SPIKING NEURON AS DISCRETE GATING FOR LONG-TERM MEMORY TASKS

Anonymous authors
Paper under double-blind review

ABSTRACT

Efficient long-term memory is important for improving the sample efficiency of Partially Observable Reinforcement Learning. In memory-based RL methods, the long-term memory capacity relies on the sequence models used in agent architecture. Two main approaches improve long-term dependency for sequence models, using linear recurrence and using information selection mechanisms such as gating. However, the sample efficiency of existing approaches remains low in long-term memory tasks. In this paper, we first present a saliency-based framework to illustrate why existing methods do not perform well on long-term memory tasks. Specifically, they cannot effectively filter out noisy information irrelevant to the memory task in the early stage of training. To this end, we design a novel linear recurrent module, in which the gating is controlled by spiking neurons. Spiking neurons output discrete values and can more effectively mask noise in the early stages of training, thus improving sample efficiency. The effectiveness of our proposed module is demonstrated on Passive Visual Match, a classic long-term memory task, and several different types of partially observable tasks. The code is attached in the supplementary material and will be made publicly available.

1 INTRODUCTION

Long-term memory is an important ability for humans to make complex decisions in environments Kumaran et al. (2016). Specifically, humans can recall historical information for decision-making, and humans can prevent memory from being interfered with by other events that are not relevant to the memory-intensive task McNab & Klingberg (2008). For Reinforcement Learning (RL) agents, achieving efficient long-term memory will improve the sample efficiency of their training, which is important for their practical real-life applications Dulac-Arnold et al. (2019).

The computational and memory efficiency of recurrent neural network (RNN)-based sequence models Parisotto et al. (2020); Zhao et al. (2023); Morad et al. (2023b); Le et al. (2024) makes them well-suited for memory-based RL, particularly in real-world applications that involve near-infinite temporal contexts, as shown in Figure 1 (Left). These RNN-based sequence models involve two essential mechanisms: Linear recurrence improves the long-term dependency Orvieto et al. (2023) of the sequence model, while gating Hochreiter & Schmidhuber (1997); Chung et al. (2014) explicitly selects relevant information for the working memory Chaudhari et al. (2021); Morad et al. (2023b). However, neither of these mechanisms thoroughly prevents working memory from being interfered with by irrelevant information in the early stages of the training process, leading to unstable training and inefficiency in long-term memory tasks, as mentioned by recent studies Lu et al. (2024); Le et al. (2024) and evidenced by the results shown in Figure 1 (Right).

In this paper, we introduce a unified framework based on the signal-to-noise ratio (SNR) of temporal saliency Ismail et al. (2019), from the perspective of gradient analysis of backpropagation, to evaluate the long-term memory effectiveness of sequence models. This framework not only explains the necessity of linear and gating mechanisms but also reveals why linear and existing continuous gating designs still fall short in supporting effective long-term memory learning (see Section 3). To overcome this limitation, we justify the importance of discrete gating mechanism under the proposed unified framework, designing a linear recurrent cell equipped with such discrete gating (implemented by spiking neurons Fang et al. (2023)). Extensive experiments on long-term memry tasks demonstrate the superiority of our approach, outperforming several SOTA methods. Moreover, in more general

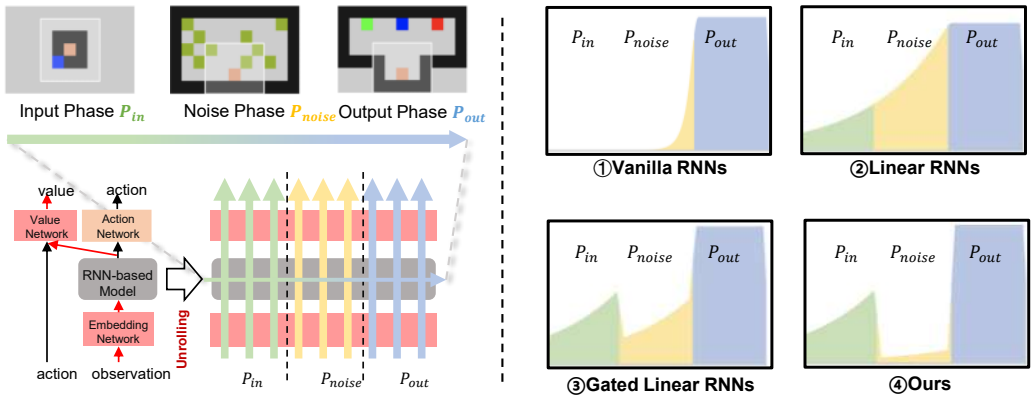

Figure 1: **Left Top:** Illustration of a classical long-term memory task, Passive Visual Match Hung et al. (2019) (images extracted from Ni et al. (2023)). According to Hung et al. (2019), it can be divided into input, noise, and output phases. The agent needs to utilize the information remembered in input phase P_{in} for decision-making in output phase P_{out}, *i.e.*, success in this task requires the agent to remember the color from P_{in} and correctly match it to the color in P_{out}. Noise phase P_{noise} is an interference task and could be arbitrarily long. **Left Botton:** To align with the three temporal phases described above, we unfold the RNN-based model framework. Intuitively, the contribution from P_{in} to P_{out} must be substantially greater than that from P_{noise} to P_{out}. This ensures that the model effectively learns and retains as much information from P_{in} during training. **Right:** We evaluate and compare the temporal contributions of different phases to the output phase P_{out} across several existing RNN-based models, including vanilla Recurrent Neural Networks (RNNs) Elman (1990), linear RNNs Orvieto et al. (2023)), and gated linear RNNs Morad et al. (2023b). For each subfigure, the horizontal axis represents the temporal progression which is consistent with that in Figure 1 (Left), while the vertical axis quantifies the contribution of each of the three phases to P_{out}. The contributions are calculated based on temporal saliency Ismail et al. (2019) (or Equation 3), which measures the influence of inputs at each time step on the final output. The results reveal several key patterns: linear RNNs exhibit a slower decay in contribution over time compared to vanilla RNNs, while gated RNNs display abrupt shifts of saliency across different phases, contrasting with the smooth and gradual evolution observed in their non-gated counterparts. Critically, none of the existing methods effectively suppress the influence of the noise phase (indicated by the notable height of yellow regions). In contrast, the proposed approach successfully mitigates the impact of noise phase, leading to more effective temporal credit assignment.

scenarios that rely primarily on short-term memory, our method achieves performance comparable to existing SOTA models. Our contributions are summarized as follows:

- We propose a saliency-based metric using SNR of temporal saliency to analyze RNN-based sequence models in long-term memory tasks, showing that their inefficiency stems from an inability to suppress irrelevant information.
- We design a discrete gating mechanism using spiking neurons within a linear recurrent cell, which improves noise filtering and sample efficiency in long-term memory tasks, while maintaining parallel training capability.
- We empirically validate the framework's insights and our model's strong performance across long-term and general memory scenarios.

2 Background

2.1 Partially Observable Reinforcement Learning

POMDP. Partially Observable Markov Decision Process (POMDP) is a mathematical framework that is used to model the environment that is partially observable by an agent Åström (1965). It formally describes the problem of long-term memory. Since the agent cannot directly perceive the

true state of the environment, it must maintain an internal memory of past observations and actions to make optimal decisions. Thus, effectively solving a POMDP inherently requires the ability to perform long-term memory tasks. A POMDP is defined as a tuple $(\mathcal{S}, \mathcal{A}, \mathcal{O}, T, O, R, \gamma)$, where $\mathcal{S}$ represents the state space, which is inaccessible to agents. $\mathcal{A}$ represents the action space. $\mathcal{O}$ represents the observation space of agents. $T = P(s_t|s_{t-1}, a_{t-1})$ is the transition function. $O = P(o_t|s_t)$ is the observation function. $R = P(r_{t+1}|s_t, a_t, s_{t+1})$ is the reward function. γ is the discount factor. To obtain an optimal policy, the agent must condition the policy on all accessible information *i.e.*, the trajectory $\tau_t = (o_1, a_1, r_1, \cdots, o_t, a_t, r_t)$ up to timestep t. The agent's goal is to learn a policy $\pi(a_t|\tau_t)$ to maximize the expected discounted return, where T is the time horizon.

Memory-Based RL. Early approaches for solving POMDPs are based on belief states Kaelbling et al. (1998), which require a lot of computations. Recently, model-free methods have been verified as a general and efficient approach Hausknecht & Stone (2015); Ni et al. (2021). As shown in Figure 1 (Left), the agent uses an MLP encoder and a sequence model to map the trajectory τ_t to a latent Markov state $\hat{s}_t$. Empirical studies Morad et al. (2023a) have found that traditional RNNs Hochreiter & Schmidhuber (1997); Chung et al. (2014) are sufficiently useful in simple POMDP environments. Recently, increasing amounts of research have focused on the use of modern sequence models to tackle more challenging POMDP tasks. Some methods use Transformer Ni et al. (2023), or State Space Models (SSMs) Lu et al. (2023), or linear RNNs Lu et al. (2024). Some methods modify the structure of linear RNNs Morad et al. (2023b) or Memory-Augmented Neural Networks (MANN) Le et al. (2024) to make them perform better in POMDP tasks. Some studies introduce more efficient training algorithms Morad et al. (2024); Elelimy et al. (2024) for certain types of sequence models. In this paper, we focus on RNNs-based sequence model for Memory-Based RL.

2.2 Recurrent Neural Networks

RNNs capture the internal temporal dependency of sequential data. Early RNNs Elman (1990) trained with the BPTT algorithm suffer from the gradient vanishing problem Hochreiter (1998), which will hinder the model from learning long-term dependency.

Gated RNNs. Gating mechanisms that control the flow of information are widely found in the human brain Gisiger & Boukadoum (2011). A similar design exists in sequence models as gated RNNs. Gated RNNs Hochreiter & Schmidhuber (1997); Chung et al. (2014); Lei et al. (2017) use gating to alleviate the gradient vanishing problem and thus increase the length of temporal dependency, but their sequential nature slows down the training process with a time complexity of $O(T)$, making it difficult to train effectively on tasks with a very long period of time.

Linear RNNs. Linear RNNs Orvieto et al. (2023); Feng et al. (2024) or SSMs Smith et al. (2022); Gu & Dao (2023) reduce the vanishing gradient by removing the non-linear activation function in the recurrence structure while keeping the other layers non-linear. The linear recurrence can be trained in parallel using convolution with a causal mask or the associative scan operation. To preserve their representational power when non-linearity is removed, many Linear RNNs use complex values to extend their hidden state.

This paper provides a unified framework based on temporal saliency Ismail et al. (2019) to analyze different variants of RNNs and shows their limitations when handling long-term memory tasks.

2.3 Discrete models and selection mechanism

Discrete models. Discrete models use 0 or 1 to represent the data itself. Works related to discrete models include Binary Neural Networks (BNNs) Qin et al. (2020) and Spiking Neural Networks (SNNs) Roy et al. (2019), which employ the Straight-Through Estimator (STE) or Surrogate Gradients (SG) to facilitate gradient-based learning. These discrete models have many applications, such as image classification Rathi & Roy (2020); Fang et al. (2021b); Yao et al. (2022), sequence modeling Fang et al. (2023); Li et al. (2024); Chen et al. (2024), and RL Chen et al. (2022); Zhang et al. (2024); Qin et al. (2022; 2025). Their primary advantage lies in reducing computational costs, and they also serve as foundational model architectures for the discrete selective mechanism.

Discrete selection mechanism. The discrete selection mechanism uses 0 or 1 to decide whether to retain or discard certain information. While this mechanism already has many applications, there may be additional, yet-to-be-explored benefits. Sequence models such as SkipRNN Campos et al.

(2017) and Phased LSTM Neil et al. (2016) utilize discrete selection mechanisms, significantly reducing network computation while achieving superior performance compared to traditional RNNs on supervised tasks. In other domains, Spiking-NeRF Liao et al. (2024) leverages spiking neurons, the fundamental units of SNNs, as an information selection mechanism for NeRF Mildenhall et al. (2021), functioning similarly to a gating mechanism that forms a discontinuous representation space and filters out irrelevant information.

However, almost no prior work has investigated whether discrete model or discrete selection mechanisms can effectively remove irrelevant information in RNN-based RL tasks. To the best of our knowledge, our method is the first to explore discrete gating in memory-based RL tasks. A more detailed comparison with existing works across different aspects is provided in Appendix B.

3 Long-term Memory Efficiency Analysis

3.1 Long-term Memory Efficiency Analysis Framework

According to Hung et al. (2019), timesteps in a long-term memory task can be divided into input, noise, and output phases. We define these phases as sets of timesteps, denoted as P_{in}, P_{noise}, and P_{out}. P_{in} contains the timesteps in which the agents are receiving useful information from the environment and updating their memory, while the actions in P_{in} will not receive immediate reward feedback from the environment. P_{out} contains the timesteps when the agent is required to recall the information that is received from P_{in} to obtain the optimal policy. Timesteps in P_{noise} are those with observations that are useless for inferring actions or values in P_{out}, and the timesteps in P_{noise} can be arbitrarily long. Observations in P_{noise} can be considered noisy inputs for the memory task. Despite this, there may be some memory-agnostic tasks in P_{noise}.

Our framework focuses on the value network since it plays a crucial role in many Deep RL algorithms Schulman et al. (2017); Haarnoja et al. (2018); Fujimoto et al. (2018). Its output is denoted as $Q(\hat{s}_t, a_t)$. To obtain good performance of the model, the contribution of observation $o_t, t \in P_{in}$ to value $Q(\hat{s}_t, a_t), t \in P_{out}$, denoted as $\mathcal{C}_{in\to out}$, should be substantially greater than that of $\mathcal{C}_{noise\to out}$ (the contribution of observation $o_t, t \in P_{noise}$ to $Q(\hat{s}_t, a_t), t \in P_{out}$). Based on this observation, we define the SNR of temporal saliency for long-term memory efficiency analysis:

$$\text{SNR}_{\mathcal{C}} = \frac{\mathcal{C}_{in\to out}}{\mathcal{C}_{in\to out} + \mathcal{C}_{noise\to out}}, \mathcal{C}_{in\to out} \geq 0, \mathcal{C}_{noise\to out} \geq 0. \tag{1}$$

The ratio of $\mathcal{C}_{in\to out}$ to $\mathcal{C}_{noise\to out}$ can be reflected by $\text{SNR}_{\mathcal{C}}$, which is bounded in $(0, 1)$. The larger the value of $\text{SNR}_{\mathcal{C}}$, the better a memory-based RL agent can eliminate irrelevant information. To make that happen, we could **either increase $\mathcal{C}_{in\to out}$ or decrease $\mathcal{C}_{noise\to out}$**. We can quantify these two terms as the sum of the temporal saliency Ismail et al. (2019) of their respective phases:

$$\mathcal{C}_{in\to out} = \frac{1}{|P_{in}|} \sum_{t\in P_{in}} R_t, \mathcal{C}_{noise\to out} = \frac{1}{|P_{noise}|} \sum_{t\in P_{noise}} R_t. \tag{2}$$

Formally, we define temporal saliency Ismail et al. (2019) as the partial derivative of the output of the value network in P_{out} with respect to observation o_t. For timestep t, the temporal saliency R_t is:

$$R_t = \Big| \sum_{i\in P_{out}} \frac{\partial Q(\hat{s}_i, a_i)}{\partial o_t} \Big| = \Big| \sum_{i\in P_{out}} \Big[\frac{\partial Q(\hat{s}_i, a_i)}{\partial \hat{s}_i} \underbrace{\frac{\partial \hat{s}_i}{\partial h_i} \Big(\prod_{j=t+1}^{i} \frac{\partial h_j}{\partial h_{j-1}} \Big) \frac{\partial h_t}{\partial x_t}}_{G_t} \frac{\partial x_t}{\partial o_t} \Big] \Big|, \tag{3}$$

where $Q(\hat{s}_i, a_i)$ is the output of the value network at timestep $i \in P_{out}$, $\hat{s}$ and h are the output and hidden state of the sequence model, respectively. x_t is the encoded feature of o_t. The value of R_t reflects the contribution of the input in the timestep t to the output in P_{out}. G_t is the term that is dependent on the structure of the sequence model; it can establish a connection between the structure of the sequence model and $\text{SNR}_{\mathcal{C}}$.

Temporal saliency Ismail et al. (2019) is introduced to interpret the behavior of RNN Ismail et al. (2019) and has recently been adapted as a memory introspection tool in POMDPs Wang et al. (2025). However, Wang et al. (2025) only uses it for qualitative memory visualization, without distinguishing

input, noise, and output phases. As such, it lacks the quantitative measures needed to analytically assess the effectiveness of specific model designs. In contrast, the proposed $\text{SNR}_{\mathcal{C}}$ explicitly models this distinction, enabling the evaluation of architectural choices.

3.2 Long-Term Memory Efficiency Analysis for Different RNN-based Models

In this section, we will show how existing sequence models with linear recurrence or gating mechanisms essentially devise a strategy to increase $\text{SNR}_{\mathcal{C}}$ by either increasing $\mathcal{C}_{in\to out}$ or decreasing $\mathcal{C}_{noise\to out}$ with their specific gradient dynamics.

Non-linear RNNs vs Linear RNNs. A unified formulation of h_t in Non-linear and Linear RNNs can be formulated as:

$$h_t = f(\mathbf{W}_h h_{t-1} + \mathbf{W}_x x_t), \quad h_t, x_t \in \mathbb{R}^{N_h}, \tag{4}$$

where x_t is the input of the RNN and N_h is the hidden size.

For a Non-linear RNN such as vanilla RNN Elman (1990), $f(\cdot)$ is usually $\tanh(\cdot)$. $\mathbf{W}_x \in \mathbb{R}^{N_x \times N_h}$ and $\mathbf{W}_h \in \mathbb{R}^{N_h \times N_h}$ are matrices of learnable parameters. With these components in the recurrence, the term $\prod_{j=t+1}^{i} \frac{\partial h_j}{\partial h_{j-1}}$ in R_t, Equation 3, will decay rapidly, causing the gradient to vanish Hochreiter (1998). It means that if the timestep in P_{in} is in the distant past, $\mathcal{C}_{in\to out} = \frac{1}{|P_{in}|}\sum_{t\in P_{in}} R_t$ would be relatively small, as shown in Figure 1 (Right, ①).

In contrast, Linear RNNs, such as LRU Orvieto et al. (2023); Gu et al. (2020), use identity mapping as $f(\cdot)$ and replace $\mathbf{W}_h h_{t-1}$ in Equation 4 by $c_t \odot h_{t-1}$, where c_t is a vector of size N_h and $\odot$ indicates the element-wise product. With linearity, the term $\prod_{j=t+1}^{i} \frac{\partial h_j}{\partial h_{j-1}}$ in Equation 3 will decay slower as $|i-t|$ increases. Therefore, the gradient can propagate to earlier timesteps. Once timesteps in P_{in} are distant from P_{out} in long-term memory tasks, $\mathcal{C}_{in\to out}$ in Equation 2 can be higher than that in vanilla RNNs, thus increasing the value of $\text{SNR}_{\mathcal{C}}$ in Equation 1, as shown in Figure 1 (Right, ②).

Non-gated RNNs vs Gated RNNs. Our analysis above indicates that linear recurrence could increase $\mathcal{C}_{in\to out}$ by slowing down the decay of temporal saliency R_t. However, both linear and non-linear RNNs cannot decrease $\mathcal{C}_{noise\to out}$ to eliminate the impact from P_{noise}. We show that Gated RNNs such as Morad et al. (2023b) increase the long-term efficiency by decreasing $\mathcal{C}_{noise\to out}$.

A unified formulation of h_t in Non-gated and Gated RNNs can be formulated as:

$$\begin{aligned} h_t &= c_t \odot h_{t-1} + (\mathbf{W}_x x_t) \odot \phi. \\ \phi &= \begin{cases} 1, & \text{non-gated,} \\ \sigma(\mathbf{W}_i x_t), & \text{gated,} \end{cases} \end{aligned} \tag{5}$$

where $\sigma(\cdot)$ is the sigmoid function, defined as $\sigma(x) = \frac{1}{1-e^{-x}}, x \in \mathbb{R}$. $\mathbf{W}_i \in \mathbb{R}^{N_x \times N_h}$ is the linear layer of the input gate. The partial derivative of h_t with respect to x_t in Equation 3 can be written as

$$\frac{\partial h_t}{\partial x_t} = \begin{cases} \text{diag}\big(\frac{\partial h_t}{\partial u_t}\big)\frac{\partial u_t}{\partial x_t}, & \text{non-gated,} \\ \text{diag}\big(\sigma(g_t) \odot \frac{\partial h_t}{\partial u_t}\big)\frac{\partial u_t}{\partial x_t} + \text{diag}\big(u_t \odot \frac{\partial h_t}{\partial \sigma(g_t)} \odot \sigma'(g_t)\big)\frac{\partial g_t}{\partial x_t}, & \text{gated,} \end{cases} \tag{6}$$

where $u_t = \mathbf{W}_x x_t$, $g_t = \mathbf{W}_i x_t$.

For non-gated case, the term $\frac{\partial h_t}{\partial x_t}$ is numerically equal to $\mathbf{W}_x$. Therefore, as long as $\mathbf{W}_x$ is a non-zero matrix, R_t in Equation 3 will also be non-zero (assuming that all the other terms are non-zero). To learn valuable information at timestep $t \in P_{in}$, $\mathbf{W}_x$ cannot be a zero matrix. This is why a non-gated linear RNN does not have a way to reduce $\mathcal{C}_{noise\to out}$.

For gated case, the gating function outputs a near-zero value to filter out information from the noisy phase P_{noise}. Both red terms in Equation 6 will be small, making R_t in Equation 3 sufficiently small for $t \in P_{noise}$, as evidenced by the abrupt shifts of saliency across different phases in Figure 1 (Right, ③). Consequently, $\mathcal{C}_{noise\to out}$ will be decreased, and $\text{SNR}_{\mathcal{C}}$ will be higher. Ideally, we can have the following proposition.

Proposition 1. *If c_t is input-independent, and $\exists \mathbf{W}_i \in \mathbb{R}^{N_x \times N_h}$ so that $\sigma(\mathbf{W}_i x_t) = 0, t \in P_{noise}$ in a certain environment, $\text{SNR}_{\mathcal{C}}$ can reach its maximum value* 1.

A detailed proof of Proposition 1 can be found in Appendix A.1.

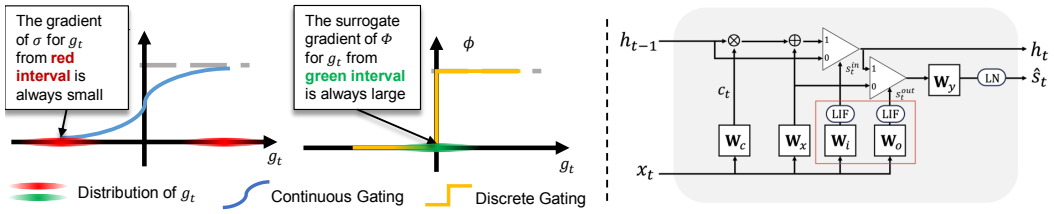

Figure 2: **Left:** Comparison of continuous Gating and discrete Gating. **Right:** Visualization of the structure of our proposed recurrent cell.

4 METHOD

4.1 SPIKING NEURON AS DISCRETE GATING

Continuous Gating vs Discrete Gating. Despite that sigmoid-based continuous gating could increase $\text{SNR}_{\mathcal{C}}$ by decreasing $\mathcal{C}_{noise\to out}$, due to the way of parameter initialization and the softness of the sigmoid function, saturated values, *i.e.*, 0 and 1, cannot be taken at the beginning of training. After sufficient training, the value of the gates may saturate, but this also causes the gradient of the gating function to vanish. This is because sigmoid gating requires inputs to be extremely large to reach saturated outputs, while the derivative of the sigmoid function $\sigma'(x) = \sigma(x)(1-\sigma(x))$ at those input values is near-zero, as shown in Figure 2 (Left). This is a dilemma that has been discussed in Gu et al. (2020). Thus, sigmoid gating can cause vanishing gradients, which in turn diminishes training efficiency. Furthermore, its inability to effectively filter noise early in training also reduces the sample efficiency of memory-based RL agents.

In contrast, a discrete gating function ϕ with surrogate gradient can **reach saturated outputs without requiring inputs to be extremely large**, thus preventing the problem of gradient vanishing. As shown in Figure 2 (Left), with Heaviside step function $\Theta(\cdot)$ as the discrete gating $\mathbf{s}_t = \Theta(f(x, \mathbf{W}_i))$, $\mathbf{W}_i$ can satisfy the conditions of Proposition 1 without having extremely large values, which in turn allows $\mathcal{C}_{noise\to out}$ to be minimized and $\text{SNR}_{\mathcal{C}}$ to be maximized at the early stage of training.

Parallel Spiking Neuron. In POMDP tasks where inputs are partially observable, the gating function should be history-dependent so that it can control the selection mechanism based on more information. Therefore, we could use spiking neurons with temporal dynamics as the gating function. To maintain the benefit of linear RNN, which is parallel training, we use Parallel Spiking Neuron (PSN) Fang et al. (2023) as our discrete gating function, which is

$$\begin{aligned} \mathbf{m}_t &= (1-\frac{1}{\tau_m})\mathbf{m}_{t-1} + \frac{1}{\tau_m}\mathbf{u}_t, \quad && \mathbf{m}_t, \mathbf{u}_t \in \mathbb{R}^{N_h}, \\ \mathbf{s}_t &= \Theta(\mathbf{m}_t - V_{th}), && \mathbf{s} \in \{0,1\}^{N_h}, \end{aligned} \tag{7}$$

where V_{th} is the threshold, $\mathbf{u}_t = \mathbf{W}_i x_t$ is the input current and $\mathbf{m}_t$ is the membrane potential. We choose the derivative of the $\arctan$ function as the surrogate gradient for $\Theta(\cdot)$ during training. Following Fang et al. (2021a), we have $\Theta'(x) \triangleq \frac{1}{1+(\pi x)^2}$. To improve the expressivity of neurons, we define $\frac{1}{\tau_m} \in (0,1)^{N_h}$ as a learnable vector, where τ_m is the membrane time constant of the spiking neuron.

Adding Stochasticity. PSN already meets the desired property of being a gating function. However, due to the absence of stochasticity in its neuronal dynamics, if a neuron outputs 0 at some point during training, it is likely to remain 0 for the rest of the training process. Therefore, we refer to the trick in discrete model compression Gao et al. (2020), which uses a stochastic threshold to determine the state of discrete gating. We add stochasticity to V_{th}, which is now summed by a constant and a stochastic variable during training, and is fixed to a constant during inference based on the expectation of the stochastic variable. Specifically, we have

$$V_{th,i} = \begin{cases} \theta + X_i, & \text{parallel training}, \\ \theta + \mathbb{E}X_i, & \text{sequential inference}, \end{cases} \tag{8}$$

where $V_{th,i}$ is the threshold voltage of the i-th neuron, θ is the threshold potential of the neuron, and $X_i \sim U(0,1)$ is the random variable that has a uniform distribution.

4.2 Linear Recurrent Cell with Discrete Gating

In this section, we introduce the structure of our linear recurrent cell with discrete gating. As shown in Figure 2 (Right), c_t is constructed in an input-dependent manner. Components in the red box indicate spiking neurons defined in Section 4.1, which are used as discrete gating. The triangle symbol represents the highway connection controlled by the gating function. LN represents the Layer Normalization function. The key difference between our method and existing sequence models is that we use discrete gating implemented with spiking neurons.

We apply highway connections Srivastava et al. (2015) in both temporal and spatial dimensions, which can control two streams with a single gating signal:

$$\begin{aligned} h_{t,i} &= \begin{cases} c_{t,i} \odot h_{t-1,i} + (\mathbf{W}_x x_t)_i, & s^{in}_{t,i} = 1, \\ h_{t-1,i}, & s^{in}_{t,i} = 0, \end{cases} \quad h_t \in \mathbb{C}^{N_h}, \\ \hat{y}_{t,i} &= \begin{cases} h_{t,i}, & s^{out}_{t,i} = 1, \\ (\mathbf{W}_x x_t)_i, & s^{out}_{t,i} = 0, \end{cases} \quad \hat{y}_t \in \mathbb{C}^{N_h}. \end{aligned} \tag{9}$$

Here, i refers to the i-th element of the cell. s^{in}_t and s^{out}_t refer to the output of the spiking neuron in Section 4.1, their u_t in Equation 7 is calculated as $\mathbf{W}_i x_t$ and $\mathbf{W}_o x_t$, respectively. We follow the setting of existing linear RNNs Orvieto et al. (2023); Morad et al. (2023b) and make our recurrent cell have complex hidden values.

Our method allows multiple ways to construct c_t. An approach is to construct an input-independent c_t in the style of LRU Orvieto et al. (2023). We also propose a novel way to construct input-dependent c_t, which can be written as

$$\hat{c}_t = \mathbf{W}_h x_t, \quad \hat{c}_t \in \mathbb{C}^{N_h}, \quad c_t = \hat{c}_t \frac{\tanh(|\hat{c}_t|)}{|\hat{c}_t|}, \quad c_t \in \mathbb{C}^{N_h}, \tag{10}$$

where $|\cdot|$ is the element-wise absolute value of a complex input. In practice, to avoid the divided-by-zero error, it is implemented as $\sqrt{|\cdot|^2 + 1}$. $\tanh(\cdot)$ is used to clip the elements of c_t inside the unit disk to mitigate value explosion.

Finally, the output $\hat{s}_t$ of our recurrent cell is

$$\hat{s}_t = \text{LN}(\mathbf{W}_y \cdot \text{Concat}(\text{Re}[\hat{y}_t], \text{Im}[\hat{y}_t])), \quad \hat{s}_t \in \mathbb{R}^{N_{out}}, \tag{11}$$

where N_{out} is the output size of the sequence model, $\text{Re}[\cdot]$ and $\text{Im}[\cdot]$ are the real and imaginary components of a complex value. LN denotes the nonparametric layer normalization.

It can be proven that the update process in Equation 9 is still parallelizable. The basic idea is to preprocess the input of the associative scan operation, replacing $c_{t,i}$ with 1 and $(\mathbf{W}_x x_t)_i$ with 0 when $s^{in}_{t,i}$ is 0. A detailed proof of associativity can be found in Appendix A.2.

In addition, with the highway connection defined in Equation 9, even if c_t is input-dependent, the conclusion in Proposition 1 still holds. That is, when s_t is a 0 vector, no gradient can propagate from c_t or $(\mathbf{W}_x x_t)_i$ back to o_t, allowing R_t to be zero. A detailed discussion is provided in Appendix A.3.

5 Experiments

We conducted our experiments on **Passive Visual Match** Hung et al. (2019) and **POPGym** Morad et al. (2023a). Table 1 shows the sequence models used in our experiments. All settings of network structure, RL algorithms, and hyperparameters used in the experiments are provided in Appendix D.

5.1 Verifying the Proposed Saliency-Based Framework

We conducted experiments on Passive Visual Match with a memory length of 250. Training was performed for a total of 1000 episodes, with temporal saliency R^c_t calculated every 50 episodes. According to Equation 2 in Section 3, we computed both $\mathcal{C}_{in \to out}$ and $\mathcal{C}_{noise \to out}$. We performed such experiments on methods including LSTM Hochreiter & Schmidhuber (1997), LRU Orvieto et al. (2023), FFM Morad et al. (2023b), and ours. As shown in Figure 3 (Left), our method achieves the

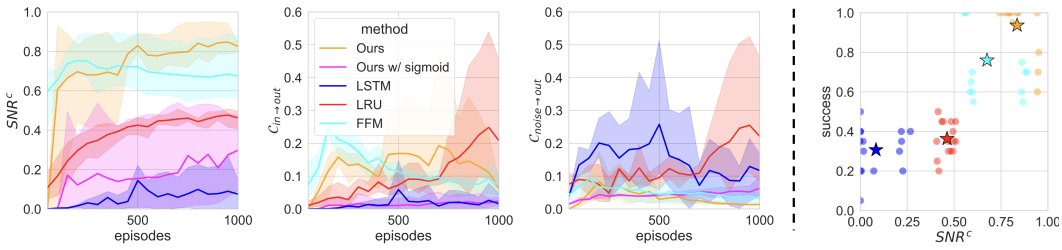

Figure 3: **Left:** Results of $\text{SNR}_{\mathcal{C}}$, $\mathcal{C}_{in\to out}$, and $\mathcal{C}_{noise\to out}$. **Right:** Correlation between $\text{SNR}_{\mathcal{C}}$ and success rate.

	L	G	O
LSTM Hochreiter & Schmidhuber (1997)		✓	
GRU Chung et al. (2014)		✓	
LRU Orvieto et al. (2023)	✓		
FFM Morad et al. (2023b)	✓	✓	
SHM Le et al. (2024)			✓
LiT Katharopoulos et al. (2020)			✓

Table 1: Sequence models used in experiments. "L" indicates linear recurrence. "G" indicates gating mechanism. "O" indicates Non-RNNs sequence models.

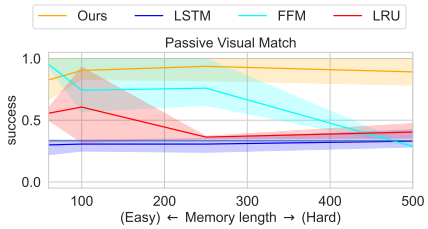

Figure 4: Results of Passive Visual Match with different memory lengths.

highest $\text{SNR}_{\mathcal{C}}$ after around 500 episodes, while other methods have lower $\text{SNR}_{\mathcal{C}}$, indicating their inability to filter out irrelevant information efficiently.

Non-linear RNNs vs Linear RNNs. The second and the third plot in Figure 3 (Left) show the values of $\mathcal{C}_{in\to out}$ and $\mathcal{C}_{noise\to out}$ during training. It can be observed that both values are very low for LSTM due to the vanishing gradient problem of non-linear RNNs, whereas both values are high for LRU, as it alleviates gradient vanishing through linear recurrence. However, since both values remain high, the $\text{SNR}_{\mathcal{C}}$ of LRU is still low.

Non-Gated RNNs vs Gated RNNs. We will still focus on the second and the third plot in Figure 3 (Left). For FFM, $\mathcal{C}_{in\to out}$ remains at a similar level compared to LRU, because both of them are linear RNNs. What increases FFM's $\text{SNR}_{\mathcal{C}}$ is that $\mathcal{C}_{noise\to out}$ of FFM is much lower than that of LRU, indicating that the gating mechanism helps filter out irrelevant information.

Continuous Gating vs Discrete Gating. Nevertheless, because the gating of FFM is continuous, the $\mathcal{C}_{noise\to out}$ value cannot reach its optimal level. In contrast, the third plot in Figure 3 (Left) indicates that our method reduces $\mathcal{C}_{noise\to out}$ to a very low level compared to FFM, thereby effectively increasing the value of $\text{SNR}_{\mathcal{C}}$.

Ablation Study. To further illustrate the importance of discrete gating, we perform an ablation study by replacing our spiking neuron by a sigmoid function while retaining the same network architecture. This method is referred to as **Ours w/ sigmoid**. Results in Figure 3 (Left) show that the sigmoid gate yields a higher $\mathcal{C}_{noise\to out}$, indicating that sigmoid gating cannot effectively filter out noise.

Correlation between $\text{SNR}_{\mathcal{C}}$ and model performance. Figure 3 (Right) demonstrates the correlation between $\text{SNR}_{\mathcal{C}}$ and method performance. The $\text{SNR}_{\mathcal{C}}$-success pair is sampled from the last 5 evaluations of the 3 runs for each method. It can be observed that, given the same number of training steps, a clear positive correlation exists between $\text{SNR}_{\mathcal{C}}$ and the success rate. Since our method achieves the highest $\text{SNR}_{\mathcal{C}}$, it also achieves the highest success rate under the same training steps, fully demonstrating that using discrete gating to filter out irrelevant information can improve sample efficiency in long-term memory tasks.

5.2 Comparison Under Different Memory Settings

Single Long-term Memory Task. In this section, we explore the impact of memory lengths on model performance on Passive Visual Match. In this environment, the first 15 timesteps belong to P_{in} and the last 15 timesteps belong to P_{out}, and the size of P_{noise}, *i.e.*, memory length, can be set to different values. Figure 4 shows a comparison between our method and other methods under various memory lengths of Passive Visual Match. Specifically, we train each model on tasks with memory lengths of $\{60, 100, 250, 500\}$. It can be observed that when the memory length is short, almost all

Table 2: Results of our proposed method compared to existing methods on RepeatPrevious. Methods with $*$ are reported from Le et al. (2024).

Environment	Level	LiT$*$	GRU$*$	FFM$*$	SHM$*$	Ours
RepeatPrevious	Easy	6.0±4.0	**99.9±0.0**	98.4±0.3	88.9±11.1	96.2±1.0
	Medium	-46.8±1.1	-34.7±1.7	-24.3±0.4	48.2±7.2	**96.6±0.5**
	Hard	-48.5±0.3	-41.7±1.8	-33.9±1.0	-19.4±9.9	**88.3±3.5**

Table 3: Results of our proposed method compared to existing methods on general memory tasks in POPGym Benchmarks.Methods with $*$ are reported from Le et al. (2024).

Environment	Level	LiT$*$	GRU$*$	FFM$*$	SHM$*$	Ours
Autoencode	Easy	-44.7±1.4	-37.9±7.7	-32.7±0.6	**49.5±23.3**	38.7±12.5
	Medium	-47.8±0.2	-43.6±3.5	-32.7±0.6	**-28.8±14.4**	-32.9±1.1
	Hard	-48.1±0.1	-48.1±0.7	-47.7±0.5	**-43.9±0.9**	**-43.9±2.5**
Battleship	Easy	-41.3±0.5	-41.1±1.0	-34.0±7.1	**-12.3±2.4**	-35.3±0.5
	Medium	-39.2±0.3	-39.4±0.5	-37.1±3.1	**-16.8±0.6**	-36.9±1.0
	Hard	-38.4±0.2	-38.5±0.5	-38.8±0.3	**-21.2±2.3**	-38.6±0.3
Concentration	Easy	-18.5±0.2	-10.9±1.0	**10.7±1.2**	-1.9±2.4	3.4±0.9
	Medium	**-18.6±0.2**	-21.4±0.5	-24.7±0.1	-21.0±0.8	-21.3±1.1
	Hard	**-83.0±0.1**	-84.0±0.3	-87.5±0.5	-83.3±0.1	-84.5±0.4

methods except LSTM exhibit high sample efficiency. However, as the memory length increases, the performance of all methods except ours declines significantly. The decline happens first on LRU at a memory length of 250, followed by FFM at a memory length of 500. These results indicate that our method is able to maintain higher sample efficiency than other methods for long-term memory tasks.

Interleaving Long-term Memory Tasks. We further conducted experiments to test our proposed method on tasks that have interleaving phases. Following prior works Morad et al. (2023a;b); Le et al. (2024); Morad et al. (2024), we conduct experiments on POPGym Morad et al. (2023a). We refer to the experimental setting of Le et al. (2024) and choose the task "Repeat Previous", where the goal at the i-th time step is to output the input from the (i-k)-th step. This can be seen as multiple sub-tasks that are stacked together. The memory length of each sub-task is k-1. The task has three levels: easy, medium, and hard, with the distinction being the value of k. A larger k corresponds to a longer memory length and a higher task difficulty. As shown in Table 2, our method achieves strong performance across all three difficulty levels and is the only method that scores above 90 on both the medium and hard tasks. This experimental conclusion aligns with our findings on the Passive Visual Match task, further demonstrating the effectiveness of our method in long-term memory tasks.

General Memory Tasks. To verify the general ability of our method in other discrete tasks that require memory. We conduct experiments on tasks in POPGym following the setting of Le et al. (2024). In these tasks, P_{in} is significantly larger than P_{noise} (Autoencode), or the memory lengths are not specified (Battleship, Concentration). As shown in Table 3, our method performs comparably to SOTA methods. Although SHM achieves better results on Battleship, its memory size is 4x larger than ours. To sum up, our method is effective across a wide range of long-term memory tasks.

Short-term Memory Tasks. We also verified whether our discrete gating method can perform well on short-term memory tasks. Following Ni et al. (2023); Lu et al. (2024), we conduct experiments in a standard POMDP task used in prior works Ni et al. (2021; 2023); Lu et al. (2024). The hyperparameter settings and other training details are listed in Appendix D.2. The results are provided in Appendix C.2. The results indicate that while our method is not designed for this type of task, it can still have comparable performance against SOTA methods.

6 Discussion

Long-term memory is an important challenge for partially observable reinforcement learning. In this paper, we analyze why existing sequence models in memory-based RL fail to train efficiently on such tasks and propose spiking neurons as a discrete gating mechanism to solve this problem. Our experimental results underscore the importance of incorporating discrete gating into modern RNN architectures. However, training models with discrete functions is challenging due to the use of surrogate gradient. More limitations are discussed in Appendix E.

REFERENCES

Karl Johan Åström. Optimal control of markov processes with incomplete state information i. *Journal of mathematical analysis and applications*, 10:174–205, 1965.

Víctor Campos, Brendan Jou, Xavier Giró-i Nieto, Jordi Torres, and Shih-Fu Chang. Skip rnn: Learning to skip state updates in recurrent neural networks. *arXiv preprint arXiv:1708.06834*, 2017.

Sneha Chaudhari, Varun Mithal, Gungor Polatkan, and Rohan Ramanath. An attentive survey of attention models. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 12(5):1–32, 2021.

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

Xinyi Chen, Jibin Wu, Chenxiang Ma, Yinsong Yan, Yujie Wu, and Kay Chen Tan. Pmsn: A parallel multi-compartment spiking neuron for multi-scale temporal processing. *arXiv preprint arXiv:2408.14917*, 2024.

Petros Christodoulou. Soft actor-critic for discrete action settings. *arXiv preprint arXiv:1910.07207*, 2019.

Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

Gabriel Dulac-Arnold, Daniel Mankowitz, and Todd Hester. Challenges of real-world reinforcement learning. *arXiv preprint arXiv:1904.12901*, 2019.

Esraa Elelimy, Adam White, Michael Bowling, and Martha White. Real-time recurrent learning using trace units in reinforcement learning. *arXiv preprint arXiv:2409.01449*, 2024.

Jeffrey L Elman. Finding structure in time. *Cognitive science*, 14(2):179–211, 1990.

Wei Fang, Zhaofei Yu, Yanqi Chen, Tiejun Huang, Timothée Masquelier, and Yonghong Tian. Deep residual learning in spiking neural networks. *Advances in Neural Information Processing Systems*, 34:21056–21069, 2021a.

Wei Fang, Zhaofei Yu, Yanqi Chen, Timothée Masquelier, Tiejun Huang, and Yonghong Tian. Incorporating learnable membrane time constant to enhance learning of spiking neural networks. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 2661–2671, 2021b.

Wei Fang, Zhaofei Yu, Zhaokun Zhou, Ding Chen, Yanqi Chen, Zhengyu Ma, Timothée Masquelier, and Yonghong Tian. Parallel spiking neurons with high efficiency and ability to learn long-term dependencies. *Advances in Neural Information Processing Systems*, 36:53674–53687, 2023.

Leo Feng, Frederick Tung, Mohamed Osama Ahmed, Yoshua Bengio, and Hossein Hajimirsadeghi. Were rnns all we needed? *arXiv preprint arXiv:2410.01201*, 2024.

Scott Fujimoto, Herke Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *International conference on machine learning*, pp. 1587–1596. PMLR, 2018.

Shangqian Gao, Feihu Huang, Jian Pei, and Heng Huang. Discrete model compression with resource constraint for deep neural networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 1899–1908, 2020.

Thomas Gisiger and Mounir Boukadoum. Mechanisms gating the flow of information in the cortex: what they might look like and what their uses may be. *Frontiers in computational neuroscience*, 5: 1, 2011.

Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. *arXiv preprint arXiv:2312.00752*, 2023.

Albert Gu, Caglar Gulcehre, Thomas Paine, Matt Hoffman, and Razvan Pascanu. Improving the gating mechanism of recurrent neural networks. In *International conference on machine learning*, pp. 3800–3809. PMLR, 2020.

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.

Matthew J Hausknecht and Peter Stone. Deep recurrent q-learning for partially observable mdps. In *AAAI fall symposia*, volume 45, pp. 141, 2015.

Sepp Hochreiter. The vanishing gradient problem during learning recurrent neural nets and problem solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 6(02): 107–116, 1998.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8): 1735–1780, 1997.

Chia-Chun Hung, Timothy Lillicrap, Josh Abramson, Yan Wu, Mehdi Mirza, Federico Carnevale, Arun Ahuja, and Greg Wayne. Optimizing agent behavior over long time scales by transporting value. *Nature communications*, 10(1):5223, 2019.

Aya Abdelsalam Ismail, Mohamed Gunady, Luiz Pessoa, Hector Corrada Bravo, and Soheil Feizi. Input-cell attention reduces vanishing saliency of recurrent neural networks. *Advances in Neural Information Processing Systems*, 32, 2019.

Leslie Pack Kaelbling, Michael L Littman, and Anthony R Cassandra. Planning and acting in partially observable stochastic domains. *Artificial intelligence*, 101(1-2):99–134, 1998.

Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. Transformers are rnns: Fast autoregressive transformers with linear attention. In *International conference on machine learning*, pp. 5156–5165. PMLR, 2020.

Dharshan Kumaran, Demis Hassabis, and James L McClelland. What learning systems do intelligent agents need? complementary learning systems theory updated. *Trends in cognitive sciences*, 20(7): 512–534, 2016.

Hung Le, Kien Do, Dung Nguyen, Sunil Gupta, and Svetha Venkatesh. Stable hadamard memory: Revitalizing memory-augmented agents for reinforcement learning. *arXiv preprint arXiv:2410.10132*, 2024.

Tao Lei, Yu Zhang, Sida I Wang, Hui Dai, and Yoav Artzi. Simple recurrent units for highly parallelizable recurrence. *arXiv preprint arXiv:1709.02755*, 2017.

Yang Li, Yinqian Sun, Xiang He, Yiting Dong, Dongcheng Zhao, and Yi Zeng. Parallel spiking unit for efficient training of spiking neural networks. In *2024 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8. IEEE, 2024.

Zhanfeng Liao, Yan Liu, Qian Zheng, and Gang Pan. Spiking nerf: Representing the real-world geometry by a discontinuous representation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 13790–13798, 2024.

Chenhao Lu, Ruizhe Shi, Yuyao Liu, Kaizhe Hu, Simon S Du, and Huazhe Xu. Rethinking transformers in solving pomdps. *arXiv preprint arXiv:2405.17358*, 2024.

Chris Lu, Yannick Schroecker, Albert Gu, Emilio Parisotto, Jakob Foerster, Satinder Singh, and Feryal Behbahani. Structured state space models for in-context reinforcement learning. *Advances in Neural Information Processing Systems*, 36:47016–47031, 2023.

Fiona McNab and Torkel Klingberg. Prefrontal cortex and basal ganglia control access to working memory. *Nature neuroscience*, 11(1):103–107, 2008.

Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. *Communications of the ACM*, 65(1):99–106, 2021.

Steven Morad, Ryan Kortvelesy, Matteo Bettini, Stephan Liwicki, and Amanda Prorok. Popgym: Benchmarking partially observable reinforcement learning. *arXiv preprint arXiv:2303.01859*, 2023a.

Steven Morad, Ryan Kortvelesy, Stephan Liwicki, and Amanda Prorok. Reinforcement learning with fast and forgetful memory. *Advances in Neural Information Processing Systems*, 36:72008–72029, 2023b.

Steven Morad, Chris Lu, Ryan Kortvelesy, Stephan Liwicki, Jakob Foerster, and Amanda Prorok. Revisiting recurrent reinforcement learning with memory monoids. *arXiv e-prints*, pp. arXiv–2402, 2024.

Daniel Neil, Michael Pfeiffer, and Shih-Chii Liu. Phased lstm: Accelerating recurrent network training for long or event-based sequences. *Advances in neural information processing systems*, 29, 2016.

Tianwei Ni, Benjamin Eysenbach, and Ruslan Salakhutdinov. Recurrent model-free rl can be a strong baseline for many pomdps. *arXiv preprint arXiv:2110.05038*, 2021.

Tianwei Ni, Michel Ma, Benjamin Eysenbach, and Pierre-Luc Bacon. When do transformers shine in rl? decoupling memory from credit assignment. *Advances in Neural Information Processing Systems*, 36:50429–50452, 2023.

Antonio Orvieto, Samuel L Smith, Albert Gu, Anushan Fernando, Caglar Gulcehre, Razvan Pascanu, and Soham De. Resurrecting recurrent neural networks for long sequences. In *International Conference on Machine Learning*, pp. 26670–26698. PMLR, 2023.

Emilio Parisotto, Francis Song, Jack Rae, Razvan Pascanu, Caglar Gulcehre, Siddhant Jayakumar, Max Jaderberg, Raphael Lopez Kaufman, Aidan Clark, Seb Noury, et al. Stabilizing transformers for reinforcement learning. In *International conference on machine learning*, pp. 7487–7498. PMLR, 2020.

Haotong Qin, Ruihao Gong, Xianglong Liu, Xiao Bai, Jingkuan Song, and Nicu Sebe. Binary neural networks: A survey. *Pattern Recognition*, 105:107281, 2020.

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.

David Raposo, Sam Ritter, Adam Santoro, Greg Wayne, Theophane Weber, Matt Botvinick, Hado van Hasselt, and Francis Song. Synthetic returns for long-term credit assignment. *arXiv preprint arXiv:2102.12425*, 2021.

Nitin Rathi and Kaushik Roy. Diet-snn: Direct input encoding with leakage and threshold optimization in deep spiking neural networks. *arXiv preprint arXiv:2008.03658*, 2020.

Kaushik Roy, Akhilesh Jaiswal, and Priyadarshini Panda. Towards spike-based machine intelligence with neuromorphic computing. *Nature*, 575(7784):607–617, 2019.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Jimmy TH Smith, Andrew Warrington, and Scott W Linderman. Simplified state space layers for sequence modeling. *arXiv preprint arXiv:2208.04933*, 2022.

Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. Highway networks. *arXiv preprint arXiv:1505.00387*, 2015.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Zekang Wang, Zhe He, Edan Toledo, and Steven Morad. Popgym arcade: Parallel pixelated pomdps. *arXiv preprint arXiv:2503.01450*, 2025.

Xingting Yao, Fanrong Li, Zitao Mo, and Jian Cheng. Glif: A unified gated leaky integrate-and-fire neuron for spiking neural networks. *Advances in Neural Information Processing Systems*, 35: 32160–32171, 2022.

Duzhen Zhang, Qingyu Wang, Tielin Zhang, and Bo Xu. Biologically-plausible topology improved spiking actor network for efficient deep reinforcement learning. *arXiv preprint arXiv:2403.20163*, 2024.

Xuanle Zhao, Duzhen Zhang, Han Liyuan, Tielin Zhang, and Bo Xu. Ode-based recurrent model-free reinforcement learning for pomdps. *Advances in Neural Information Processing Systems*, 36: 65801–65817, 2023.

A Theoretical Results

A.1 Proof of Proposition 1

Proposition 2. *If c_t is input-independent, and $\exists \mathbf{W}_i \in \mathbb{R}^{N_x \times N_h}$ so that $\sigma(\mathbf{W}_i x_t) = 0, t \in P_{noise}$ in a certain environment, $SNR_{\mathcal{C}}$ can reach its maximum value* 1.

Proof. The expression of $\text{SNR}_{\mathcal{C}}$ is

$$\text{SNR}_{\mathcal{C}} = \frac{\mathcal{C}_{in\to out}}{\mathcal{C}_{in\to out} + \mathcal{C}_{noise\to out}}, \mathcal{C}_{in\to out} \geq 0, \mathcal{C}_{noise\to out} \geq 0, \tag{12}$$

According to Equation 2 and Equation 3, $\mathcal{C}_{noise\to out}$ is

$$\mathcal{C}_{noise\to out} = \frac{1}{|P_{noise}|} \sum_{t\in P_{noise}} R_t^c = \frac{1}{|P_{noise}|} \sum_{t\in P_{noise}} \Big| \sum_{i\in P_{out}} \Big[\frac{\partial Q(\hat{s}_i, a_i)}{\partial \hat{s}_i} G_t \frac{\partial z_t}{\partial o_t} \Big] \Big| \tag{13}$$

For a memory-based agent with gated linear recurrence defined in Equation 5, which is

$$h_t = c_t \odot h_{t-1} + (\mathbf{W}_x x_t) \odot \sigma(\mathbf{W}_i x_t), \tag{14}$$

the term G_t in Equation 13 is

$$\begin{aligned} G_t =& \frac{\partial \hat{s}_i}{\partial h_i} \Big(\prod_{j=t+1}^{i} \frac{\partial h_j}{\partial h_{j-1}} \Big) \frac{\partial h_t}{\partial z_t} \\ =& \frac{\partial \hat{s}_i}{\partial h_i} \Big(\prod_{j=t+1}^{i} \frac{\partial h_j}{\partial h_{j-1}} \Big) \Big[\text{diag}\big(\sigma(g_t) \odot \frac{\partial h_t}{\partial u_t}\big) \frac{\partial u_t}{\partial z_t} + \\ & \text{diag}\big(u_t \odot \frac{\partial h_t}{\partial \sigma(g_t)} \odot \sigma'(g_t)\big) \frac{\partial g_t}{\partial z_t} \Big], \end{aligned} \tag{15}$$

where $u_t = \mathbf{W}_x z_t$ and $g_t = \mathbf{W}_i z_t$. $\text{diag}(\mathbf{x})$ denotes the diagonal matrix formed by vector $\mathbf{x}$. Note that $\sigma'(x) = \sigma(x)(1 - \sigma(x))$. Therefore, when $\sigma(\mathbf{W}_i z_t) = 0, t \in P_{noise}$, the red terms $\sigma(g_t)$ and $\sigma'(g_t)$ in Equation 15 are 0 vectors. We have

$$R_t^c = 0, t \in P_{noise} \tag{16}$$

$$\mathcal{C}_{noise\to out} = \frac{1}{|P_{noise}|} \sum_{t\in P_{noise}} R_t^c = 0 \tag{17}$$

$$\text{SNR}_{\mathcal{C}} = \frac{\mathcal{C}_{in\to out}}{\mathcal{C}_{in\to out}} = 1. \tag{18}$$

□

A.2 Proof of Associativity of Equation 9

We focus on the first equation in Equation 9, which is:

$$h_{t,i} = \begin{cases} c_{t,i} \odot h_{t-1,i} + (\mathbf{W}_x x_t)_i, & s^{in}_{t,i} = 1, \\ h_{t-1,i}, & s^{in}_{t,i} = 0. \end{cases} \tag{19}$$

It can be rewritten as

$$\begin{aligned} h_t &= (c_t \odot h_{t-1} + \mathbf{W}_x x_t) \odot s^{in}_t + h_{t-1} \odot (1 - s^{in}_t), \\ &= c_t \odot h_{t-1} \odot s^{in}_t + h_{t-1} \odot (1 - s^{in}_t) + \mathbf{W}_x x_t \odot s^{in}_t, \\ &= [c_t \odot s^{in}_t + (1 - s^{in}_t)] \odot h_{t-1} + \mathbf{W}_x x_t \odot s^{in}_t, \end{aligned} \tag{20}$$

where $\odot$ is the element-wise multiplication.

Let $\bullet$ be the binary operator that operates on element e_k, which is defined as:

$$e_k = (e_{k,a}, e_{k,b}) := \left(c_t \odot s^{in}_t + (1 - s^{in}_t), \mathbf{W}_x x_t \odot s^{in}_t\right). \tag{21}$$

And the operator $\bullet$ is defined as:

$$e_i \bullet e_j = (e_{j,a} \odot e_{i,a}, e_{j,a} \odot e_{i,b} + e_{j,b}). \tag{22}$$

This can be seen as a special case of the scan operator of S5Smith et al. (2022), where the matrices are all diagonal. Note that $\bullet$ is associative, which means for any element x, y, z, we have $(x \bullet y) \bullet z = x \bullet (y \bullet z)$. Thus, the operation can be computed in parallel with a time complexity of $O(\log T)$, where T is the sequence length. The associativity of S5 operator is proven in Smith et al. (2022).

A.3 Discussion of Discrete Gating

Following Equation 20 and the BPTT algorithm, the term G_t in Equation 3, which relates to the structure of the sequence model is

$$\begin{aligned} G_t =& \frac{\partial \hat{s}_i}{\partial h_i} \Big(\prod_{j=t+1}^{i} \frac{\partial h_j}{\partial h_{j-1}} \Big) \frac{\partial h_t}{\partial z_t} \\ =& \frac{\partial \hat{s}_i}{\partial h_i} \Big(\prod_{j=t+1}^{i} \frac{\partial h_j}{\partial h_{j-1}} \Big) \Big[\mathrm{diag}\Big(s^{in}_t \odot \frac{\partial h_t}{\partial u_t} \Big) \frac{\partial u_t}{\partial z_t} + \\ & \mathrm{diag}\Big(h_{t-1} \odot s^{in}_t \odot \frac{\partial h_t}{\partial c_t} \Big) \frac{\partial c_t}{\partial z_t} \Big], \end{aligned} \tag{23}$$

where $c_t = \mathbf{W}_c z_t$ for an input-dependent c_t.

As defined in Ismail et al. (2019), saliency quantifies the contribution of each input to the output, measuring how input perturbations affect model responses. For temporal saliency analysis, we therefore employ the true gradient of the step function (0 almost everywhere, undefined at zero) instead of the surrogate gradient, which is used for training spiking neurons. In this case, the gradient will not propagate through the discrete s^{in}_t, and the term $\frac{\partial s^{in}_t}{\partial z_t}$ is omitted.

In this case, when s^{in}_t is a vector filled with 0, we can still have the conclusion of Equation 18. Thus, Proposition 1 still holds for input-dependent c_t with the proposed network structure.

B Comparison of Related Works

Table 4 highlights the difference between our method and other related works.

C Additional Results

C.1 Learning Curves of Passive Visual Match.

In this section, we show the learning curves of success rate and return of the Passive Visual Match task. For tasks with memory lengths of 60, 100, and 250, the training was performed on a total of

Table 4: Comparison of Related Works. "SL" refers to Supervised Learning and "RL" refers to Reinforcement Learning.

Method	Learning Method	Discrete Selection Mechanism	Element-wise Gating	Input-dependent Discrete Gating
Diet-SNN Rathi & Roy (2020)	SL			
GLIF Yao et al. (2022)	SL		✓	
SkipRNN Campos et al. (2017)	SL	✓		
Phased LSTM Neil et al. (2016)	SL	✓	✓	
GRSN Qin et al. (2025)	RL		✓	
Ours	RL	✓	✓	✓

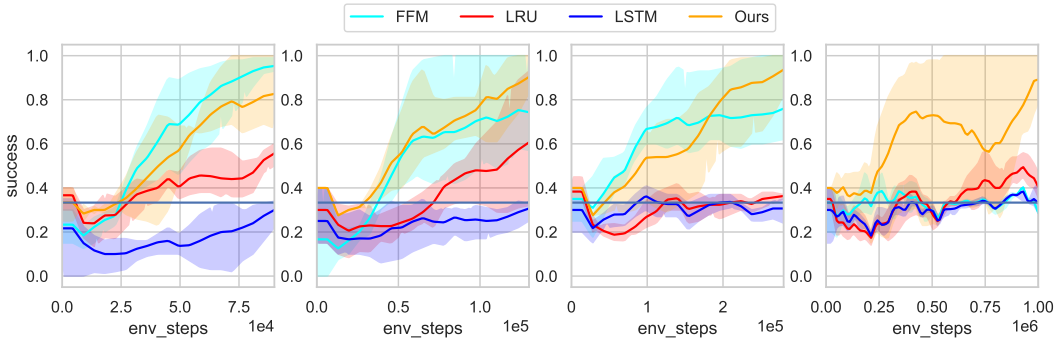

Figure 5: Learning curves of success rate of Passive Visual Match with memory length of 60, 100, 250, and 500. The shaded area indicates 95% confidence interval.

1000 episodes. For the task with a memory length of 500, the training was performed on a total of 2000 episodes. This setting is to make sure the training on each task reaches a stable stage while allowing the evaluation of the final episode to reflect the sample efficiency of each method. The results are shown in Figure 5 and Figure 6. While the success rate has a strong correlation to $\mathrm{SNR}_{\mathcal{C}}$, the return does not seem to have that strong correlation. This is because most of the rewards in this environment come from the apple-picking task in Phase 2. This result reflects the sample efficiency of the agent on the memory-independent task. It can be observed that although our method has higher upper bounds on the confidence intervals than the other methods at times during training, the overall difference between our method and the others is marginal, reflecting that the advantage of our method on memory-irrelevant tasks may not be significant.

C.2 Short-term Memory Tasks

For short-term memory tasks, we choose Pybullet-P for our verification, where the observation space contains only position information. We compare our method with three methods: LSTM Hochreiter & Schmidhuber (1997), LRU Orvieto et al. (2023), and Transformer Vaswani et al. (2017). The

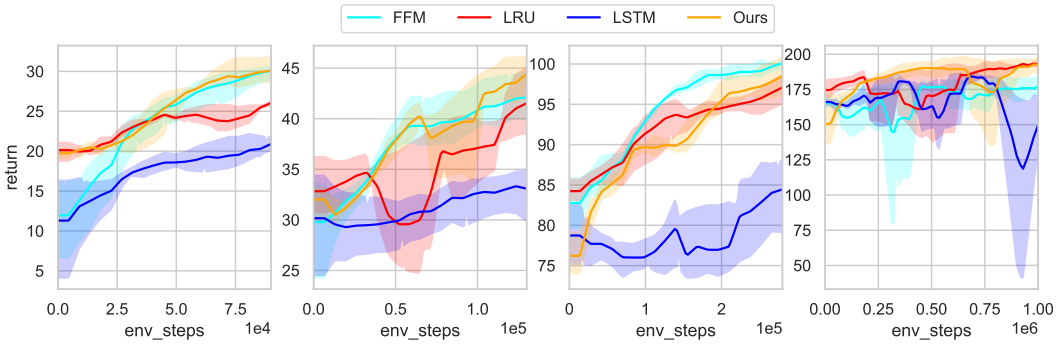

Figure 6: Learning curves of return of Passive Visual Match with memory length of 60, 100, 250, and 500. The shaded area indicates 95% confidence interval.

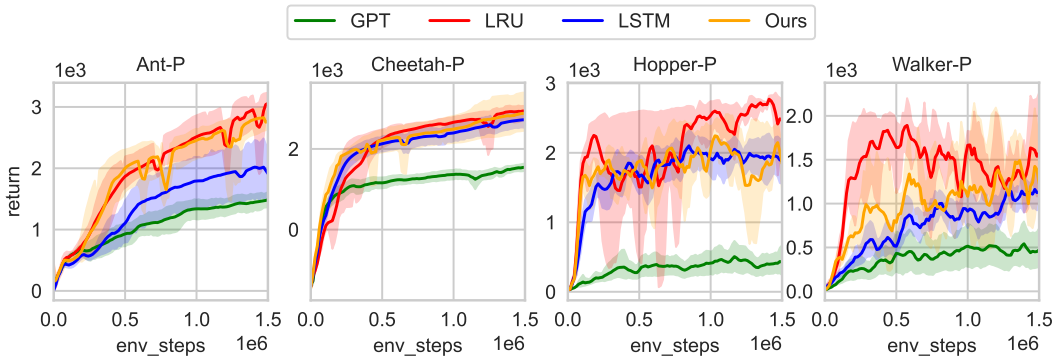

Figure 7: Learning curves of partially observable Pybullet tasks. The shaded area indicates 95% confidence interval.

results of LRU and our method are averaged over 3 runs with different seeds[1], while the results of LSTM and Transformer are reported from Ni et al. (2023). As shown in Figure 7, our method has similar performance to LRU, a linear RNN without gating, in all environments except Ant-P, but its performance does not greatly surpass that of LRU. This is probably because the inductive bias introduced by discrete gating is more suited for long-term memory tasks, instead of continual control tasks that require short-term memory. Nevertheless, our method can still outperform LSTM and Transformer in most environments. This validates the design of our linear recurrent cell in Section 4.2.

D Experimental Details

D.1 Descriptions and Setups of Passive Visual Match

Passive Visual Match Hung et al. (2019) is divided into 3 phases, the first phase lasts for 15 time steps, where the agent gets a color signal. In the second phase, the agent is transported to a room in which the agent needs to perform the task of picking apples. With each apple picked up, the agent can get a reward of 1. In the third phase, the agent needs to use the color information obtained in the first phase to choose their action. When it reaches the correct location, it receives a reward of 10, and the episode is marked as successful. While the environment is a 7*11 grid world, the agent can only observe a 5*5 space around its location.

Note that the seed for creating the training and evaluation environment in the original implementation[2] is not fixed. This will probably result in different color permutations for training and evaluation. As a consequence, the evaluation result does not correctly reflect the true performance of the agent and has a high variance. In our implementation, we manually set the seed for both training and evaluation environments so that the permutation of the color remains the same. We conducted experiments for all methods on this modified implementation.

D.2 Training Details

Hyperparameters. We use the same RL algorithms and hyperparameter configurations as in prior works Ni et al. (2023); Lu et al. (2024); Le et al. (2024). Specifically, we use SACD Christodoulou (2019) for Passive Visual Match, PPO Schulman et al. (2017) for POPGym and TD3 Fujimoto et al. (2018) for Pybullet. The hyperparameters used in TD3 and SACD algorithms are shown in Table 5. The PPO algorithms' hyperparameters are the same as in Le et al. (2024).

In Table 6, we provide the configuration of the sequence models and network architecture in different environments. For the results that are reported from prior works Ni et al. (2023); Le et al. (2024), *e.g.*, LSTM and GPT in Pybullet tasks, we copy their configurations from the original paper Ni et al. (2023).

[1]We use the JAX implementation provided with Ni et al. (2023).

[2]https://github.com/twni2016/Memory-RL

Table 5: Hyperparameters of SACD and TD3.

	Hyperparameter	Value
	Network Hidden Size	(256, 256)
	Batch Size	64
	Learning Rate	3e-4
	Replay Buffer Size	1M
	Smoothing Coefficient	0.05
	Discount Factor	0.99
SACD	Entropy Temperature	0.1
TD3	Exploration Noise	0.1
	Target Noise	0.2
	Target Noise Clip	0.5

Table 6: Hyperparameters of sequence models in different tasks. "o", "a" and "r" in input rows refer to the observation, previous action, and reward, respectively. "-" indicates the model is not used in this experiment, or the value is not in the model configuration.

Environment	Hyperparameter	LSTM	GRU	LRU	FFM	GPT	LiT	SHM	Ours
Passive Visual Match	Inputs	o							
	Embedding Size	100							
	Sequence Length	90, 130, 280, 530							
	Num Layers	1							
	Hidden Size	256	-	200	128	100	-	-	128
	Num Heads	-	-	-	-	2	-	-	-
POPGym	Inputs	o							
	Input Size	128							
	Sequence Length	1024							
	Num Layers	1							
	Recurrent State Size	-	256	-	1024	-	256	16384	4096
Pybullet	Inputs	oa	-	oar	-	oa	-	-	oar
	Sequence Length	64							
	Embedding Size	[32, 16]	-	[64, 16, 16]	-	[64, 64]	-	-	[64, 16, 16]
	Hidden Size	128	-	256	-	128	-	-	128
	Num Layers	1	-	2	-	1	-	-	2
	Num Heads	-	-	-	-	1	-	-	-

In Passive Visual Match, the implementation of LRU is directly extracted from the code of Lu et al. (2024)[3], which forces the hidden size to be the size of the embedding size multiplied by 2. We keep this setting so the hidden size of LRU is 200 in this environment.

The "Recurrent State Size" in Table 6 refers to the size of the recurrent state of an RNN-like sequence model, flattened to a vector and converted to the **float32** data type. SHM Le et al. (2024) is a matrix-based memory module. The matrix size of SHM used in POPGym tasks is 128*128, indicating that the state size is 16384. To calculate the state size of our module, we treated the membrane potential of input and output gates as a part of the hidden state, so the "actual hidden size" (similar to the definition of LSTM and GRU) of our module for POPGym tasks is 1024. Although it is still larger than that of GRU and FFM, the hyperparameter tuning experiment in SHM Le et al. (2024) showed that their performance will not increase much as their state size increases.

One unique hyperparameter has been introduced in our proposed method, namely, θ, which controls the threshold of the spiking neuron. In Passive Visual Match tasks, θ is set to 1, resulting in an expected threshold of 1.5. In other environments, θ is set to 0, resulting in an expected threshold of 0.5. The θ is set higher in Passive Visual Match since we want to make spiking neurons have sparser output, the hidden state will then be changed less frequently, and the model should have better long-term dependency.

[3]https://github.com/CTP314/TFPORL

Compute Resource. We run all our experiments on NVIDIA 3090 and H100 GPUs. For the Passive Visual Match task with a memory length of 250, when running 3 experiments in parallel on a single H100 GPU, these experiments could be completed in approximately 7 hours. For the same task with a memory length of 500, these 3 experiments could be completed in approximately 10 hours. For the POPGym task, when running 2 experiments in parallel on a single 3090 GPU, both experiments would be completed in 7 hours. For the Pybullet task, when running 4 experiments in parallel on a single 3090 GPU, these experiments can be completed in approximately 2 days.

Passive Visual Match and Pybullet require less GPU memory and could theoretically run more experiments in parallel, but a roughly linear increase in total run time is also observed. POPGym, on the other hand, requires more GPU memory, and increasing the number of parallelizations may result in CUDA out-of-memory errors.

E LIMITATIONS

While the experiments presented in the paper confirm the performance of the proposed method, there are also some limitations.

Training Instability. Discrete gating and surrogate gradient could potentially introduce instability during training. It also causes the divergence of the Q value in some runs of continual control tasks. Those invalid runs are discarded in our experimental results. Stabilizing the training process is important to enhance the practical use of discrete gating.

Long-term Temporal Credit Assignment. In early exploratory experiments, we found that our approach did not have similar performance gains over existing methods in long-term temporal credit assignment tasks Ni et al. (2023), such as Key-to-Door Raposo et al. (2021). Future work could attempt to combine the proposed module with algorithms specialized for temporal credit assignment to address this challenge.

F USE OF LLMS

The use of LLM in this paper was limited to correcting grammar, polishing the text, and translating some sections from drafts written by the authors into English.