LongSpike: Fractional-Order State Space Modeling for Efficient Long-Memory Learning

Xinrui He[1], **Qiyu Kang**[1]*, **Xuhao Li**[2], **Zheng-Jun Zha**[1]
[1]University of Science and Technology of China [2]Anhui University

Abstract

Spiking Neural Networks (SNNs) are well-regarded for their biological plausibility and energy efficiency in processing sequential data. However, dominant SNN architectures typically rely on *first-order* Ordinary Differential Equations (ODEs) to govern neuronal state transitions. This first-order formulation induces an exponential-decay memory bias, which can be less suited to representing heavy-tailed temporal dependencies in long-sequence tasks. In this work, we propose **LongSpike**, a novel SNN framework that integrates fractional-order State-Space Modeling (***f*-SSM**) from control theory into the spiking domain. By extending traditional integer-order SSMs to the fractional-calculus regime, LongSpike enables the hierarchical integration of neuronal dynamics with long-memory kernels. To mitigate the computational overhead and parallelization challenges typically associated with fractional operators, we leverage a state-space formulation that supports efficient, parallel training. Empirical evaluations on challenging benchmarks, including Long Range Arena (LRA), large-scale WikiText-103, and Speech Commands, demonstrate that LongSpike outperforms state-of-the-art SNNs in accuracy while preserving sparse synaptic computation. The code is available at `https://github.com/xinruihe389-commits/LongSpike`.

1 Introduction

Sequential data are ubiquitous in applications such as natural language processing, speech recognition, and time-series analysis. A central challenge in modeling such data lies in capturing long-range dependencies, where current observations depend on events occurring far in the past. Traditional recurrent neural networks (RNNs) [1] maintain a hidden state that evolves over time, but they often suffer from vanishing gradients and a lack of parallel computation, which limits their ability to model long-term dependencies [2]. Gated architectures such as LSTMs [3] and GRUs [4] partially alleviate this issue. However, they still rely on strictly sequential computation, making them inefficient for very long sequences.

Spiking Neural Networks (SNNs) represent a class of biologically-grounded architectures [5–8] with significant potential for energy-efficient computing. Despite these advantages, traditional SNNs are also inherently recurrent, as the neural potential state updates strictly follow a sequential order [9]. Transformers address the limitations of RNNs through self-attention, enabling parallel computation [10], but the quadratic complexity of attention restricts scalability to long sequences. Even efficient Transformer variants [11–13] struggle to model fine-grained temporal dynamics over long horizons.

Recently, state-space models (SSMs) have emerged as an effective alternative for long-sequence modeling [14–18]. By formulating sequence modeling as a dynamical system, SSMs achieve near linear-time complexity and strong inductive biases for temporal data. The integration of SSMs and SNNs is natural, and recent examples [19–21] enable long-sequence learning by leveraging

*Correspondence to: Qiyu Kang<qiyukang@ustc.edu.cn>.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

the parallel computation of SSMs and the sparse synaptic computation of SNNs simultaneously. However, existing SSMs are fundamentally based on *integer-order* dynamics, which implicitly assume Markovian state transitions. This assumption may still limit their ability to model long-range memory effects and non-local temporal dependencies commonly observed in real-world sequential data [22, 23].

In control theory, the fractional-order state-space model (f-SSM) framework [24, 25] provides a principled method for modeling systems with memory, as fractional derivatives depend on the history of a signal through a power-law kernel [26, 27]. Motivated by this property, we propose to model the neural potential state update dynamics using f-SSM to capture long-range and non-Markovian temporal dependencies. By extending traditional integer-order SSMs to the fractional-calculus regime, our proposed **LongSpike** enables the hierarchical integration of neuronal dynamics with long-memory kernels. This is related to recent work [28] that incorporates fractional-order dynamics into neuronal membrane potential charging steps. However, that approach still relies on a strictly recurrent diagram where states are updated iteratively. In this work, to mitigate the computational overhead and parallelization challenges typically associated with fractional operators, we leverage a state-space formulation that supports efficient, parallel training.

Main contributions. Our main contributions are summarized as follows:

- We introduce LongSpike, a spiking sequence modeling framework that integrates f-SSM into SNNs, replacing the standard first-order Markovian state transition with a fractional-order dynamics that captures long-memory effects. We demonstrate how hierarchical integration of long-memory kernels provides a more direct inductive bias for heavy-tailed temporal dependencies than standard first-order ODE-based SNNs.

- We address the computational bottleneck of fractional operators by reformulating the dynamics into a parallelizable state-space representation. This formulation allows the model to bypass the constraints of strictly sequential updates, enabling efficient parallel training on long sequences while maintaining the sparse computation inherent in SNNs.

- We conduct extensive experiments on long-sequence benchmarks, including Long Range Arena (LRA), the large-scale WikiText-103 language modeling dataset, and Speech Commands. Our results show that LongSpike consistently outperforms state-of-the-art SNNs in accuracy and long-range modeling capability while preserving high energy efficiency.

2 Related work

2.1 Development of Sequence Models

Early sequence models such as RNNs [1] and gated variants including LSTMs and GRUs [3, 4] laid the foundation for temporal modeling, but they can suffer from vanishing and exploding gradients, which limits their ability to capture long-range dependencies. Transformers [10] mitigate this issue through self-attention, enabling parallel computation and improved long-range modeling. However, the quadratic complexity of attention remains a major bottleneck for very long sequences [29, 30], motivating more efficient alternatives.

SSMs have recently emerged as a strong paradigm for long-sequence modeling. The HiPPO framework [14] introduced principled state representations based on orthogonal polynomials, while S4 and its variants [15, 16, 31] improved efficiency and stability through structured parameterizations. More recent models such as Mamba [17] and Mamba-2 [18] further improve efficiency and selectivity, making SSMs competitive with Transformer-based approaches. Nevertheless, existing SSMs rely on integer-order dynamics and implicitly assume Markovian state transitions, which can limit their ability to represent long-memory effects from the perspective of fractional differential equations (FDEs). Specifically, the impulse response of integer-order systems exhibits fast exponential decay $\propto e^{-t/\tau}$, which is characterized as short-tailed. In contrast, the impulse response of an FDE follows a power-law decay $\propto t^{-\alpha}$, providing a long-tailed distribution that inherently preserves fundamental memory over significantly longer horizons.

2.2 SNNs for Sequential Modeling

SNNs [5–8] are biologically inspired models that communicate via discrete spikes, enabling event-driven computation and high energy efficiency. Advances in surrogate gradient methods [7] and Backpropagation Through Time (BPTT) [32] have made it feasible to train deep SNNs effectively.

Recent studies have explored leveraging SNNs for long-sequence modeling by combining spiking dynamics with structured sequence models. Notable examples include spiking variants of SSMs such as SpikingS4 [20] and SpikingSSMs [19], as well as hybrid architectures that integrate spiking activations into Transformer- or RWKV-style blocks [21, 33]. Other representative approaches include SpikingLMUFormer [34], based on Legendre Memory Units [35], Binary S4D [36], which employs binary spiking activations for improved parallel efficiency, and P-SpikeSSM [37], which enhances sparsity through stochastic spiking. While these approaches benefit from sparse computation and energy efficiency, their temporal modeling capacity remains limited by integer-order dynamics, leaving long-range dependency modeling as an open challenge.

2.3 Fractional Calculus in Neural Networks

Fractional calculus provides a principled framework for modeling systems with memory and non-local temporal dependencies, as fractional derivatives depend on the entire history of a signal through power-law kernels [26, 27]. FDEs have been successfully applied in physics and engineering to model long-memory processes [22, 38]. In machine learning, fractional operators have been introduced into neural architectures to enrich temporal and structural modeling [39–41]. Recent studies incorporate fractional calculus into graph learning to improve accuracy and robustness [40, 42], and into generative modeling through fractional diffusion to encourage more diverse samples [43]. In spiking neural modeling, fractional dynamics have mainly been explored at the neuron level. For example, fractional leaky integrate-and-fire models improve spike-frequency adaptation and robustness [44, 45]. More recent efforts develop fractional-order SNN frameworks [28], but these models remain recurrent and require iterative state updates. In contrast, our approach applies fractional dynamics to the state evolution of a state-space model, rather than directly modifying spiking neuron dynamics, which enables scalable long-range sequence modeling with parallel computation.

3 Preliminaries

In this section, we will briefly review the key concepts that underpin this work, including fractional-order calculus, FDEs, state-space models, and the LIF neuron model, which form the foundation of our advancements.

3.1 Basic Fractional Order Calculus

Fractional-order calculus generalizes the concepts of differentiation and integration to non-integer orders. While integer-order derivatives are local operators, fractional-order derivatives are non-local, making them particularly effective for modeling dynamical systems with hereditary properties and long-range temporal dependencies.

For a differentiable function $y(t)$, the standard first-order derivative is defined by the local rate of change: $\frac{dy(t)}{dt} = \lim_{\Delta t \to 0} \frac{y(t+\Delta t)-y(t)}{\Delta t}$. To capture historical dependencies within the derivative itself, we utilize the Caputo formulation, which is widely preferred in physical and engineering applications because it incorporates standard initial conditions.

Definition 1. *(Caputo Fractional Derivative) [46]. For a function $y(t)$ defined on the interval $[0, T]$, the Caputo fractional derivative of order $\alpha \in (0, 1)$ is defined as:*

$$D^{\alpha} y(t) = \frac{1}{\Gamma(1-\alpha)} \int_0^t \frac{y'(\tau)}{(t-\tau)^{\alpha}} d\tau, \tag{1}$$

where $y'(\tau)$ is the first-order derivative of $y(\tau)$, and $\Gamma(\cdot)$ denotes the Gamma function.

Remark 1. *The Caputo derivative provides a natural generalization from local first-order dynamics to non-local dynamics with long-memory effects. As the order α approaches 1, the integral kernel in* (1) *simplifies, and the operator converges to the classical first-order derivative. For $0 < \alpha < 1$,*

the derivative at time t becomes a weighted accumulation of the entire history of $y'(\tau)$, where the power-law kernel $(t-\tau)^{-\alpha}$ governs the rate of decay for past information.

3.1.1 Fractional Differential Equations

A fractional differential equation replaces the integer-order derivative in a standard ODE with a fractional derivative, which introduces history dependence and can better represent long-memory dynamics. For comparison, a first-order initial value problem is

$$\frac{dy(t)}{dt} = f(t, y(t)), \quad y(0) = y_0. \tag{2}$$

Using the Caputo derivative in Equation (1), the corresponding fractional-order form is

$$D^{\alpha} y(t) = f(t, y(t)), \quad y(0) = y_0. \tag{3}$$

Because closed-form solutions for FDEs are rarely tractable, these systems are typically addressed through numerical methods. Common techniques involve temporal discretization to approximate the non-local fractional operator, such as the fractional Adams–Bashforth–Moulton (ABM) predictor-corrector method [26, 40].

3.2 State Space Models

SSMs provide a robust framework for representing linear dynamical systems. These models describe the evolution of a system's internal state over time as a function of its prior state and an external input. In a continuous-time setting, the dynamics are governed by a first-order differential equation.

Definition 2. *(Continuous-Time State Space Models) [15, 17]. Let $h(t) \in \mathbb{R}^N$ denote the latent state vector and $x(t) \in \mathbb{R}$ denote the input signal at time t. The state evolution and the corresponding output $y(t)$ are defined as:*

$$\frac{dh(t)}{dt} = Ah(t) + Bx(t), \quad y(t) = Ch(t) + Dx(t) \tag{4}$$

where $A \in \mathbb{R}^{N \times N}$ is the state transition matrix, $B \in \mathbb{R}^{N \times 1}$ is the input matrix, $C \in \mathbb{R}^{1 \times N}$ is the output matrix, and $D \in \mathbb{R}$ is the feedthrough term.

In alignment with existing literature on deep SSMs [16, 17], we assume $D = 0$ to simplify notations, as the feedthrough term functions as a standard skip-connection.

To process discrete sequences, the continuous system must be discretized using a sampling step Δ. The resulting Discrete-Time SSM is defined as:

$$h[t] = \bar{A}h[t-1] + \bar{B}x[t], \quad y[t] = Ch[t] \tag{5}$$

where $\bar{A}$ and $\bar{B}$ are discretized matrices derived from A, B, and Δ via rules such as the Bilinear or Zero-Order Hold (ZOH) transformation [15]. Structured parameterizations, such as enforcing $\bar{A}$ to be diagonal or low-rank, are often employed to ensure computational efficiency [31].

For efficient training on long sequences, the linear and time-invariant nature of the discretized SSM allows it to be reformulated as a global convolution:

$$y[k] = \sum_{j=1}^{k} C\bar{A}^{k-j}\bar{B}x[j] \tag{6}$$

By pre-computing the convolution kernel $K = \left(C\bar{B}, C\bar{A}\bar{B}, \ldots, C\bar{A}^{L-1}\bar{B}\right)$, where L is the sequence length, the output can be computed as: $y = x * K$. This representation enables full parallelization across the temporal dimension. By utilizing the Fast Fourier Transform (FFT), the computational complexity is reduced to $O(L \log L)$ [47]. Furthermore, on modern parallel hardware such as GPUs, the kernel K can be computed efficiently through optimized parallel scan or summation algorithms, bypassing the bottlenecks associated with strictly sequential updates.

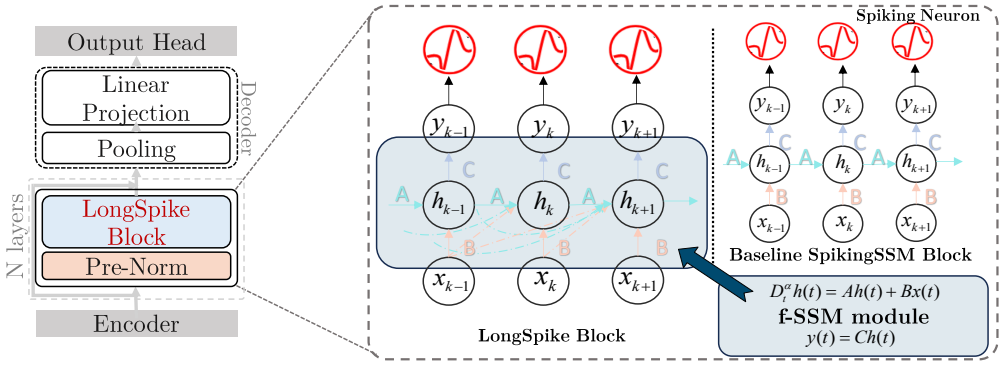


Figure 1: Overview of the LongSpike architecture. Compared with the baseline SpikingSSM [19], LongSpike incorporates fractional-order state-space dynamics into spiking neurons via an f-SSM module, enabling enhanced temporal modeling with efficient parallel computation.

3.3 LIF Neuron and SNN

The LIF model is a standard representation of neuronal dynamics in SNNs [48]. It describes a neuron integrating input signals into a membrane potential $u[t]$, which triggers a spike upon reaching a threshold v_{th}. In discrete time, the potential evolution is governed by:

$$\bar{u}[t] = \beta u[t-1] + I[t], \tag{7}$$

where $\bar{u}[t]$ is the potential before spike determination, $\beta \in (0,1)$ is the decay constant, and $I[t]$ is the input current. The firing state $s[t]$ is determined by the activation function $H(\cdot)$, expressed as:

$$s[t] = H(\bar{u}[t] - v_{\text{th}}), \tag{8}$$

where v_{th} is the threshold potential and H denotes the Heaviside step function. When the membrane potential reaches this threshold, the neuron generates a spike and the potential is reset. Common reset mechanisms include the soft reset, where the membrane potential is reduced by the threshold value as $u[t] = \bar{u}[t] - s[t]v_{\text{th}}$, and the hard reset, where the potential is set to a fixed value u_r such that $u[t] = \bar{u}[t](1 - s[t]) + s[t]u_r$. In LIF-based SNNs, the sparsity of the resulting spike train $s[t]$ allows for event-driven computation, which significantly reduces energy consumption compared to dense artificial neural networks.

4 LongSpike Framework

In this section, we present the LongSpike framework in detail. We begin by formulating the f-SSM block, then derive its discretization and parallelizable representation, and finally describe how it integrates with spiking neural dynamics to form the complete LongSpike architecture. By incorporating f-SSM, LongSpike replaces standard first-order Markovian state transitions with fractional-order dynamics, enabling the explicit inclusion of long-memory kernels within the spiking domain.

4.1 Fractional State Space Model Block

Let $\alpha \in (0,1)$ denote the fractional order. The f-SSM is defined as [24]:

$$D^{\alpha} h(t) = Ah(t) + Bx(t), \quad y(t) = Ch(t), \tag{9}$$

where D^{α} denotes the Caputo fractional derivative, and we assume the initial condition $h(0) = 0$. For the state matrix A, we adopt the diagonal HiPPO-D parameterization [16], which provides a structured initialization for modeling long-range dependencies.

To solve (9), we transform the differential form into a Volterra integral equation [46] by introducing the fractional kernel $k_{\alpha}(t) = \frac{t^{\alpha-1}}{\Gamma(\alpha)}$:

$$h(t) = \int_0^t k_{\alpha}(t-\tau)(Ah(\tau) + Bx(\tau))d\tau. \tag{10}$$

This integral representation explicitly demonstrates that the state $h(t)$ depends on the complete history of the system's previous states and inputs, weighted by the power-law kernel k_α. While standard numerical solvers (cf. Appendix B.2) can be used for these equations [46, 49], they often lack the parallelization properties required for efficient SSM training.

To leverage the computational advantages of state-space modeling, we adopt a parallel solving method based on a diffusive representation [50]. We express the power-law kernel $k_\alpha(t)$ as a mixture of exponentials. Using the identity $\int_0^\infty e^{-\omega t}\omega^{-\alpha}d\omega = \Gamma(1-\alpha)t^{\alpha-1}$ and the reflection formula $\Gamma(\alpha)\Gamma(1-\alpha) = \pi/\sin(\pi\alpha)$, the kernel can be rewritten as:

$$k_\alpha(t) = \frac{t^{\alpha-1}}{\Gamma(\alpha)} = \frac{\sin(\pi\alpha)}{\pi}\int_0^\infty e^{-\omega t}\omega^{-\alpha}d\omega. \tag{11}$$

This formulation reveals $k_\alpha(t)$ as an infinite continuum of exponentials. To make this computationally tractable, we approximate the continuum with a finite Sum-of-Exponentials (SOE). By selecting nodes $\{\omega_i\}_{i=1}^M \in (0,\infty)$ and weights $\{\eta_i\}_{i=1}^M$ that approximate the kernel over the range $t \in [0,T]$, we obtain:

$$k_\alpha(t) \approx \sum_{i=1}^M \eta_i e^{-\omega_i t}. \tag{12}$$

Practical methods for generating these $\{\omega_i, \eta_i\}$ pairs are detailed in Section B.3 of the Appendix. Furthermore, the $\{\omega_i, \eta_i\}$ pairs may be set as learnable in our implementation.

Subsequently, we define the auxiliary memory states $z_i(t) \in \mathbb{R}^N$ for $i = 1, \ldots, M$ as:

$$z_i(t) = \int_0^t e^{-\omega_i(t-\tau)}g(\tau)d\tau, \tag{13}$$

where $g(t) = Ah(t) + Bx(t)$. This integral is equivalent to the system of first-order linear ODEs:

$$\dot{z}_i(t) = -\omega_i z_i(t) + g(t), \quad z_i(0) = 0. \tag{14}$$

The original state $h(t)$ and output $y(t)$ are then reconstructed via the weighted sum:

$$h(t) \approx \sum_{i=1}^M \eta_i z_i(t), \quad y(t) = Ch(t). \tag{15}$$

Intuitively, this decomposition maps the singular fractional kernel onto a *multi-timescale* memory bank, where ω_i governs the decay rate and η_i acts as a mixing coefficient. Upon discretization (e.g., via ZOH or bilinear transform [16]), the dynamics of each trace follow:

$$z_i[t] = \bar{a}_i z_i[t-1] + \bar{b}_i g[t], \tag{16}$$

where $\bar{a}_i$ and $\bar{b}_i$ are the discretized parameters. Since each trace (16) corresponds to a Linear Time-Invariant (LTI) system with an exponential kernel $k_i[\ell] = \bar{b}_i\bar{a}_i^\ell$, the full state evolution $z_i = g * k_i$ can be computed via FFT-based convolution. This yields a computational complexity of $O(L\log L)$, aligning with modern efficient SSMs [15, 17]. Thus, LongSpike captures power-law memory without explicit history accumulation, ensuring the computational cost scales linearly with M.

Overall, we avoid explicit power-law history accumulation while preserving long-memory behavior through the SOE approximation. The cost scales linearly in M (typically small) and inherits the parallel training advantages of SSM-based sequence models. To further optimize memory efficiency on parallel GPU hardware, we can compute the *effective kernel* $K_{\text{eff}} = \sum_{i=1}^M \eta_i k_i$ via aggregation prior to convolution. This approach allows us to compute the final state contribution without ever materializing the M intermediate traces z_i, significantly reducing memory I/O.

4.2 LongSpike Block

The LongSpike architecture integrates fractional state-space dynamics with LIF neurons. The output $y[t]$ from the preceding f-SSM block acts as the injection current driving the neuron's membrane potential:

$$\bar{u}[t] = \beta u[t-1] + y[t], \quad s[t] = H(\bar{u}[t] - v_{th}), \tag{17}$$

where $H(\cdot)$ is the Heaviside step function, $\beta \in (0, 1)$ is the membrane decay constant, and v_{th} is a learnable threshold. This coupling enables the model to encode temporal information into sparse, energy-efficient spike representations.

To circumvent the sequential bottleneck inherent in iterative LIF simulation, we adopt a parallelization strategy utilizing a Surrogate Dynamic Network (SDN) [19]. The SDN serves as a non-autoregressive mapping that predicts the entire spike train $s_{1:L}$ directly from the input current sequence $y_{1:L}$:

$$s_{1:L} = f_{\text{SDN}}(y_{1:L}). \tag{18}$$

We instantiate the SDN as a CNN layer. This design is theoretically grounded in the observation that the linear recurrence of the LIF dynamics can be effectively approximated by convolutional operators with local receptive fields. During inference, the SDN replaces step-by-step integration with parallel prediction, reducing the sequential complexity to $O(1)$ (constant parallel depth) and significantly enhancing computational throughput.

The full LongSpike architecture is shown in Fig. 1, where the model stacks multiple LongSpike blocks on top of an encoder, followed by a decoder and an output head.

4.3 Theoretical Analysis: Long-Memory Properties

In this section, we theoretically analyze the memory characteristics of the f-SSM module relative to standard SSMs. We demonstrate that while both systems converge to identical steady states under constant excitation, f-SSM exhibits Mittag–Leffler relaxation characterized by power-law decay. This reveals a persistent, heavy-tailed temporal dependence that is structurally absent in standard first-order models.

Proposition 1 (Long-Memory Dynamics)**.** *Consider a scalar state-space model governed by parameters $A = -\lambda$ (with $\lambda > 0$) and $B = 1$, subject to a constant input $x(t) = x_c$ and initial condition $h(0) = 0$. Let the system evolve according to a derivative of order $\alpha \in (0, 1]$. The state trajectories $h(t)$ for the standard ($\alpha = 1$) and fractional ($0 < \alpha < 1$) cases are given by:*

$$\text{Standard SSM:} \quad h(t) = \frac{x_c}{\lambda} - \frac{x_c}{\lambda} e^{-\lambda t}, \tag{19}$$

$$f\text{-SSM:} \quad h(t) = \frac{x_c}{\lambda} - \frac{x_c}{\lambda} E_\alpha\left(-\lambda t^\alpha\right), \tag{20}$$

where $E_\alpha(z) = \sum_{k=0}^{\infty} \frac{z^k}{\Gamma(\alpha k+1)}$ is the Mittag–Leffler function [46].

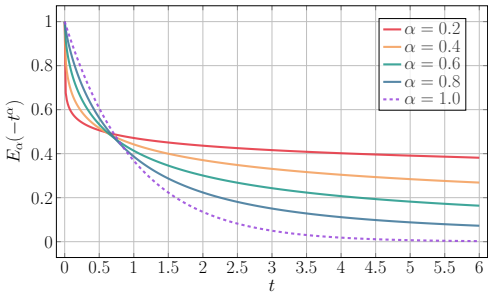


Figure 2: The Mittag-Leffler function $E_\alpha(-\lambda t^\alpha)$ for varying α. For $\alpha = 1$ (purple dashed), the function recovers the exponential decay $e^{-\lambda t}$. For $\alpha < 1$, the function transitions to an algebraic decay tail, where lower α values indicate stronger long-memory retention.

The asymptotic behavior of the Mittag–Leffler function, illustrated in Fig. 2, highlights the fundamental difference in memory retention:

- Exponential Decay ($\alpha = 1$): The function simplifies to $E_1(z) = e^z$, recovering the exponentially decaying relaxation of classical first-order systems.
- Algebraic Decay ($\alpha < 1$): For large t, the function exhibits a power-law tail:

$$E_\alpha(-\lambda t^\alpha) \sim \frac{1}{\lambda \Gamma(1-\alpha) t^\alpha}, \quad \text{as } t \to \infty. \tag{21}$$

Thus, the memory trace in f-SSM decays algebraically, significantly slower than the exponential rate of standard SSMs.

Remark 2 (Mechanism of Persistence)**.** *While both systems share the same equilibrium point $h_\infty = x_c/\lambda$, their transient relaxation reflects distinct underlying mechanisms. Standard SSMs are Markovian and the influence of past states vanishes exponentially. In contrast, the* f*-SSM is intrinsically non-Markovian due to the non-local nature of the Caputo fractional derivative. The associated history-dependent kernel induces Mittag–Leffler relaxation with heavy tails, ensuring that information from the distant past exerts a non-negligible influence on the current state, a critical property for capturing long-range dependencies in sequence modeling.*

5 Experiments

We evaluate LongSpike on a diverse range of sequence modeling tasks, including the Long Range Arena (LRA) benchmark, large-scale language modeling on WikiText-103, and sequential audio classification on Speech Commands. Additional experimental results are provided in Appendix F. Details of experimental settings are provided in Appendix D.

In addition to benchmark results, we analyze the role of fractional dynamics and the efficiency of LongSpike in terms of computational complexity and energy-related behavior (see Section 5.3).

5.1 Datasets

LRA. The Long-Range Arena (LRA) benchmark [51] evaluates models on long-context sequences ranging from 1K to 16K steps, across diverse modalities including text, visual data, and mathematical expressions. It contains six tasks, including LISTOPS, TEXT, RETRIEVAL, IMAGE, PATHFINDER, and PATH-X, which cover sequence classification, document matching, image recognition, and pathfinding. These tasks are widely used to assess a model's ability to capture long-range dependencies and process extended sequences efficiently.

WikiText-103. WikiText-103 [52] is a large-scale language modeling dataset containing over 100 million tokens from high-quality Wikipedia articles. The dataset consists of long and coherent documents and is commonly used for evaluating language models on extended textual contexts. We use perplexity (PPL) as the evaluation metric to measure the quality of next-token prediction.

Speech Commands. The Speech Commands dataset [53] contains over 100,000 one-second audio clips of 35 spoken English words (e.g., "yes," "no," "up," and "down") from thousands of speakers, sampled at 16 kHz. The task is to classify the spoken word in each audio sequence, requiring the model to capture temporal acoustic patterns under variations in speaking speed, tone, and background noise.

5.2 Experimental Results

Table 1: Performance comparison of LongSpike with other works on LRA benchmark. We take the 50% accuracy for the absence of Path-X accuracy as did in the work of S4D, then compute the overall average metrics across all tasks as AVG. LongSpike results are reported as mean $\pm$ standard deviation over three random seeds.

Model	SNN	LISTOPS	TEXT	RETRIEVAL	IMAGE	PATHFINDER	Path-X	AVG
Transformer	No	36.37	64.27	57.46	42.44	71.40	-	53.66
LMUFormer	No	34.43	68.27	78.65	54.16	69.90	-	59.24
S4	No	58.35	76.02	87.09	87.26	86.05	88.10	80.48
S4D-Lin	No	**60.52**	**86.97**	**90.96**	**87.93**	**93.96**	**92.8**	**85.52**
Spiking LMUFormer	Yes	37.30	65.80	79.76	55.65	72.68	-	60.20
Binary S4D	Yes	54.80	82.50	85.03	82.00	82.60	61.20	74.69
P-SpikeSSM	Yes	58.20	81.20	88.53	82.40	84.80	-	74.18
SpikingSSM	Yes	60.23	80.41	88.77	88.21	93.51	94.82	84.33
LongSpike	Yes	$\mathbf{60.78 \pm 0.31}$	$\mathbf{88.11 \pm 0.08}$	$\mathbf{90.00 \pm 0.24}$	$\mathbf{88.80 \pm 0.38}$	$\mathbf{94.17 \pm 0.13}$	$\mathbf{95.18 \pm 0.31}$	**86.17**

We first evaluate LongSpike on the six tasks of the LRA benchmark. As shown in Table 1, LongSpike achieves an average score of 86.17%, outperforming prior spiking baselines while remaining competitive with strong non-spiking SSMs. Compared with SpikingSSM, LongSpike improves the overall LRA average from 84.33% to 86.17%. Moreover, the small standard deviations across three random seeds indicate that the performance improvements are stable across different training runs. The improvements are particularly pronounced on the TEXT and RETRIEVAL tasks, demonstrating the effectiveness of the proposed fractional dynamics for long-range sequence modeling. For illustration, Figure 3 shows the loss and accuracy curves for the RETRIEVAL task, where LongSpike consistently surpasses competing models. It also remains highly competitive on more complex tasks such as PATH-X, highlighting its robustness across diverse sequence modeling challenges. To better understand the source of these improvements, we further evaluate a vanilla (non-spiking) fractional SSM. As shown in Table 9 in the appendix, it consistently outperforms its integer-order counterpart.

Table 2: Performance comparison on WikiText-103.

Model	SNN	PPL	Params	Spiking Rate
Transformer	No	20.51	231M	-
S4	No	20.95	249M	-
SpikeGPT	Yes	39.75	213M	-
SpikingSSM	Yes	33.94	75M	26.40%
LongSpike	Yes	**32.31**	75M	27.05%

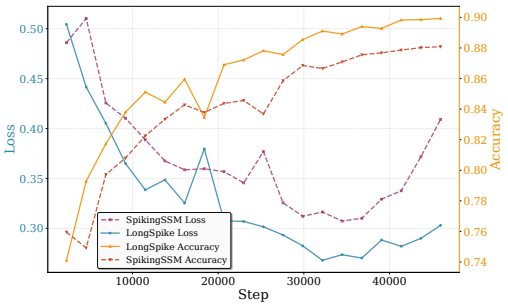


Figure 3: Loss/accuracy trends on the LRA RETRIEVAL task.

In addition to the LRA benchmark, we evaluate LongSpike on the large-scale WikiText-103 dataset to assess its effectiveness on real-world language modeling tasks. As shown in Table 2, LongSpike outperforms SpikingSSM with comparable model size, and achieves lower perplexity than SpikeGPT [33] despite using significantly fewer parameters. Moreover, its spiking rate remains comparable to that of SpikingSSM, indicating that the performance gain is not obtained by substantially increased neural activity. These results further narrow the performance gap with ANN-based models, highlighting the advantage of fractional dynamics for long-context language modeling.

Finally, we evaluate LongSpike on the Speech Commands dataset. As shown in Table 3, it achieves a competitive accuracy of 96.31%, surpassing the spiking baseline SpikingSSM. These results indicate that LongSpike can effectively model temporal dependencies in sequential audio data.

Table 3: Performance comparison on Speech Commands dataset.

Model	#Params	Accuracy(%)
S4D-Lin	306K	96.25
SpikingSSM	305K	96.09
LongSpike	305K	**96.31**

5.3 Efficiency and Fractional Approximation Analysis

A key feature of LongSpike is the use of a sum-of-exponentials approximation to implement fractional dynamics. When reduced to a single exponential ($M = 1, \omega = 0, \eta = 1$) in Equation (12), corresponding to the kernel $k_\alpha(t)$ with $\alpha = 1$, the model recovers the standard SpikingSSM [19], yielding identical performance. Thus, LongSpike can be viewed as a fractional-order generalization of SpikingSSM, where increasing M improves the approximation of the long-memory kernel at the cost of additional computation.

In our experiments, we use a small number of exponential terms ($M = 2$) to balance modeling capacity and efficiency. This setting improves accuracy while preserving sparse neural activity. For example, on the TEXT task of the LRA benchmark, LongSpike achieves 88.11% accuracy, substantially outperforming SpikingSSM with 80.41%. Meanwhile, the spiking rate remains low (6.46% vs. 6.75%), indicating that the performance gain does not come from substantially denser spiking activity, but from the enhanced long-memory dynamics introduced by the fractional kernel.

From a computational perspective, LongSpike preserves the efficient SSM computation pattern. SpikingSSM has a forward-pass complexity of $O(HNL + BHL \log L)$, dominated by FFT-based convolution, while LongSpike only increases kernel construction from $O(HNL)$ to $O(MHNL)$ and leaves the FFT convolution unchanged. For small M (e.g., $M = 2$), this introduces only a constant-factor overhead; moreover, the memory complexity can be kept at $O(HNL)$ with a loop-based implementation, matching SpikingSSM. In practice, the computation can be efficiently parallelized on GPUs with CUDA-level optimizations, reducing intermediate memory materialization and runtime overhead. A full analysis of time complexity, space complexity, and spiking-rate-based energy proxy is provided in Appendix E.

6 Conclusion

In this work, we introduced LongSpike, a novel spiking neural network framework designed to enhance heavy-tailed memory modeling in traditional first-order SNN architectures. By integrating fractional-order state-space modeling (f-SSM) into the spiking domain, we have shifted the

paradigm from Markovian state transitions to dynamics that inherently capture long-range temporal dependencies through power-law memory kernels. Our approach successfully addresses the historical computational challenges of fractional operators. By reformulating these dynamics into a parallelizable state-space representation, LongSpike achieves the best of both worlds: the biological plausibility and energy efficiency of SNNs, and the high-performance parallel training capabilities of modern SSMs. Empirical results across LRA, WikiText-103, and Speech Commands benchmarks demonstrate that LongSpike not only outperforms state-of-the-art SNNs in accuracy but also preserves sparse synaptic computation in modeling sequences with fine-grained, long-term temporal structures.

7 Limitations and Future Work

We have not yet deployed LongSpike on dedicated neuromorphic hardware platforms such as Intel Loihi, as our current work mainly focuses on the algorithmic and modeling aspects of f-SSM. Nevertheless, LongSpike remains naturally compatible with neuromorphic deployment. During inference, the model operates entirely through recurrent state updates without dense sequence-level matrix multiplications, which is favorable for hardware-efficient computation. We provide a spiking-rate-based energy analysis in Appendix E.2; future work will investigate hardware-level implementations and optimized sparse approximations of the fractional kernel to further improve efficiency and evaluate real-world latency and energy consumption.

Acknowledgments and Disclosure of Funding

This work was supported in part by the National Natural Science Foundation of China (NSFC) under Grants 62225207, 62436008, 62576326, and 12301491.

References

[1] Mike Schuster and Kuldip K Paliwal. Bidirectional recurrent neural networks. *IEEE transactions on Signal Processing*, 45(11):2673–2681, 1997.

[2] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. In *International conference on machine learning*, pages 1310–1318. Pmlr, 2013.

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

[4] 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.

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

[6] Samanwoy Ghosh-Dastidar and Hojjat Adeli. Spiking neural networks. *International journal of neural systems*, 19(04):295–308, 2009.

[7] Jun Haeng Lee, Tobi Delbruck, and Michael Pfeiffer. Training deep spiking neural networks using backpropagation. *Frontiers in neuroscience*, 10:508, 2016.

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

[9] Jason K Eshraghian, Max Ward, Emre Neftci, Xinxin Wang, Gregor Lenz, Girish Dwivedi, Mohammed Bennamoun, Doo Seok Jeong, and Wei D Lu. Training spiking neural networks using lessons from deep learning. *Proceedings of the IEEE*, 111(9):1016–1054, 2023.

[10] 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.

[11] Sinong Wang, Belinda Z Li, Madian Khabsa, Han Fang, and Hao Ma. Linformer: Self-attention with linear complexity. *arXiv preprint arXiv:2006.04768*, 2020.

[12] Nikita Kitaev, Łukasz Kaiser, and Anselm Levskaya. Reformer: The efficient transformer. *arXiv preprint arXiv:2001.04451*, 2020.

[13] Iz Beltagy, Matthew E Peters, and Arman Cohan. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*, 2020.

[14] Albert Gu, Tri Dao, Stefano Ermon, Atri Rudra, and Christopher Ré. Hippo: Recurrent memory with optimal polynomial projections. *Advances in neural information processing systems*, 33: 1474–1487, 2020.

[15] Albert Gu, Karan Goel, and Christopher Ré. Efficiently modeling long sequences with structured state spaces. *arXiv preprint arXiv:2111.00396*, 2021.

[16] Albert Gu, Karan Goel, Ankit Gupta, and Christopher Ré. On the parameterization and initialization of diagonal state space models. *Advances in Neural Information Processing Systems*, 35:35971–35983, 2022.

[17] Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. In *First conference on language modeling*, 2024.

[18] Tri Dao and Albert Gu. Transformers are ssms: Generalized models and efficient algorithms through structured state space duality. *arXiv preprint arXiv:2405.21060*, 2024.

[19] Shuaijie Shen, Chao Wang, Renzhuo Huang, Yan Zhong, Qinghai Guo, Zhichao Lu, Jianguo Zhang, and Luziwei Leng. Spikingssms: Learning long sequences with sparse and parallel spiking state space models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 20380–20388, 2025.

[20] Yu Du, Xu Liu, and Yansong Chua. Spiking structured state space model for monaural speech enhancement. In *ICASSP 2024-2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 766–770. IEEE, 2024.

[21] Changze Lv, Tianlong Li, Weiming Qiao, Xiaohua Wang, Muling Wu, Wenhao Liu, Shihan Dou, Xiaoqing Zheng, and Xuanjing Huang. Spikebert: A language spikformer learned from bert with knowledge distillation. *Neural Networks*, page 108482, 2025.

[22] Igor Podlubny. *Fractional differential equations: an introduction to fractional derivatives, fractional differential equations, to methods of their solution and some of their applications*, volume 198. elsevier, 1998.

[23] Jean-Claude Trigeassou, Nezha Maamri, and Alain Oustaloup. The infinite state approach: Origin and necessity. *Computers & Mathematics with Applications*, 66(5):892–907, 2013.

[24] Denis Matignon. Stability results for fractional differential equations with applications to control processing. In *Computational engineering in systems applications*, volume 2, pages 963–968. Lille, France, 1996.

[25] Andrzej Dzieliński and Dominik Sierociuk. Stability of discrete fractional order state-space systems. *Journal of Vibration and Control*, 14(9-10):1543–1556, 2008.

[26] Diethelm Kai. The analysis of fractional differential equations. an application-oriented exposition using differential operators of caputo type. *Lecture Notes in Mathematics*, 2004.

[27] Dumitru Baleanu, Kai Diethelm, Enrico Scalas, and Juan J Trujillo. *Fractional calculus: models and numerical methods*, volume 3. World Scientific, 2012.

[28] Chengjie Ge, Yufeng Peng, Zihao Li, Qiyu Kang, Xueyang Fu, Xuhao Li, Qixin Zhang, Junhao Ren, and Zheng-Jun Zha. Fractional-order spiking neural network. In *Proc. Int. Conf. Learn. Representations*, 2026.

[29] Yutao Sun, Li Dong, Shaohan Huang, Shuming Ma, Yuqing Xia, Jilong Xue, Jianyong Wang, and Furu Wei. Retentive network: A successor to transformer for large language models. *arXiv preprint arXiv:2307.08621*, 2023.

[30] Songlin Yang, Bailin Wang, Yikang Shen, Rameswar Panda, and Yoon Kim. Gated linear attention transformers with hardware-efficient training. *arXiv preprint arXiv:2312.06635*, 2023.

[31] Ankit Gupta, Albert Gu, and Jonathan Berant. Diagonal state spaces are as effective as structured state spaces. *Advances in neural information processing systems*, 35:22982–22994, 2022.

[32] Emre O Neftci, Hesham Mostafa, and Friedemann Zenke. Surrogate gradient learning in spiking neural networks: Bringing the power of gradient-based optimization to spiking neural networks. *IEEE Signal Processing Magazine*, 36(6):51–63, 2019.

[33] Rui-Jie Zhu, Qihang Zhao, Guoqi Li, and Jason K Eshraghian. Spikegpt: Generative pre-trained language model with spiking neural networks. *arXiv preprint arXiv:2302.13939*, 2023.

[34] Zeyu Liu, Gourav Datta, Anni Li, and Peter Anthony Beerel. Lmuformer: Low complexity yet powerful spiking model with legendre memory units. *arXiv preprint arXiv:2402.04882*, 2024.

[35] Aaron Voelker, Ivana Kajić, and Chris Eliasmith. Legendre memory units: Continuous-time representation in recurrent neural networks. *Advances in neural information processing systems*, 32, 2019.

[36] Matei-Ioan Stan and Oliver Rhodes. Learning long sequences in spiking neural networks. *Scientific Reports*, 14(1):21957, 2024.

[37] Malyaban Bal and Abhronil Sengupta. P-spikessm: Harnessing probabilistic spiking state space models for long-range dependency tasks. International Conference on Representation Learning (ICLR) 2025, 2025.

[38] Francesco Mainardi. *Fractional calculus and waves in linear viscoelasticity: an introduction to mathematical models*. World Scientific, 2022.

[39] Wenjun Cui, Qiyu Kang, Xuhao Li, Kai Zhao, Wee Peng Tay, Weihua Deng, and Yidong Li. Neural variable-order fractional differential equation networks. In *Proc. AAAI Conference on Artificial Intelligence*, Philadelphia, USA, Feb. 2025.

[40] Qiyu Kang, Kai Zhao, Qinxu Ding, Feng Ji, Xuhao Li, Wenfei Liang, Yang Song, and Wee Peng Tay. Unleashing the potential of fractional calculus in graph neural networks with FROND. In *Proc. International Conference on Learning Representations*, Vienna, Austria, 2024.

[41] Qiyu Kang, Wenjun Cui, Xuhao Li, Yuxin Ma, Xueyang Fu, Wee Peng Tay, Yidong Li, and Zheng-Jun Zha. Neural fractional attention differential equations. In *Advances in Neural Information Processing Systems*, San Diego, USA, Dec. 2025.

[42] Qiyu Kang, Kai Zhao, Yang Song, Yihang Xie, Yanan Zhao, Sijie Wang, Rui She, and Wee Peng Tay. Coupling graph neural networks with fractional order continuous dynamics: A robustness study. In *Proc. AAAI Conference on Artificial Intelligence*, Vancouver, Canada, Feb. 2024.

[43] Gabriel Nobis, Maximilian Springenberg, Marco Aversa, Michael Detzel, Rembert Daems, Roderick Murray-Smith, Shinichi Nakajima, Sebastian Lapuschkin, Stefano Ermon, Tolga Birdal, et al. Generative fractional diffusion models. *Advances in Neural Information Processing Systems*, 37:25469–25509, 2024.

[44] Wondimu Teka, Toma M Marinov, and Fidel Santamaria. Neuronal spike timing adaptation described with a fractional leaky integrate-and-fire model. *PLoS computational biology*, 10(3): e1003526, 2014.

[45] Yabin Deng, Bijing Liu, Zenan Huang, Xiaojie Liu, Shan He, Qiuhong Li, and Donghui Guo. Fractional spiking neuron: Fractional leaky integrate-and-fire circuit described with dendritic fractal model. *IEEE Transactions on Biomedical Circuits and Systems*, 16(6):1375–1386, 2022.

[46] Kai Diethelm. *The analysis of fractional differential equations: an application-oriented exposition using differential operators of Caputo type*, volume 2004. Lect. Notes Math., 2010.

[47] Daniel Y Fu, Tri Dao, Khaled K Saab, Armin W Thomas, Atri Rudra, and Christopher Ré. Hungry hungry hippos: Towards language modeling with state space models. *arXiv preprint arXiv:2212.14052*, 2022.

[48] Richard B Stein. Some models of neuronal variability. *Biophysical journal*, 7(1):37–68, 1967.

[49] Kai Zhao, Xuhao Li, Qiyu Kang, Feng Ji, Qinxu Ding, Yanan Zhao, Wenfei Liang, and Wee Peng Tay. Distributed-order fractional graph operating network. *Advances in Neural Information Processing Systems*, 37:103442–103475, 2024.

[50] Shidong Jiang, Jiwei Zhang, Qian Zhang, and Zhimin Zhang. Fast evaluation of the caputo fractional derivative and its applications to fractional diffusion equations. *Communications in Computational Physics*, 21(3):650–678, 2017.

[51] Yi Tay, Mostafa Dehghani, Samira Abnar, Yikang Shen, Dara Bahri, Philip Pham, Jinfeng Rao, Liu Yang, Sebastian Ruder, and Donald Metzler. Long range arena: A benchmark for efficient transformers. *arXiv preprint arXiv:2011.04006*, 2020.

[52] Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. *arXiv preprint arXiv:1609.07843*, 2016.

[53] Pete Warden. Speech commands: A dataset for limited-vocabulary speech recognition. *arXiv preprint arXiv:1804.03209*, 2018.

[54] Alex Sherstinsky. Fundamentals of recurrent neural network (rnn) and long short-term memory (lstm) network. *Physica D: Nonlinear Phenomena*, 404:132306, 2020.

[55] Alexander Rehmer and Andreas Kroll. On the vanishing and exploding gradient problem in gated recurrent units. *IFAC-PapersOnLine*, 53(2):1243–1248, 2020.

[56] Igor Podlubny. *Fractional Differential Equations*. Academic Press, 1999.

[57] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers)*, pages 4171–4186, 2019.

[58] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[59] Mike Davies, Narayan Srinivasa, Tsung-Han Lin, Gautham Chinya, Yongqiang Cao, Sri Harsha Choday, Georgios Dimou, Prasad Joshi, Nabil Imam, Shweta Jain, et al. Loihi: A neuromorphic manycore processor with on-chip learning. *Ieee Micro*, 38(1):82–99, 2018.

[60] Christian Bakke Vennerød, Adrian Kjærran, and Erling Stray Bugge. Long short-term memory rnn. *arXiv preprint arXiv:2105.06756*, 2021.

[61] Shimul Kanti Nath, Sujan Kumar Das, Sanjoy Kumar Nandi, Chen Xi, Camilo Verbel Marquez, Armando Rúa, Mutsunori Uenuma, Zhongrui Wang, Songqing Zhang, Rui-Jie Zhu, et al. Optically tunable electrical oscillations in oxide-based memristors for neuromorphic computing. *Advanced Materials*, 36(25):2400904, 2024.

[62] Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N Dauphin. Convolutional sequence to sequence learning. In *International conference on machine learning*, pages 1243–1252. PMLR, 2017.

[63] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

[64] Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. Palm: Scaling language modeling with pathways. *Journal of Machine Learning Research*, 24(240): 1–113, 2023.

[65] Liliang Ren, Yang Liu, Yadong Lu, Yelong Shen, Chen Liang, and Weizhu Chen. Samba: Simple hybrid state space models for efficient unlimited context language modeling. *arXiv preprint arXiv:2406.07522*, 2024.

[66] Opher Lieber, Barak Lenz, Hofit Bata, Gal Cohen, Jhonathan Osin, Itay Dalmedigos, Erez Safahi, Shaked Meirom, Yonatan Belinkov, Shai Shalev-Shwartz, et al. Jamba: A hybrid transformer-mamba language model. *arXiv preprint arXiv:2403.19887*, 2024.

[67] Xingrun Xing, Zheng Liu, Shitao Xiao, Boyan Gao, Yiming Liang, Wanpeng Zhang, Haokun Lin, Guoqi Li, and Jiajun Zhang. Efficientllm: Scalable pruning-aware pretraining for architecture-agnostic edge language models. *arXiv preprint arXiv:2502.06663*, 2025.

[68] Malyaban Bal and Abhronil Sengupta. Spikingbert: Distilling bert to train spiking language models using implicit differentiation. In *Proceedings of the AAAI conference on artificial intelligence*, volume 38, pages 10998–11006, 2024.

[69] Zhaokun Zhou, Yuesheng Zhu, Chao He, Yaowei Wang, Shuicheng Yan, Yonghong Tian, and Li Yuan. Spikformer: When spiking neural network meets transformer. *arXiv preprint arXiv:2209.15425*, 2022.

[70] Zhaokun Zhou, Kaiwei Che, Wei Fang, Keyu Tian, Yuesheng Zhu, Shuicheng Yan, Yonghong Tian, and Li Yuan. Spikformer v2: Join the high accuracy club on imagenet with an snn ticket. *arXiv preprint arXiv:2401.02020*, 2024.

[71] Jiayu Zhang, Jiangrong Shen, Zeke Wang, Qinghai Guo, Rui Yan, Gang Pan, and Huajin Tang. Spikingminilm: energy-efficient spiking transformer for natural language understanding. *Science China Information Sciences*, 67(10):200406, 2024.

[72] Xingrun Xing, Zheng Zhang, Ziyi Ni, Shitao Xiao, Yiming Ju, Siqi Fan, Yequan Wang, Jiajun Zhang, and Guoqi Li. Spikelm: Towards general spike-driven language modeling via elastic bi-spiking mechanisms. *arXiv preprint arXiv:2406.03287*, 2024.

[73] Xingrun Xing, Boyan Gao, Zheng Zhang, David A Clifton, Shitao Xiao, Li Du, Guoqi Li, and Jiajun Zhang. Spikellm: Scaling up spiking neural network to large language models via saliency-based spiking. *arXiv preprint arXiv:2407.04752*, 2024.

[74] Yan Zhong, Ruoyu Zhao, Chao Wang, Qinghai Guo, Jianguo Zhang, Zhichao Lu, and Luziwei Leng. Spike-ssm: A sparse, precise, and efficient spiking state space model for long sequences learning. *arXiv preprint arXiv:2410.17268*, 2024.

[75] Yangfan Hu, Qian Zheng, Guoqi Li, Huajin Tang, and Gang Pan. Toward large-scale spiking neural networks: A comprehensive survey and future directions. *arXiv preprint arXiv:2409.02111*, 2024.

[76] Guanqiao Qu, Qiyuan Chen, Wei Wei, Zheng Lin, Xianhao Chen, and Kaibin Huang. Mobile edge intelligence for large language models: A contemporary survey. *IEEE Communications Surveys & Tutorials*, 2025.

[77] Cheng Kevin Qu, Andrew Ly, and Pulin Gong. Fractional neural attention for efficient multiscale sequence processing. *arXiv preprint arXiv:2511.10208*, 2025.

[78] Songlin Yang, Bailin Wang, Yu Zhang, Yikang Shen, and Yoon Kim. Parallelizing linear transformers with the delta rule over sequence length. *Advances in neural information processing systems*, 37:115491–115522, 2024.

[79] Yuhong Chou, Man Yao, Kexin Wang, Yuqi Pan, Rui-Jie Zhu, Jibin Wu, Yiran Zhong, Yu Qiao, Bo Xu, and Guoqi Li. Metala: Unified optimal linear approximation to softmax attention map. *Advances in Neural Information Processing Systems*, 37:71034–71067, 2024.

[80] Songlin Yang, Jan Kautz, and Ali Hatamizadeh. Gated delta networks: Improving mamba2 with delta rule. *arXiv preprint arXiv:2412.06464*, 2024.

[81] Qiyu Kang, Xuhao Li, Kai Zhao, Wenjun Cui, Yanan Zhao, Weihua Deng, and Wee Peng Tay. Efficient training of neural fractional-order differential equation via adjoint backpropagation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 17750–17759, 2025.

[82] Vasily E Tarasov. *Fractional dynamics: applications of fractional calculus to dynamics of particles, fields and media*. Springer Science & Business Media, 2011.

[83] Pierre Duhamel and Martin Vetterli. Fast fourier transforms: a tutorial review and a state of the art. *Signal processing*, 19(4):259–299, 1990.

[84] Yann LeCun. The mnist database of handwritten digits. *http://yann. lecun. com/exdb/mnist/*, 1998.

[85] Quoc V Le, Navdeep Jaitly, and Geoffrey E Hinton. A simple way to initialize recurrent networks of rectified linear units. *arXiv preprint arXiv:1504.00941*, 2015.

[86] Bo Peng, Eric Alcaide, Quentin Anthony, Alon Albalak, Samuel Arcadinho, Stella Biderman, Huanqi Cao, Xin Cheng, Michael Chung, Leon Derczynski, et al. Rwkv: Reinventing rnns for the transformer era. In *Findings of the association for computational linguistics: EMNLP 2023*, pages 14048–14077, 2023.

[87] Zachary C Lipton, John Berkowitz, and Charles Elkan. A critical review of recurrent neural networks for sequence learning. *arXiv preprint arXiv:1506.00019*, 2015.

[88] Tri Dao, Dan Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. Flashattention: Fast and memory-efficient exact attention with io-awareness. *Advances in neural information processing systems*, 35:16344–16359, 2022.

[89] Tri Dao. Flashattention-2: Faster attention with better parallelism and work partitioning. *arXiv preprint arXiv:2307.08691*, 2023.

[90] Harsh Mehta, Ankit Gupta, Ashok Cutkosky, and Behnam Neyshabur. Long range language modeling via gated state spaces. *arXiv preprint arXiv:2206.13947*, 2022.

[91] Olaf Sporns. *Networks of the Brain*. MIT press, 2016.

[92] Alan L Hodgkin and Andrew F Huxley. A quantitative description of membrane current and its application to conduction and excitation in nerve. *The Journal of physiology*, 117(4):500, 1952.

[93] Holden T Lee and Daniel M Monahan. Exploring fractional leaky integrate-and-fire neurons in spiking neural networks. *Artem Arefev*, page 138.

[94] Anand Subramoney, Khaleelulla Khan Nazeer, Mark Schöne, Christian Mayr, and David Kappel. Efficient recurrent architectures through activity sparsity and sparse back-propagation through time. *arXiv preprint arXiv:2206.06178*, 2022.

[95] Mark Schöne, Neeraj Mohan Sushma, Jingyue Zhuge, Christian Mayr, Anand Subramoney, and David Kappel. Scalable event-by-event processing of neuromorphic sensory signals with deep state-space models. In *2024 International Conference on Neuromorphic Systems (ICONS)*, pages 124–131. IEEE, 2024.

[96] Man Yao, Jiakui Hu, Tianxiang Hu, Yifan Xu, Zhaokun Zhou, Yonghong Tian, Bo XU, and Guoqi Li. Spike-driven transformer v2: Meta spiking neural network architecture inspiring the design of next-generation neuromorphic chips. In B. Kim, Y. Yue, S. Chaudhuri, K. Fragkiadaki, M. Khan, and Y. Sun, editors, *International Conference on Learning Representations*, volume 2024, pages 52885–52907, 2024. URL `https://proceedings.iclr.cc/paper_files/paper/2024/file/e9882f7f7c44a10acc01132302bac9d8-Paper-Conference.pdf`.

[97] Mark Horowitz. 1.1 computing's energy problem (and what we can do about it). In *2014 IEEE international solid-state circuits conference digest of technical papers (ISSCC)*, pages 10–14. IEEE, 2014.

[98] Boyan Li, Luziwei Leng, Shuaijie Shen, Kaixuan Zhang, Jianguo Zhang, Jianxing Liao, and Ran Cheng. Efficient deep spiking multilayer perceptrons with multiplication-free inference. *IEEE transactions on neural networks and learning systems*, 36(4):7542–7554, 2024.

A Overview

This supplementary material provides extended theoretical details, implementation information, and additional experimental results that support the analysis and conclusions presented in the main paper. The content is organized as follows:

1. Additional technical details on fractional calculus, including conventional numerical methods for solving fractional differential equations and methods for constructing the sum-of-exponentials (SOE) approximation, are provided in Section B.
2. Detailed derivations of the standard and fractional state-space solutions used in the theoretical analysis are presented in Section C.
3. Experimental settings, implementation details, and hyperparameter configurations for all evaluated benchmarks are summarized in Section D.
4. Computational complexity, memory consumption, and energy-efficiency analyses of LongSpike are provided in Section E.
5. Extended experimental results, including evaluations on additional sequence modeling benchmarks and controlled studies of fractional dynamics, are presented in Section F.
6. Broader impacts are discussed in Appendix G.

B More Technical Details

B.1 More About Fractional Calculus

This section provides additional mathematical background for readers less familiar with fractional calculus; it is not required for understanding the main method.

In Section 3.1 of the main paper, we introduced the left Caputo fractional derivative and discussed numerical schemes for solving fractional-order ordinary differential equations (ODEs). Here we provide additional background on fractional calculus underlying our approach.

It is worth emphasizing that fractional derivatives are not uniquely defined; multiple formulations exist with different analytical properties. In the main paper, unless otherwise specified, we adopt the left-sided Caputo fractional derivative, denoted by D^α, due to its suitability for modeling physical systems and compatibility with standard initial conditions.

B.1.1 Classical Derivatives and Integrals

For a scalar function $y(t)$, the first-order derivative describes its instantaneous rate of change:

$$\frac{dy(t)}{dt} = \dot{y}(t) = \lim_{\Delta t \to 0} \frac{y(t+\Delta t) - y(t)}{\Delta t}. \tag{22}$$

Let J denote the integral operator that maps a Riemann-integrable function $y(t)$ on $[0, T]$ to its antiderivative starting from a:

$$Jy(t) := \int_a^t y(\tau)\, d\tau, \quad t \in [0, T]. \tag{23}$$

For a positive integer $n \in \mathbb{N}^+$, we define the n-fold integral operator recursively as $J^n := J \circ J^{n-1}$for $n \geq 2$, and $J^1 = J$. It can be shown via repeated integration by parts [26] that:

$$J^n y(t) = \frac{1}{(n-1)!} \int_a^t (t-\tau)^{n-1} y(\tau) d\tau. \tag{24}$$

B.1.2 Fractional Integrals

Fractional calculus generalizes classical integration by extending the order of integration from integers to real (or even complex) values. Among various definitions, the Riemann–Liouville (RL) fractional integral is one of the most fundamental [82].

For $\alpha \in \mathbb{R}^+$, the left- and right-sided RL fractional integrals are defined as:

$$J^{\alpha}_{\text{left}} y(t) = \frac{1}{\Gamma(\alpha)} \int_a^t (t-\tau)^{\alpha-1} y(\tau)\, d\tau, \tag{25}$$

$$J^{\alpha}_{\text{right}} y(t) = \frac{1}{\Gamma(\alpha)} \int_t^b (\tau-t)^{\alpha-1} y(\tau)\, d\tau, \tag{26}$$

where $\Gamma(\alpha)$ is the Gamma function, which generalizes the factorial to non-integer arguments.

Compared with (24), fractional integrals allow the order parameter to vary continuously, providing a smooth interpolation between different levels of accumulation and enabling richer modeling flexibility.

B.1.3 Fractional Derivatives

Extending differentiation to non-integer orders is more subtle, and several definitions exist. Among them, the Riemann–Liouville (RL) and Caputo derivatives are the most commonly used.

Riemann–Liouville fractional derivative. The RL fractional derivative is defined by applying an integer-order derivative after a fractional integral:

$${}^{\text{RL}}D^{\alpha} y(t) := \frac{d^m}{dt^m} J^{m-\alpha}_{\text{left}} y(t) = \frac{1}{\Gamma(m-\alpha)} \frac{d^m}{dt^m} \int_a^t \frac{y(\tau)\, d\tau}{(t-\tau)^{\alpha-m+1}}, \tag{27}$$

$${}^{\text{RL}}D^{\alpha}_{b-} y(t) := (-1)^m \frac{d^m}{dt^m} J^{m-\alpha}_{\text{right}} y(t) = \frac{(-1)^m}{\Gamma(m-\alpha)} \frac{d^m}{dt^m} \int_t^b \frac{y(\tau)\, d\tau}{(\tau-t)^{\alpha-m+1}}. \tag{28}$$

where m is the smallest integer satisfies $m-1 < \alpha \le m$.

Caputo fractional derivative. The Caputo derivative reverses the order of operations: it first computes an integer-order derivative and then applies a fractional integral:

$$D^{\alpha} y(t) := J^{m-\alpha}_{\text{left}} \frac{d^m}{dt^m} y(t) = \frac{1}{\Gamma(m-\alpha)} \int_a^t \frac{\frac{d^m}{d\tau^m} y(\tau)\, d\tau}{(t-\tau)^{\alpha-m+1}}, \tag{29}$$

$$D^{\alpha}_{b-} y(t) := (-1)^m J^{m-\alpha}_{\text{right}} \frac{d^m}{dt^m} y(t) = \frac{(-1)^m}{\Gamma(m-\alpha)} \int_t^b \frac{\frac{d^m}{d\tau^m} y(\tau)\, d\tau}{(\tau-t)^{\alpha-m+1}}. \tag{30}$$

Compared with the RL formulation, the Caputo derivative has two important advantages: (i) it yields zero when applied to constant functions, consistent with classical derivatives; (ii) it allows the use of standard integer-order initial conditions in differential equations.

For these reasons, the Caputo formulation is more commonly adopted in practical modeling and is used throughout this work.

B.1.4 Memory Effect and Non-locality

A key feature of fractional derivatives is their intrinsic non-locality. As shown in (27) to (30), the derivative at time t depends on the entire history of the function over an interval (e.g., $[a, t]$ for left-sided operators). This stands in contrast to classical derivatives, which are purely local.

The kernel $(t-\tau)^{-\alpha}$ determines how past states contribute to the present, with the fractional order α controlling the decay rate of historical influence. This naturally introduces a form of long-range memory, which is particularly suitable for modeling systems with hereditary effects, long-term dependencies, or anomalous diffusion.

When the order α becomes an integer $n \in \mathbb{N}$, fractional operators recover their classical integer-order counterparts under appropriate regularity conditions [26]. This ensures that fractional calculus forms a consistent generalization of standard calculus.

For vector-valued functions, fractional operators act component-wise, analogous to classical differentiation and integration.

B.2 Traditional Numerical Methods for Solving FDEs

Considering the problem given by (3) where y_0 is the given initial condition, we define $h > 0$ as the discretization parameter (step size) and create a uniform grid over the interval $[0, T]$ given by $\{t_j = a + jh : j = 0, 1, \ldots, N\}$, where $N = T/h$ represents the number of time steps.

B.2.1 Fractional ABM predictor:

The numerical method is based on the Volterra integral representation of (3), where the D^α takes the Caputo Derivative:

$$y(t) = y_0 + \frac{1}{\Gamma(\alpha)} \int_0^t (t - \tau)^{\alpha-1} f(\tau, y(\tau)) ds \tag{31}$$

The fractional Adams-Bashforth predictor [26] is given by the following iterations:

$$y_{k+1} = y_0 + \frac{1}{\Gamma(\alpha)} \sum_{j=0}^{k} b_{j,k+1} f(t_j, y_j) \tag{32}$$

with weights

$$b_{j,k+1} = \frac{h^\alpha}{\alpha}((k + 1 - j)^\alpha - (k - j)^\alpha), \quad j = 0, 1, \ldots, k \tag{33}$$

- **short memory version:** Given a memory length M and we can truncate the convolution tail and get the short memory version:

$$y_{k+1} = y_0 + \frac{1}{\Gamma(\alpha)} \sum_{j=max\{0,k-M+1\}}^{k} b_{j,k+1} f(t_j, y_j) \tag{34}$$

B.3 Methods for Generating $\{\omega_i, \eta_i\}_{i=1}^M$

Here we provide two methods, actually the $\{\omega_i, \eta_i\}_{i=1}^M$ can be learnable.

B.3.1 Log-Trapezoid Quadrature for the Diffusive Integral

The first method uses log-trapezoidal quadrature to approximate the diffusive integral. Starting from the exact "diffusive" representation in (11), we let $\omega = e^x$, then the (11) becomes:

$$\frac{t^{\alpha-1}}{\Gamma(\alpha)} = \frac{\sin(\pi\alpha)}{\pi} \int_{-\infty}^{\infty} e^{-te^x} e^{(1-\alpha)x} dx \tag{35}$$

Then we truncate the frequency range $\omega \in [\omega_{\text{min}}, \omega_{\text{max}}]$ and apply the trapezoidal rule for numerical integration.

- **Choose frequency band:** Select a practical band that matches the time grid $[h, T]$:

$$\omega_{\text{min}} = \frac{1}{T}, \quad \omega_{\text{max}} = \frac{c}{h} \tag{36}$$

  where c is a safety factor (commonly chosen between 5 and 20, e.g., $c = 10$).

- **Nodes and Weights (Closed Form):** The nodes ω_i and weights η_i are determined as follows:

$$x_i = x_{\text{min}} + (i-1)\Delta x, \quad \omega_i = e^{x_i}, \quad \eta_i = \frac{\sin(\pi\alpha)}{\pi} \Delta x e^{(1-\alpha)x_i} = \frac{\sin(\pi\alpha)}{\pi} \Delta x \omega_i^{(1-\alpha)} \tag{37}$$

  with the trapezoidal half-weights:

$$\eta_1 \leftarrow \frac{1}{2}\eta_1, \quad \eta_M \leftarrow \frac{1}{2}\eta_M \tag{38}$$

B.3.2 Log-Spaced Least Squares (Nonnegative)

The second method uses log-spaced least squares to compute the weights η_i. This method is particularly effective for improving accuracy, especially for the same M.

- **Choose ω_i:** We choose the same frequency band as the first method, but now the nodes ω_i are chosen on a logarithmic scale:

$$\omega_i = \omega_{\text{min}} \left(\frac{\omega_{\text{max}}}{\omega_{\text{min}}} \right)^{\frac{i-1}{M-1}}, \quad i = 1, \ldots, M \tag{39}$$

- **Fit η on Sampled Times:** We pick K sample times, and log-spaced in the interval $[h, T]$:

$$t_n = h \left(\frac{T}{h} \right)^{\frac{n-1}{K-1}}, \quad n = 1, \ldots, K \tag{40}$$

  Then build the matrix:

$$A_{ni} = e^{-\omega_i t_n}, \quad b_n = \frac{t_n^{\alpha-1}}{\Gamma(\alpha)} \tag{41}$$

  Finally solve for η via non-negative least squares (NNLS):

$$\min_{\eta \geq 0} \|A\eta - b\|_2^2 \tag{42}$$

  Furthermore, we can also use the ridge regression:

$$\eta = (A^T A + \lambda I)^{-1} A^T b, \quad \text{where} \quad \lambda > 0 \tag{43}$$

C Derivation of Standard and Fractional SSM Solutions

C.1 Standard SSM ($\alpha = 1$)

Consider the standard scalar SSM:

$$\frac{dh(t)}{dt} = -\lambda h(t) + x_c, \quad h(0) = h_0 \tag{44}$$

with constant input x_c.

The homogeneous equation is

$$\frac{dh_h(t)}{dt} = -\lambda h_h(t) \quad \Rightarrow \quad h_h(t) = Ce^{-\lambda t}. \tag{45}$$

For constant input, assume $h_p = K$. Then

$$0 = -\lambda K + x_c \quad \Rightarrow \quad K = \frac{x_c}{\lambda}. \tag{46}$$

So the general solution is:

$$h(t) = h_h(t) + h_p = Ce^{-\lambda t} + \frac{x_c}{\lambda}. \tag{47}$$

We apply initial condition $h(0) = h_0$ gives $C = h_0 - \frac{x_c}{\lambda}$. Therefore the solution is

$$h(t) = \frac{x_c}{\lambda} + \left(h_0 - \frac{x_c}{\lambda} \right) e^{-\lambda t}. \tag{48}$$

C.2 Fractional SSM ($0 < \alpha < 1$)

Consider the fractional case:

$$D^\alpha h(t) = -\lambda h(t) + x_c, \quad h(0) = h_0 \tag{49}$$

Using the Caputo derivative property:

$$\mathcal{L}\{D^\alpha h(t)\} = s^\alpha H(s) - s^{\alpha-1} h_0 \tag{50}$$

we have

$$s^{\alpha}H(s) - s^{\alpha-1}h_0 = -\lambda H(s) + \frac{x_c}{s} \tag{51}$$

Then we solve for $H(s)$

$$(s^{\alpha}+\lambda)H(s) = s^{\alpha-1}h_0 + \frac{x_c}{s} \quad \Rightarrow \quad H(s) = \frac{s^{\alpha-1}h_0}{s^{\alpha}+\lambda} + \frac{x_c}{s(s^{\alpha}+\lambda)}. \tag{52}$$

Using the Mittag-Leffler function property:

$$\mathcal{L}^{-1}\left\{\frac{s^{\alpha-1}}{s^{\alpha}+\lambda}\right\} = E_{\alpha}(-\lambda t^{\alpha}), \tag{53}$$

and

$$\frac{1}{s(s^{\alpha}+\lambda)} = \frac{1}{\lambda}\left(\frac{1}{s} - \frac{s^{\alpha-1}}{s^{\alpha}+\lambda}\right), \tag{54}$$

we obtain

$$h(t) = h_0 E_{\alpha}(-\lambda t^{\alpha}) + \frac{x_c}{\lambda}\left[1 - E_{\alpha}(-\lambda t^{\alpha})\right] = \frac{x_c}{\lambda} + \left(h_0 - \frac{x_c}{\lambda}\right) E_{\alpha}(-\lambda t^{\alpha}). \tag{55}$$

D Experimental Details

The specific parameter settings for all experiments are summarized in Table 4. These settings cover six tasks from the Long Range Arena (LRA) benchmark, as well as experiments on sMNIST, psMNIST, WikiText-103, Speech Commands and DVS128-Gesture.

The LongSpike model utilizes the piecewise quadratic surrogate spiking function. We chose the piecewise quadratic surrogate because it balances stability, efficiency, and sparsity in SNN training: Its gradient is bounded and non-zero only in the local neighborhood near the threshold, which ensures stable backpropagation while avoiding noise updates from distant threshold states. It is computationally efficient and has a symmetric shape, which is also compatible with learnable thresholds. The gradient of this function is defined as:

$$g'(x) = \begin{cases} 0, & \text{if } |x| > \frac{1}{\gamma} \\ -\gamma^2|x| + \gamma, & \text{if } |x| \le \frac{1}{\gamma} \end{cases} \tag{56}$$

The primitive function is defined as:

$$g(x) = \begin{cases} 0, & \text{if } x < -\frac{1}{\gamma} \\ -\frac{1}{2}\gamma^2|x|x + \gamma x + \frac{1}{2}, & \text{if } |x| \le \frac{1}{\gamma} \\ 1, & \text{if } x > \frac{1}{\gamma} \end{cases} \tag{57}$$

In our experiments, the surrogate-gradient parameter γ is set to 1.

Experiments are conducted on NVIDIA GPUs (e.g., NVIDIA V100, A100, A800, H800,and RTX 4090), selected according to the memory requirements of different tasks. Unless otherwise specified, all experiments are run on a single GPU.

E Computational Cost and Energy Analysis

E.1 Computational Complexity Analysis

We analyze the computational complexity of LongSpike relative to SpikingSSM, focusing on the common setting with two fractional terms ($M = 2$).

Time Complexity. SpikingSSM has a forward-pass complexity of $O(HNL + BHL\log L)$, which is dominated by FFT-based convolution. LongSpike approximates the fractional kernel using M exponential terms, increasing the kernel construction cost from $O(HNL)$ to $O(MHNL)$, while leaving the FFT-based convolution unchanged. Furthermore, on GPU hardware, we can compute the effective kernel K_{eff} in parallel via aggregation prior to convolution. Overall, the additional

Table 4: The hyperparameters used in the experiments for various benchmark tasks. H denotes the model dimension, N denotes the state dimension, LR denotes learning rate, WD denotes weight decay, and BS denotes the batch size. BN and LN refer to Batch Normalization and Layer Normalization.

Dataset	Depth	H	N	Norm	pNorm	Dropout	LR	BS	Epochs	WD	$(\Delta_{\min}, \Delta_{\max})$	Hardware
ListOps	8	128	64	BN	False	0.05	0.011	50	40	0.03	(0.001, 0.1)	1×V100
Text	6	256	64	BN	True	0	0.002	16	32	0.01	(0.001, 0.1)	1×A100
Retrieval	6	256	64	BN	True	0	0.01	64	20	0.01	(0.001, 0.1)	1×A100
Image	6	512	64	LN	False	0.1	0.01	50	200	0.03	(0.001, 0.1)	1×V100
Pathfinder	6	256	64	BN	True	0.05	0.004	64	200	0.005	(0.001, 0.1)	1×V100
Path-X	6	256	64	BN	True	0	0.0011	16	50	0.01	(0.0001, 0.01)	1×A800
WT-103	16	1024	64	LN	True	0.1	0.0005	1	200	0.01	(0.001,0.1)	8×H800
psMNIST	4	256	64	LN	False	0.1	0.01	50	100	0.01	(0.001, 0.1)	1×V100
sMNIST	2	400	64	LN	False	0.1	0.01	50	100	0.01	(0.001, 0.1)	1×V100
DVS128	6	512	16	LN	False	0.1	0.005	32	200	0.05	(0.001, 0.1)	1×RTX 4090
SC	6	128	32	BN	True	0	0.01	16	40	0.01	(0.001, 0.1)	1×RTX 4090

cost introduces only a constant-factor overhead and preserves the same asymptotic dependence on sequence length.

Space Complexity. LongSpike supports both vectorized and loop-based implementations. The vectorized implementation requires $O(MHNL)$ memory, which remains acceptable for small M (e.g., $M = 2$), while the loop-based implementation reduces memory usage to $O(HNL)$, matching that of SpikingSSM.

In practice, we observe that LongSpike ($M = 2$) incurs no additional runtime overhead under our optimized implementation. As shown in Table 5, its per-epoch training time is comparable to, and slightly lower than, that of SpikingSSM, while GPU memory usage is also reduced.

Table 5: Training time and GPU memory usage on the Speech Commands dataset.

Model	Training Time (Per-epoch)	GPU Memory (MiB)
SpikingSSM	∼26.6 min	14429
LongSpike ($M = 2$)	∼26.0 min	12925
Relative Change	↓∼2.3%	↓∼10.4%

The reduced memory footprint can be attributed to implementation-level optimizations, including CUDA-based kernel fusion and memory-efficient aggregation, which avoid materializing unnecessary intermediate states. These optimizations also help offset the additional constant-factor cost introduced by the fractional kernel construction. Overall, the results indicate that the proposed fractional formulation can be implemented with comparable runtime and lower memory consumption relative to SpikingSSM.

E.2 Energy Analysis via Spiking Activity

Spiking neural networks (SNNs) are widely regarded as energy-efficient due to their sparse and binary activation patterns. Unlike artificial neural networks (ANNs), where the dominant operation is multiply-and-accumulate (MAC), SNNs primarily rely on synaptic accumulation (AC). In particular, binary spike activations allow multiplication with weights to be simplified to addition operations on certain neuromorphic hardware, leading to significantly lower energy consumption[96].

Although the hardware specifics and neuron dynamics are not considered here, a theoretical analysis can provide an estimate of SNN efficiency. Following prior work[96, 98], we assume the energy cost of a MAC operation to be $E_{\text{MAC}} = 4.6$ pJ and that of an AC operation to be $E_{\text{AC}} = 0.9$ pJ[97].

We define the *spiking rate* as the ratio of spike events to total neuron activations over time, and use the average spiking rate across all neurons as a proxy for network activity. Since the number of AC operations in SNNs is proportional to the spiking rate, this metric provides an estimate of effective computation and energy consumption.

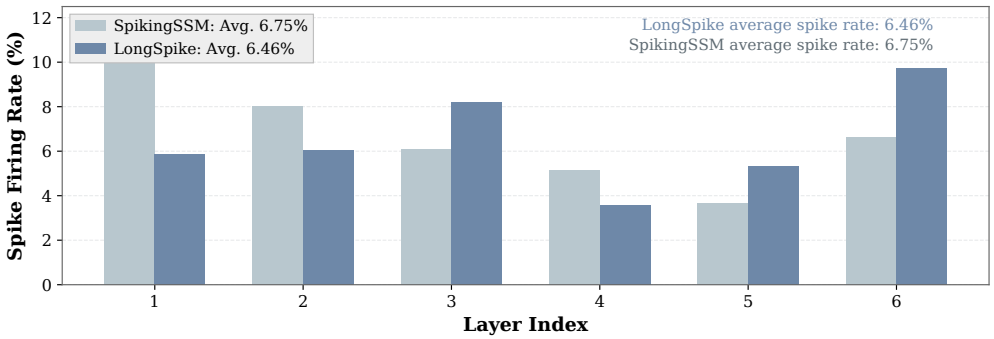


Figure 4: Layer-wise spiking rates on the TEXT task of the LRA benchmark. LongSpike maintains a comparable or lower average spiking rate (6.46%) than SpikingSSM (6.75%), indicating that performance improvements do not rely on increased neural activity.

In this analysis, we focus on the TEXT task of the LRA benchmark, where the model consists of 6 layers with a feature-mixing projection from $d = 512$ to $d = 1024$. Figure 4 shows the layer-wise spiking rates. To quantify energy efficiency, we focus on the feature-mixing layers, which dominate computation, and estimate the number of operations and corresponding energy consumption based on these spiking statistics.

Table 6: Energy-efficiency comparison on the TEXT task of the LRA benchmark.

Model	Op Type	Ops (G)	Energy (mJ)	Accuracy (%)
S4 (ANN)	MAC	3.22	14.82	86.97
SpikingSSM	AC	0.217	0.196	80.41
LongSpike	AC	0.208	0.187	**88.11**

As shown in Table 6, both spiking models significantly reduce computation and energy consumption compared to the ANN baseline. Notably, LongSpike achieves higher accuracy while slightly reducing effective computation and energy compared to SpikingSSM. This indicates that the performance gains stem from improved long-range modeling rather than increased neural activity.

Overall, LongSpike preserves the energy-efficient characteristics of SNNs while enhancing sequence modeling capability. This analysis serves as a theoretical proxy for energy efficiency, since the actual energy consumption depends on hardware platforms and implementation details.

F Extended Experimental Results

In this section, we present additional results that were not included in the main content.

F.1 Sequential MNIST.

The MNIST dataset [84] contains 70,000 grayscale images of handwritten digits (0–9), with 60,000 for training and 10,000 for testing, each of size 28×28 pixels. The Sequential MNIST (sMNIST) [85] flattens these images into sequences of 784 elements, while the Permuted Sequential MNIST (psMNIST) [85] applies a fixed pixel permutation, disrupting spatial correlations. These variants test a model's ability to learn sequence dependencies rather than relying on spatial patterns.

We evaluate LongSpike on the sMNIST and psMNIST datasets to examine its ability to model moderately long-range dependencies. As shown in Table 7, compared to state-of-the-art non-spiking (e.g., S4 [15]) and spiking models (e.g., SpikingSSM [19]), LongSpike achieves 99.62% on sMNIST and 98.51% on psMNIST, showing competitive performance on moderately long-range sequence tasks.

Table 7: Performance comparison of LongSpike with other works on sMNIST and psMNIST datasets.

Model	SNN	sMNIST	psMNIST
LMUformer	No	98.95	98.55
S4	No	**99.63**	**98.70**
SpikingLMUformer	Yes	97.43	97.92
Binary-S4D	Yes	99.4	-
P-SpikeSSM	Yes	99.43	98.4
SpikingSSM	Yes	99.6	98.4
LongSpike	Yes	**99.62**	**98.51**

F.2 DVS128 Gesture.

The DVS128 Gesture dataset is a neuromorphic benchmark consisting of 11 hand gesture classes collected from 29 subjects under varying illumination conditions. The images are based on event-driven recordings, and we refer to previous work [94] to preprocess the data format from event streams to frames. The results are reported in Table 8.

Table 8: Performance comparison of LongSpike and other works on the DVS128 Gesture dataset.

Model	#Params	Accuracy(%)
SpikingSSM [19]	3.0M	97.1
LongSpike	3.0M	97.4

F.3 Effect of Fractional Dynamics without Spiking

To further investigate the contribution of fractional dynamics independently of the spiking mechanism, we evaluate a non-spiking variant based on a vanilla fractional state space model (f-SSM).

Specifically, we replace the integer-order state space model (S4) with its fractional counterpart while removing all spiking components. This allows us to directly assess the impact of fractional-order dynamics on long-sequence modeling performance.

As shown in Table 9, the vanilla f-SSM consistently outperforms the standard S4 model across all LRA tasks. These results suggest that the performance gains mainly stem from the improved long-memory modeling capability introduced by fractional dynamics, rather than the spiking mechanism itself.

Table 9: Comparison between integer-order SSM (S4) and fractional SSM (f-SSM) on LRA benchmark.

MODEL	SNN	LISTOPS	TEXT	RETRIEVAL	IMAGE	PATHFINDER	PATH-X	AVG
S4	No	58.35	76.02	87.09	87.26	86.05	88.10	80.48
f-SSM	No	63.20	90.82	91.96	90.28	95.87	96.21	88.06

G Impact Statement

This research contributes to the field of sequence modeling by introducing f-SSM into SNNs. The positive impacts of this work include the development of more robust and energy-efficient models for speech recognition, time-series forecasting, and physical system modeling. By improving the memory capacity of SNNs without incurring quadratic computational costs, we enable the deployment of sophisticated temporal models in real-time safety-critical systems. We acknowledge that improvements in sequence modeling can be applied to dual-use technologies, such as deepfake generation or automated surveillance. However, our work primarily targets the efficiency bottleneck of current AI systems, aiming to reduce the computational resources required for state-of-the-art performance.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The paper's contributions and scope are detailed in the abstract and introduction Section 1.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We have already discussed limitations in Section 7.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

Justification: The theorems proposed in this paper are supported by detailed proofs.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: We provide sufficient details on the model architecture, datasets, and training procedures to enable reproducibility of the reported results.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: We have provided the source code in the supplementary material.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: We have included the implementation details in both the main text and the Appendix D.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: We report mean and standard deviation values in our main experiments.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.

- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We report the computing resources in Appendix D.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer:[Yes]

   Justification: We conform, in every respect, with the NeurIPS Code of Ethics.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: We have discussed broader impacts in Appendix G.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The paper poses no such risks.

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We cite the original paper that produced the code package or dataset.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A]

    Justification: The paper does not release new assets.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [N/A].

Justification: LLMs are not used as an important, original, or non-standard component of the core method or experiments.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.