key Characteristic
LSTM-DQN	$O(Ld^2)$	Sequential computation, limited parallelism
Transformer-DQN	$O(L^2d)$	Quadratic cost from global attention
Mamba-DQN (Ours)	$O(Ld)$	Linear-time selective scan

Beyond computational efficiency, Mamba-DQN promotes faster convergence by generating compressed temporal representations. The Mamba encoder compresses a history of inputs $s_{t-T+1}, \dots, s_t$ into a latent vector z_t that preserves essential temporal context while reducing inconsistencies arising from frequent state transitions. This produces smoother state representations, which in turn yield accurate Q-value predictions $Q(z_t, a_t; \theta)$ and TD targets y_t .

As a result, the variance in the temporal difference (TD) error δ_t is reduced, leading to more efficient learning. Lower variance in the TD signal enables gradient updates with reduced fluctuations, contributing to faster convergence during training.

Empirical evidence for this effect is provided in Figures 5, 7, and 9, where the proposed Mamba-DQN exhibits consistently lower TD loss and achieves higher rewards with fewer training steps compared to baseline models. This supports the claim that the latent-state filtering mechanism contributes to both improved sample efficiency and faster convergence of the learned policy.

While LSTM-DQN and Transformer-DQN both incorporate mechanisms to capture temporal dependencies—namely, gated recurrence and global self-attention—their architectural properties entail certain trade-offs that affect their adaptability to different temporal settings.

In particular, when short sequences are used as input, both architectures face representational constraints. LSTM utilizes gated mechanisms to propagate sequential information through hidden states, but in short-horizon settings, it may retain redundant or outdated information, thereby limiting the efficiency of its state encoding [30,31]. Likewise, the Transformer’s self-attention uniformly aggregates signals across all time steps, which can dilute the influence of recent transitions that are more critical for near-term policy updates [32,33]. These limitations may reduce the model’s ability to extract task-relevant temporal patterns when immediate decision-making is required.

Nevertheless, it is worth noting that in the LunarLander-v3 environment, LSTM-DQN exhibited performance comparable to that of Mamba-DQN, suggesting that its gating mechanism can effectively capture recent temporal dependencies in certain short-horizon settings.

Mamba-DQN addresses the structural limitations of conventional time-series encoder-based DQN architectures by incorporating the following key features required for effective temporal processing across diverse environments:

- First, the linear-time computational structure and selective state summarization mechanism enable efficient compression and representation of critical information, even with short input sequences.

- Second, the enhanced temporal representation capability contributes to improved training stability and accelerated policy convergence, resulting in notable improvements in average reward levels and convergence speed in most tested scenarios.
- Third, Mamba-DQN demonstrated strong generalization capabilities by achieving robust performance across three diverse experimental environments—a high-speed driving simulation (Highway-fast-v0), a complex control task with sparse rewards (LunarLander-v3), and a high-dimensional visual domain (Atari 2600 Pong).

It should be noted that this study was conducted exclusively within controlled simulator-based environments. Real-world factors such as sensor uncertainty, physical system constraints, and external environmental noise were not considered. Future work will focus on validating the proposed Mamba-DQN in real-world settings to further assess its practical applicability.

6. Conclusions

This study proposed Mamba-DQN, an RL architecture that integrates a Mamba-SSM-based time-series encoder into the DQN framework. The architecture was designed to address the limitations of LSTM networks and Transformer-based models by enabling effective temporal information compression within short sequences and enhancing computational efficiency during policy estimation.

Comprehensive experiments were conducted in the Highway-fast-v0, LunarLander-v3, and Atari 2600 Pong environments. Across most settings, Mamba-DQN achieved higher maximum rewards, lower TD loss, and, in many cases, superior average rewards compared to baseline models, including DQN, LSTM-DQN, and Transformer-DQN. While LunarLander-v3 showed a slightly higher average reward for LSTM-DQN, Mamba-DQN maintained a clear advantage in maximum performance, indicating a higher potential policy ceiling. The selective state summarization mechanism and the linear-time computational structure of the Mamba encoder enabled efficient temporal information compression under short sequence conditions. These properties improved the training stability, accelerated the policy convergence, and facilitated faster policy learning.

These results provide empirical evidence that efficient temporal encoding plays a critical role in enhancing policy learning performance. Mamba-DQN offers a structurally simple yet computationally efficient framework with strong temporal representation capability, making it particularly suitable for control environments characterized by frequent state transitions and real-time response requirements. The method also provides an alternative to existing LSTM- or Transformer-based time-series encoders, which often suffer from information dilution or over-memory effects under short sequence conditions.

Despite these strengths, limitations remain. A key distinction must be made between training stability and policy stability. While Mamba-DQN demonstrated stable learning within a single run, evidenced by smooth TD loss reduction, its policy stability across multiple runs was relatively low. This was shown by the noticeable variance in final performance across runs, indicating that the quality of the converged policy was not consistent. Moreover, all evaluations were conducted in simulated environments, leaving open questions about robustness to real-world challenges such as sensor noise, environmental variability, and partial observability.

Future work will focus on the following:

- Deploying Mamba-DQN in physical systems (e.g., autonomous driving, UAV navigation) to evaluate real-world performance.
- Testing robustness under environmental disturbances, including noise, lighting changes, and dynamic obstacles.

- Investigating strategies to further stabilize policy learning while preserving high-reward potential.

Author Contributions: Conceptualization, H.R., C.-B.S. and D.-Y.K.; Methodology, H.R., C.-B.S. and D.-Y.K.; Software, H.R. and D.-Y.K.; Validation, C.-B.S. and D.-Y.K.; Formal analysis, H.R., C.-B.S. and D.-Y.K.; Investigation, H.R., C.-B.S. and D.-Y.K.; Resources, C.-B.S. and D.-Y.K.; Data curation, H.R.; Writing—original draft preparation, H.R.; Writing—review and editing, C.-B.S. and D.-Y.K.; Visualization, H.R.; Supervision, C.-B.S. and D.-Y.K.; Project administration, C.-B.S.; Experimental environment support, C.-B.S. and D.-Y.K. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The environments used in this study are publicly available. Highway-fast-v0 is accessible at <https://github.com/Farama-Foundation/HighwayEnv> (accessed on 30 June 2025), and LunarLander-v3 is accessible at https://gymnasium.farama.org/environments/box2d/lunar_lander/ (accessed on 30 June 2025). No new datasets were generated or analyzed during the current study.

Acknowledgments: The present research was conducted under the Research Grant of Kwangwoon University in 2024.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Zhu, Y.; Wan Hasan, W.Z.; Harun Ramli, H.R.; Norsahperi, N.M.H.; Mohd Kassim, M.S.; Yao, Y. Deep Reinforcement Learning of Mobile Robot Navigation in Dynamic Environment: A Review. *Sensors* **2025**, *25*, 3394. [\[CrossRef\]](#)
2. Park, M.; Park, C.; Kwon, N.K. Autonomous Driving of Mobile Robots in Dynamic Environments Based on Deep Deterministic Policy Gradient: Reward Shaping and Hindsight Experience Replay. *Biomimetics* **2024**, *9*, 51. [\[CrossRef\]](#) [\[PubMed\]](#)
3. Pookkuttath, S.; Gomez, B.F.; Elara, M.R. RL-Based Vibration-Aware Path Planning for Mobile Robots' Health and Safety. *Mathematics* **2025**, *13*, 913. [\[CrossRef\]](#)
4. AlMahamid, F.; Grolinger, K. VizNav: A Modular Off-Policy Deep Reinforcement Learning Framework for Vision-Based Autonomous UAV Navigation in 3D Dynamic Environments. *Drones* **2024**, *8*, 173. [\[CrossRef\]](#)
5. Yan, S.; Zhu, Y.; Chen, W.; Zhang, J.; Zhu, C.; Chen, Q. Dynamic Obstacle Avoidance for Robotic Arms Using Deep Reinforcement Learning with Adaptive Reward Mechanisms. *Appl. Sci.* **2025**, *15*, 4496. [\[CrossRef\]](#)
6. Salehi, A.; Hosseinpour, S.; Tabatabaei, N.; Soltani Firouz, M.; Yu, T. Intelligent Navigation of a Magnetic Microrobot with Model-Free Deep Reinforcement Learning in a Real-World Environment. *Micromachines* **2024**, *15*, 112. [\[CrossRef\]](#)
7. Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A.A.; Veness, J.; Bellemare, M.G.; Graves, A.; Riedmiller, M.; Fidjeland, A.K.; Ostrovski, G.; et al. Human-Level Control through Deep Reinforcement Learning. *Nature* **2015**, *518*, 529–533. [\[CrossRef\]](#)
8. Yang, S.; Sun, D.; Chen, X. Off-Policy Temporal Difference Learning with Bellman Residuals. *Mathematics* **2024**, *12*, 3603. [\[CrossRef\]](#)
9. Yang, Y.; Xi, M.; Dai, H.; Wen, J.; Yang, J. Z-Score Experience Replay in Off-Policy Deep Reinforcement Learning. *Sensors* **2024**, *24*, 7746. [\[CrossRef\]](#)
10. Yang, H.; Park, H.; Lee, K. A Selective Portfolio Management Algorithm with Off-Policy Reinforcement Learning Using Dirichlet Distribution. *Axioms* **2022**, *11*, 664. [\[CrossRef\]](#)
11. Hutsebaut-Buysse, M.; Mets, K.; Latré, S. Hierarchical Reinforcement Learning: A Survey and Open Research Challenges. *Mach. Learn. Knowl. Extr.* **2022**, *4*, 172–221. [\[CrossRef\]](#)
12. Zhang, Y.; Chen, P. Path Planning of a Mobile Robot for a Dynamic Indoor Environment Based on an SAC-LSTM Algorithm. *Sensors* **2023**, *23*, 9802. [\[CrossRef\]](#) [\[PubMed\]](#)
13. Alcayaga, J.M.; Menéndez, O.A.; Torres-Torriti, M.A.; Váscquez, J.P.; Arévalo-Ramírez, T.; Romo, A.J.P. LSTM-Enhanced Deep Reinforcement Learning for Robust Trajectory Tracking Control of Skid-Steer Mobile Robots Under Terra-Mechanical Constraints. *Robotics* **2025**, *14*, 74. [\[CrossRef\]](#)
14. Chen, Q.; Wang, R.; Lyu, M.; Zhang, J. Transformer-Based Reinforcement Learning for Multi-Robot Autonomous Exploration. *Sensors* **2024**, *24*, 5083. [\[CrossRef\]](#)

15. Zhao, R.; Fan, Y.; Li, Y.; Zhang, D.; Gao, F.; Gao, Z.; Yang, Z. Knowledge Distillation-Enhanced Behavior Transformer for Decision-Making of Autonomous Driving. *Sensors* **2025**, *25*, 191. [\[CrossRef\]](#)
16. Yang, Z.; Wu, Z.; Wang, Y.; Wu, H. Deep Reinforcement Learning Lane-Changing Decision Algorithm for Intelligent Vehicles Combining LSTM Trajectory Prediction. *World Electr. Veh. J.* **2024**, *15*, 173. [\[CrossRef\]](#)
17. Zhang, X.; Guo, H.; Yan, T.; Wang, X.; Sun, W.; Fu, W.; Yan, J. Penetration Strategy for High-Speed Unmanned Aerial Vehicles: A Memory-Based Deep Reinforcement Learning Approach. *Drones* **2024**, *8*, 275. [\[CrossRef\]](#)
18. Hausknecht, M.J.; Stone, P. Deep Recurrent Q-Learning for Partially Observable MDPs. In Proceedings of the AAAI Fall Symposium, Arlington, VA, USA, 12–14 November 2015; Volume 45, p. 141.
19. Hochreiter, S.; Schmidhuber, J. Long Short-Term Memory. *Neural Comput.* **1997**, *9*, 1735–1780. [\[CrossRef\]](#)
20. Esslinger, K.; Platt, R.; Amato, C. Deep Transformer Q-Networks for Partially Observable Reinforcement Learning. *arXiv* **2022**, arXiv:2206.01078. [\[CrossRef\]](#)
21. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, Ł.; Polosukhin, I. Attention Is All You Need. In Proceedings of the 31st International Conference on Neural Information Processing System, Long Beach, CA, USA, 4–9 December 2017.
22. Ma, X.; Huang, C.; Huang, X.; Wu, W. Mamba-DQN: Adaptively Tunes Visual SLAM Parameters Based on Historical Observation DQN. *Appl. Sci.* **2025**, *15*, 2950. [\[CrossRef\]](#)
23. Leurent, E. An Environment for Autonomous Driving Decision-Making. *GitHub Repos.* 2018. Available online: <https://github.com/eleurent/highway-env> (accessed on 30 June 2025).
24. Brockman, G.; Cheung, V.; Pettersson, L.; Schneider, J.; Schulman, J.; Tang, J.; Zaremba, W. OpenAI Gym. *arXiv* **2016**, arXiv:1606.01540. [\[CrossRef\]](#)
25. Towers, M.; Kwiatkowski, A.; Terry, J.; Balis, J.U.; De Cola, G.; Deleu, T.; Goulão, M.; Kallinteris, A.; Krimmel, M.; KG, A.; et al. Gymnasium: A Standard Interface for Reinforcement Learning Environments. *arXiv* **2024**, arXiv:2407.17032. [\[CrossRef\]](#)
26. Kim, C. Target-Network Update Linked with Learning Rate Decay Based on Mutual Information and Reward in Deep Reinforcement Learning. *Symmetry* **2023**, *15*, 1840. [\[CrossRef\]](#)
27. Kubovčík, M.; Dirgová Luptáková, I.; Pospíchal, J. Signal Novelty Detection as an Intrinsic Reward for Robotics. *Sensors* **2023**, *23*, 3985. [\[CrossRef\]](#)
28. Schaul, T.; Quan, J.; Antonoglou, I.; Silver, D. Prioritized Experience Replay. *arXiv* **2015**, arXiv:1511.05952. [\[CrossRef\]](#)
29. Gu, A.; Dao, T. Mamba: Linear-Time Sequence Modeling with Selective State Spaces. *arXiv* **2023**, arXiv:2312.00752. [\[CrossRef\]](#)
30. Zhang, S.; Zhao, Z.; Wu, J.; Jin, Y.; Jeng, D.-S.; Li, S.; Li, G.; Ding, D. Solving the Temporal Lags in Local Significant Wave Height Prediction with a New VMD-LSTM Model. *Ocean Eng.* **2024**, *313*, 119385. [\[CrossRef\]](#)
31. Ryu, G.-A.; Chuluunsai Khan, T.; Nasridinov, A.; Rah, H.; Yoo, K.-H. SCE-LSTM: Sparse Critical Event-Driven LSTM Model with Selective Memorization for Agricultural Time-Series Prediction. *Agriculture* **2023**, *13*, 2044. [\[CrossRef\]](#)
32. Dong, Z.; Zhao, Y.; Wang, A.; Zhou, M. Wind-Mambaformer: Ultra-Short-Term Wind Turbine Power Forecasting Based on Advanced Transformer and Mamba Models. *Energies* **2025**, *18*, 1155. [\[CrossRef\]](#)
33. Bing, Q.; Zhao, P.; Ren, C.; Wang, X.; Zhao, Y. Short-Term Traffic Flow Forecasting Method Based on Secondary Decomposition and Conventional Neural Network–Transformer. *Sustainability* **2024**, *16*, 4567. [\[CrossRef\]](#)

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.