SPEAK: Spiking Neurons as an Entropy-Aware Tokenizer for Large Language Models

Ming Chen, Wenyao Li, Chao Liang, Shi Gu, Peng Lin, De Ma, Huajin Tang, Qian Zheng✉, Gang Pan✉

The State Key Lab of Brain-Machine Intelligence, Zhejiang University
College of Computer Science and Technology, Zhejiang University
{mchen88, qianzheng, gpan}@zju.edu.cn

Abstract

Tokenizers play a critical role in large language model studies. Despite recent advances, existing tokenizers fail to explicitly leverage historical tokenization results when making subsequent token decisions, nor do they selectively utilize such history based on contextual relevance. We propose **SPEAK**, a gradient-based tokenizer that integrates spiking neurons to explicitly leverage historical tokenization results. Furthermore, we introduce an entropy-aware reset mechanism that selectively leverages history based on contextual relevance, which is determined by token-level entropy. High-entropy tokens are treated as contextual boundaries, whereas low-entropy tokens between consecutive such boundaries exhibit strong contextual relevance. Accordingly, we induce hard reset at high-entropy tokens to discard irrelevant historical tokenization results, and soft reset at low-entropy tokens to preserve and leverage relevant history. Experiments on 2 language models and 5 datasets spanning 16 languages demonstrate superior cross-lingual adaptability, with competitive performance and efficiency. Our code is publicly available at https://github.com/zju-bmi-lab/SPEAK.

1 Introduction

Large language models (LLMs) have demonstrated remarkable effectiveness across NLP tasks (Brown et al., 2020; Yuan et al., 2025; Fang et al., 2025). Recent advances in LLM capabilities are largely attributed to deeper theoretical insights and improved interpretability (Sivaprasad et al., 2025; Luo and Specia, 2024; Yu et al., 2025). However, current studies adopt traditional tokenizers (e.g. BPE and Unigram) by default, with little scrutiny of their effects on model behavior. Emerging evidence suggests that tokenization can significantly affect downstream performance, efficiency, and reasoning quality (Ali et al., 2024; Wang et al., 2024).

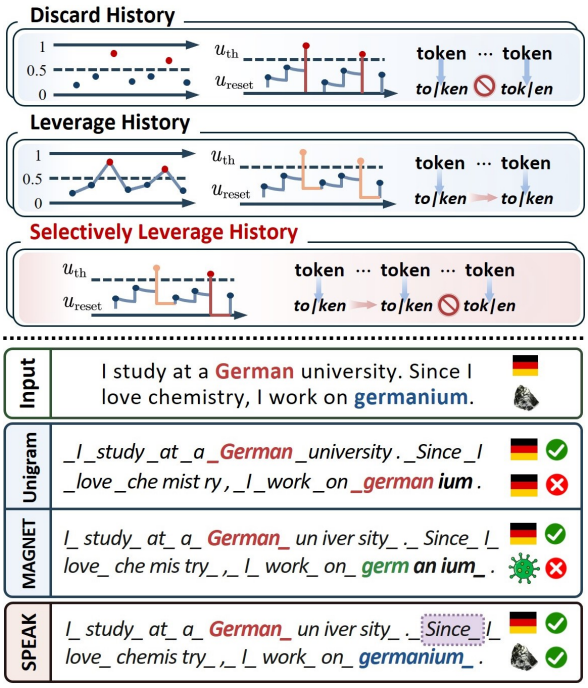

Figure 1: **Top: Theoretical comparison.** Tokenizers that discard history may produce different tokens for the same word, whereas those that leverage history yield identical tokens. In contrast, our model selectively leverages historical tokenization results, adapting history usage. **Bottom: Empirical comparison.** Only we correctly tokenize the chemical term "germanium" without being influenced by the historical token "German", via selectively leveraging history only after the high-entropy token "Since". In contrast, other methods produce misunderstanding tokens "german" or "germ".

Recently, many works focus on designing state-of-the-art (SOTA) tokenizers to improve downstream performance (Zheng et al., 2024), inference efficiency (Feher et al., 2025), adaptability (Geng et al., 2025), and multilingual fairness (Ahia et al., 2024). Many of these methods build upon traditional tokenizers and iteratively refine themselves using handcrafted or heuristic rules, resulting in rule-based tokenizers (RToks) that are easy to im-

✉Corresponding author

Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 9943–9960
July 2-7, 2026

plement and update (Oh et al., 2025). Despite their advances, such rigid rule-based designs limit adaptability and can degrade performance under diverse linguistic conditions (Feher et al., 2025; Minixhofer et al., 2024). In contrast, gradient-based tokenizers (GToks) have been proposed to optimize token boundaries end-to-end via gradient descent, enabling flexible adaptation across tasks, languages, and scripts, and often achieving strong empirical performance (Ahia et al., 2024; Nawrot et al., 2023; Thawani et al., 2023).

Despite these advances, fundamental challenges remain. **1) Historical tokenization results should be explicitly leveraged during token boundary decisions.** Current GToks predict token boundaries independently, without conditioning on historical token boundary decisions, often leading to suboptimal boundary placement (Godey et al., 2022; Thawani et al., 2023). Some methods attempt to incorporate history via indirect temporal modeling over character embeddings using transformers; however, they do not explicitly model historical tokenization results and introduce additional computational overhead (Ahia et al., 2024; Nawrot et al., 2023). Hence, existing methods fail to explicitly leverage historical tokenization results for boundary decision making. **2) Historical tokenization results should be selectively leveraged based on contextual relevance.** Natural language exhibits complex linguistic and semantic continuity (Futrell and Hahn, 2025). Accordingly, historical tokenization results are not uniformly informative for current token boundary decisions: some contribute meaningfully due to contextual relevance, while others introduce noise (Tikochinski et al., 2025). However, existing temporal models (e.g., recurrent models and spiking neurons) typically leverage history indiscriminately, without assessing contextual relevance. Hence, we need mechanisms that first identify contextual relevance, and then selectively leverage relevant while discarding irrelevant historical tokenization results based on it.

To address these challenges, we propose **SPEAK**, where **SP**iking neurons act as an **E**ntropy-**A**ware to**K**enizer. Specifically, we build a GTok with spiking neurons integrated to explicitly leverage historical tokenization results, thereby addressing challenge 1). Spiking neurons accumulate input features into membrane potentials and emit spikes upon threshold exceeding, with each spike corresponding to a token boundary. After each spike, the reset mechanism is activated with two typical modes: hard reset completely clears history, while soft reset partially preserves it. This inherent mechanism provides a natural and discrete way for history truncation, which recurrent models lack. To resolve challenge 2), we leverage token-level generation entropy to determine boundaries that define ranges of contextual relevance, referred to as "contextual boundaries" and "contextual ranges", respectively. We observe that high-entropy tokens often act as "forking" points that steer models toward diverse reasoning paths, indicating linguistic isolation, whereas low-entropy tokens tend to follow the reasoning path, signifying linguistic continuity (Wang et al., 2025). Hence, high-entropy tokens naturally serve as contextual boundaries, while low-entropy tokens between consecutive such boundaries, i.e., within a contextual range, exhibit strong contextual relevance. Accordingly, we induce hard reset at these contextual boundaries to discard irrelevant historical tokenization results, and soft reset within the contextual ranges to preserve and leverage history. This entropy-aware reset mechanism enables selectively leveraging historical tokenization results based on contextual relevance.

Contributions. 1) We propose **SPEAK**, a GTok that integrates spiking neurons to explicitly leverage historical tokenization results. **2)** We develop an **entropy-aware reset mechanism** to selectively leverage historical tokenization results based on contextual relevance, which is determined by token-level entropy. **3)** We evaluate our model on 2 language models and 5 datasets spanning 16 languages, benchmarking against SOTA GToks and RToks. Experimental results demonstrate superior cross-lingual adaptability with competitive performance and efficiency.

2 Related Works

2.1 Subword, Rule-Based and Gradient-Based Tokenizers

Subword tokenizers are widely used in LLMs (Sennrich et al., 2016; Kudo, 2018; Wang et al., 2021; Su, 2023; Kudo and Richardson, 2018), but they lack direct interaction with LLM and cannot enhance LLM capabilities. Hence, advanced tokenizers have emerged. Among them, RToks rely on handcrafted rules, easy to implement and update. But their rigidity incurs issues, such as trading performance for efficiency (Feher et al., 2025), or requiring extensive prior knowledge (Oh et al., 2025). In contrast, GToks use learnable neural networks

to predict token boundaries via end-to-end gradient optimization, consistently achieving strong empirical performance (Nawrot et al., 2023; Islam et al., 2022; Tay et al., 2022; Thawani et al., 2023). We build upon the GTok paradigm to inherit its adaptability across diverse scenarios.

2.2 Generation Entropy

Generation entropy is widely used in mainstream AI research, such as LLM reasoning (Farquhar et al., 2024; Qiu et al., 2025) and image generation (Yoon et al., 2024). The entropy quantifies the uncertainty of the predictive distribution at each position. High-entropy tokens typically indicate greater model uncertainty or semantic complexity, and are often seen as undesirable. Consequently, prior work frequently minimizes entropy to improve performance (Agarwal et al., 2025; Ma et al., 2025). Conversely, Wang et al. (2025) exploits high-entropy tokens to enhance LLM reasoning abilities. They propose that high-entropy tokens act as critical "forking" points that steer models toward diverse reasoning paths, whereas low-entropy tokens follow established paths. Inspired by this, we propose an innovative entropy-aware reset mechanism for spiking neurons.

2.3 Tokenizer Optimization

Optimizing tokenizers introduces new tokens whose embeddings need to be learned, thus requiring training the language model from scratch (Oh et al., 2025; Geng et al., 2025; Zheng et al., 2024). To address this issue, Minixhofer et al. (2024) propose a hypernetwork that enables tokenizer transfer. Hence, RToks can flexibly modify tokenization schemes, as the hypernetwork has already prepared embeddings for new tokens (Feher et al., 2025). However, GToks face an additional challenge. The tokenization process is non-differentiable, preventing the gradient flow from optimizing GToks. Prior works address this issue using differentiable approximations such as soft grouping or Gumbel-sigmoid (Maddison et al., 2017; Jang et al., 2017). But, integrating such methods with LLMs requires replacing the standard embedding pipeline and training the entire model from scratch. Hence, many current approaches are only implemented on a lightweight model, limiting applicability to mainstream LLMs (Ahia et al., 2024; Nawrot et al., 2023). By contrast, we propose momentum-based proxy supervision, enabling the usage of hypernetworks and integration with mainstream LLMs.

2.4 Spiking Neuron

Spiking neurons are widely used to capture temporal dynamics in time-varying data (Zheng et al., 2026). They encode information as discrete spikes, enabling event-driven, low-latency, and temporally heterogeneous computation (Hu et al., 2023a,b; Yan et al., 2025), and are particularly suited to neuromorphic computing architectures (Xiao et al., 2025; Ma et al., 2024; Cao et al., 2025). A unique property of spiking neurons is the reset mechanism, commonly including hard reset, which discards all historical information, and soft reset, which preserves partial state. Natural language exhibits complex contextual patterns, alternating between semantic isolation and continuity, which demands fine-grained temporal modeling (Wang et al., 2025). In this regard, spiking neurons provide a better framework than recurrent models (details in Appendix A.1), making them well-suited for capturing such patterns.

3 Methodology

3.1 Spiking Neuron-Based GTok

Spiking neuron. The Leaky-Integrate-and-Fire (LIF) model (Burkitt, 2006; Hunsberger and Eliasmith, 2015) is a widely used neuron model. The discrete LIF equations are given by:

$$
\begin{aligned}
u_t &= \gamma v_{t-1} + \delta I\,, \\
s_t &= \mathbb{1}(u_t \geqslant u_{\text{th}})\,, \\
v_t &= \begin{cases} u_t - u_{\text{th}} \cdot s_t & \text{soft reset}, \\ u_t(1-s_t) + u_{\text{reset}} \cdot s_t & \text{hard reset}, \\ u_t & \text{no reset}, \end{cases}
\end{aligned} \tag{1}
$$

where $\mathbb{1}(\cdot)$ is the indicator function. These equations formulate that the spiking neuron sequentially receives and accumulates the input current I_t into its membrane potential u_t while decaying at a constant rate γ. When reaching the threshold u_{th}, a spike is emitted (i.e., $s_t = 1$) and the reset mechanism is activated. There are typically two reset modes: in soft reset, the membrane potential is reduced by the threshold value; in hard reset, it is directly set to u_{reset}. Another, far less common, is to never reset the membrane potential. Consequently, the neuron emits spikes sequentially while either preserving partial historical information or erasing it entirely. This inherent mechanism enables simultaneous control over the retention and isolation of historical features, differentiating from recurrent networks.

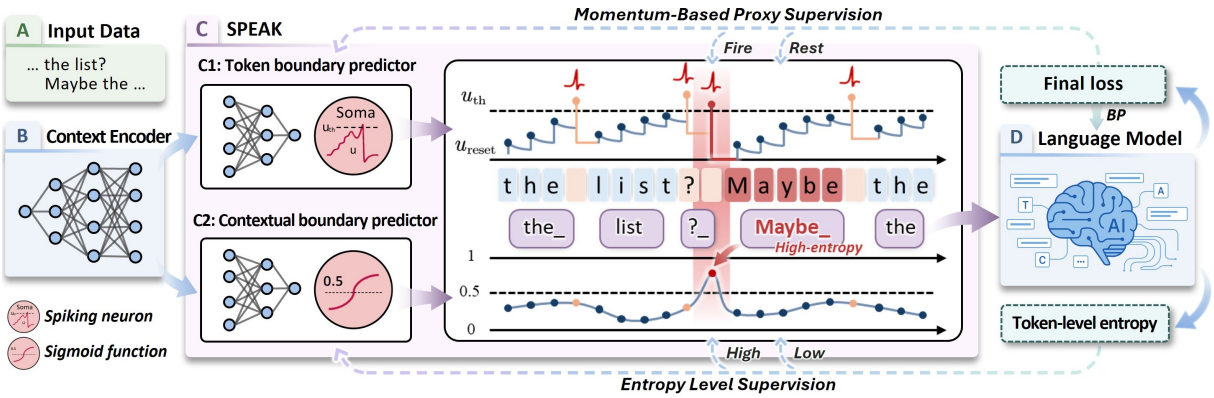

Figure 2: **A) Input data:** A character sequence is processed sequentially. **B) Context encoder:** Extract character embeddings. **C) SPEAK:** Character embeddings are used to predict token boundary logits and contextual boundary logits. The former generate spikes to determine token boundaries; the latter determine contextual boundaries to control reset modes. Specifically, hard reset is induced at the starting boundaries of high-entropy tokens to discard history, while soft reset is applied at boundaries of low-entropy tokens to leverage history. **D) Optimization:** After tokenization, the objective loss and token-level entropy are computed. The objective loss optimizes the language model via backpropagation and the token boundary predictor through proxy supervision. Token-level entropy supervises the contextual boundary predictor.

Token boundary prediction. Now, we integrate this spiking neuron into a GTok, denoted as $\mathcal{T}$, to predict token boundaries. The architecture consists of a context encoder to extract character embeddings, a linear block and a LIF neuron to produce token boundaries. It is formulated as:

$$\mathcal{T} = \text{LIF} \circ \text{Linear}_1 \circ \text{ContextEncoder}(\cdot)\,. \quad (2)$$

Here, ContextEncoder represents a neural network that extracts character embeddings from input text via fully-connected layers with non-linear activations (e.g., tanh). Linear_1 is a fully-connected layer projecting ContextEncoder outputs to the LIF spiking neuron input dimension for token boundary prediction.

Given an input character sequence $\mathbf{c} = (c_1, \ldots, c_L)$, the tokenizer sequentially outputs the membrane potential and spikes for each position:

$$\mathcal{T} : \bigcup_{i=1}^{L} \mathbb{R}^i \to (\mathbb{R}, \mathbb{R}), \quad (u_i, s_i) = \mathcal{T}(c_{1:i}), \quad (3)$$

which we denote as soft and hard boundaries:

$$\text{soft: } u_i \in \mathbb{R}\,, \quad \text{hard: } s_i \in \{0, 1\}\,. \quad (4)$$

Here, soft boundaries $\mathbf{u} = (u_1, \ldots, u_L)$ represent continuous boundary salience at each character position, while hard boundaries $\mathbf{s} = (s_1, \ldots, s_L)$ are explicit token boundaries. Our GTok explicitly leverages historical tokenization results in a causal manner when making token boundary decisions.

Tokenization and objective loss. Finally, we segment the character sequence $\mathbf{c}$ into a token sequence $\mathbf{x} = (x_1, \ldots, x_T)$ via $\mathbf{s}$, denoted as:

$$\mathbf{x} = \text{Tokenize}(\mathbf{c}, \mathbf{s})\,. \quad (5)$$

Then we feed the tokens into a language model $\mathcal{F}$ to compute the objective loss:

$$\mathcal{L}_{\text{LM}} = f(\mathcal{F}, \mathbf{x})\,, \quad (6)$$

where f denotes the task-specific loss function, such as cross-entropy for autoregressive language modeling.

3.2 Entropy-Aware Reset Mechanism

Motivation. By default, the spiking neuron uniformly adopts hard reset after each spike, discarding history for every token, deemed inferior by Nawrot et al. (2023). Sometimes, it uniformly adopts soft reset after each spike, preserving partial history after each token, functionally analogous to gated recurrent networks. Rarely, the neuron forgoes reset, accumulates all past information like a vanilla recurrent unit. The latter two settings leverage all history overlooking contextual relevance, which is theoretically suboptimal as proved by us in Appendix E.

Entropy-aware reset. Therefore, we propose an innovative reset mechanism that adaptively resets the spiking neuron based on token-level generation entropy. Specifically, we induce hard reset at the starting boundary of each high-entropy token to enforce linguistic isolation. For low-entropy tokens,

we apply a scaled soft reset at their boundaries, which preserves partial historical tokenization results to maintain contextual linguistic continuity. The scaling factor controls the proportion of history propagated to future timesteps.

As illustrated in Fig.3, we opt for soft reset rather than no reset for low-entropy tokens to avoid imposing spurious constraints upon spiking behaviors. Under no reset, the membrane potential accumulates unbounded, forcing subsequent input currents to be negative to suppress spiking—a condition that may conflict with legitimate future spiking targets (e.g., at position c_3), thereby degrading token boundary prediction.

In summary, we treat high-entropy tokens as contextual boundaries at which historical tokenization results are discarded, and regard low-entropy tokens between two such boundaries as forming a contextual range, within which historical tokenization results are leveraged. Consequently, our entropy-aware reset enables simultaneous control over the retention and isolation of historical information.

Formulation. Since token-level entropy is only available after tokenization, we initially introduce a contextual boundary predictor $\mathcal{C}$ that is trained to identify positions as contextual boundaries or not. It is formulated as:

$$\mathcal{C} = \text{Linear}_2 \circ \text{ContextEncoder}(\cdot)\,, \tag{7}$$

which shares the same ContextEncoder with $\mathcal{T}$. Linear_2 is a fully-connected layer mapping ContextEncoder outputs to a scalar for contextual boundary prediction, where the distinct subscript differentiates it from Linear_1.

Given the input $\mathbf{c}$, $\mathcal{C}$ outputs boundary logits $\mathbf{e} = (e_1, \ldots, e_L)$ for each position:

$$\mathcal{C} : \mathbb{R}^L \to \mathbb{R}^L, \quad e_i \in \mathbb{R} = \mathcal{C}(c_i)\,. \tag{8}$$

These logits further calculates binary masks indicating where hard reset should occur:

$$m_i = \mathbb{1}\big(\sigma(e_i) > 0.5\big), \quad i = 1, \ldots, L\,. \tag{9}$$

These masks control the reset modes of the spiking neuron at each position:

$$\begin{aligned} v_t = {} & m_t \cdot \big(u_t(1 - s_t) + u_{\text{reset}} \cdot s_t\big) \\ & + (1 - m_t) \cdot \big(u_t - \theta \cdot u_{\text{th}} \cdot s_t\big)\,, \end{aligned} \tag{10}$$

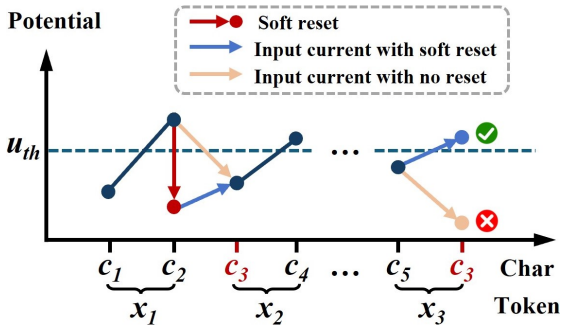

Figure 3: Reasons for soft reset over no reset. Under the soft reset setting, for a spiking target position (e.g., c_2), only its input current needs to be constrained to be sufficiently positive to drive the membrane potential above threshold u_{th}. However, under the no reset setting, an extra constraint is imposed: the input current at the next position (i.e., c_3) must be sufficiently negative to keep the membrane potential below threshold. This can interfere with future spiking guidance, particularly when a later occurrence of c_3 is itself a spiking target.

which replaces the original reset equation in Eq. (1). This equation integrates three conditions:

$$v_t = \begin{cases} u_t & \text{no spike,} \\ u_{\text{reset}} & \text{spike \& c. boundary,} \\ u_t - \theta \cdot u_{\text{th}} & \text{spike \& } \neg \text{ c. boundary,} \end{cases} \tag{11}$$

where c. boundary indicates the presence of a contextual boundary ($m_t = 1$), and $\neg$ denotes its absence ($m_t = 0$). Here, we introduce a hyperparameter $\theta \in (0, 1)$ to scale the soft reset magnitude, allowing finer control over the retention of historical information within the contextual range. Finally, we only need to supervise $\mathcal{C}$ to estimate correct contextual boundaries at high-entropy token boundaries.

Entropy level supervision. After tokenization based on $\mathcal{C}$ and $\mathcal{T}$, we compute token-level entropy. Following Wang et al. (2025), we identify the top 20% of tokens ranked by their entropy values as high-entropy tokens, and denote the set of their starting boundaries as $\mathcal{H}$ (details in Appendix A.2). We supervise $\mathcal{C}$ to predict contextual boundaries at $\mathcal{H}$ using a binary cross-entropy loss:

$$\mathcal{L}_{\text{reset}} = \text{BCEWithLogits}(\mathbf{e}, \hat{\mathbf{e}})\,, \tag{12}$$

where $\hat{\mathbf{e}} = (\hat{e}_1, \ldots, \hat{e}_L)$ and $\hat{e}_i = \mathbb{1}(i \in \mathcal{H})$.

3.3 Optimization

Momentum-based proxy supervision. At epoch N, we generate a proxy supervision signal $\hat{\mathbf{s}}_N = (\hat{s}_1^N, \ldots, \hat{s}_L^N) \in \{0, 1\}^L$ to guide the spiking behavior of the tokenizer $\mathcal{T}$. This proxy is iteratively

refined across epochs to minimize $\mathcal{L}_{\text{LM}}$, forming a momentum-based update sequence:

$$\hat{\mathbf{s}}_{N+1} = a_N \cdot \mathbf{s}_{N+1} + (1 - a_N) \cdot \hat{\mathbf{s}}_N \,, \tag{13}$$

where $a_N \in \{0, 1\}$ is a binary acceptance variable drawn via Metropolis-Hastings (MH) sampling (Metropolis et al., 1953; Hastings, 1970), depending on the relative $\mathcal{L}_{\text{LM}}$ values upon $\mathbf{s}_{N+1}$ and $\hat{\mathbf{s}}_N$. Thus, the proxy signal evolves, enabling $\mathcal{T}$ to be optimized indirectly for $\mathcal{L}_{\text{LM}}$. The detailed implementation is displayed in Appendix B.

Membrane potential-driven loss. To supervise $\mathcal{T}$, we propose a membrane potential-driven loss $\mathcal{L}_{\text{mem}}$ to encourage outputs of $\mathcal{T}$ to approach the proxy signal $\hat{\mathbf{s}}_N$. Specifically, we define $\mathcal{S}_N = \{t \mid \hat{s}_t^N = 1\}$ as the spiking target positions. The loss $\mathcal{L}_{\text{mem}}$ encourages potentials at $\mathcal{S}_N$ to exceed the threshold while suppressing others below the threshold:

$$\begin{aligned} \mathcal{L}_{\text{mem}} = &\sum_{t \in \mathcal{S}_N} \left\| \alpha u_t^{\text{fire}} + (1-\alpha) u_{\text{th}} - u_t \right\|_2^2 \\ &+ \sum_{t \notin \mathcal{S}_N} \left\| \beta u_t^{\text{rest}} + (1-\beta) u_{\text{reset}} - u_t \right\|_2^2 , \end{aligned} \tag{14}$$

where,

$$\begin{aligned} u_t^{\text{fire}} &= \max\left(u_{\text{th}}, \gamma u_{\text{th}} + \delta I_t\right) , \\ u_t^{\text{rest}} &= \min\left(u_{\text{th}} - \epsilon, \gamma u_{\text{reset}} + \delta I_t\right) . \end{aligned}$$

Here, α and β are hyperparameters, and ϵ is a small margin. This loss extends Cao et al. (2024) with explicit supervision on non-spiking positions, which we find essential for stable training. Under $\mathcal{L}_{\text{mem}}$, the spiking output $\mathbf{s}_N$ of $\mathcal{T}$ aligns with $\hat{\mathbf{s}}_N$.

4 Experimental Setup

Our experiments are organized along two comparison frameworks: against GToks and against RToks.

4.1 SOTA Tokenizers

For GToks, we select **MAGNET** (Ahia et al., 2024) and **DTP** (Nawrot et al., 2023); for RToks, we select **DyTok** (Feher et al., 2025) and **ZeTT** (Minixhofer et al., 2024). These methods provide complete, reproducible code and require only moderate training or fine-tuning costs.

4.2 Language Models

In GTok comparisons, we use **Hourglass Transformer** (Nawrot et al., 2022), which is a lightweight model. In RTok comparisons, we use **XLM-R** (Conneau et al., 2020), a widely adopted mainstream LLM.

We do not include experiments with Mistral-7B (used in DyTok and ZeTT; Jiang et al., 2023), as its experimental pipelines are based on evaluation-only benchmarks that do not provide training or validation splits for optimizing our tokenizer. Detailed explanations are provided in the Appendix A.3

4.3 Datasets

For GTok comparisons, we use **text8** (en; Mahoney, 2006), **cc-100** (en; Conneau et al., 2020) and **wiki40b** (en, fi, he, vi; Guo et al., 2020), following DTP's setting. For RTok comparisons, we use **XNLI** (Conneau et al., 2018) and **UNER** (Mayhew et al., 2024), following DyTok's pipeline. These datasets cover 16 languages in total.

4.4 Training Details

When combined with the Hourglass Transformer, we adopt the full model architecture and optimization strategy proposed in DTP to ensure a fair comparison focused on the methodological innovations. Accordingly, the Hourglass Transformer and the tokenizer are trained from scratch.

When integrated with XLM-R, we perform our optimization algorithm and employ the hypernetwork proposed by Minixhofer et al. (2024). The implementation of the hypernetwork and XLM-R follows the setting of DyTok. We fine-tune XLM-R using a LoRA adapter (Hu et al., 2021).

We run each experiment 3 times with different random seeds, presenting only the mean results due to space limit. We perform a comprehensive grid search to identify optimal hyperparameter settings and network configurations, with detailed search results and final selections reported in Appendix C.

5 Results

5.1 Cross-Lingual Adaptability

SOTA GToks are known for their strong cross-lingual adaptability, delivering consistently competitive performance and efficiency across diverse datasets and languages. Meanwhile, SOTA RToks leverage the multilingual modeling capacity of XLM-R to achieve strong cross-lingual results. To compare against both, we evaluate our model on 5 datasets spanning 16 languages in total. Notably, all datasets include English, enabling an additional

Methods	BPC ↓							SF ↑						
	text8 (en)	cc-100 (en)	wiki40b (en)	wiki40b (fi)	wiki40b (he)	wiki40b (vi)	**Avg.**	text8 (en)	cc-100 (en)	wiki40b (en)	wiki40b (fi)	wiki40b (he)	wiki40b (vi)	**Avg.**
Vanilla	1.143	1.225	1.091	0.945	1.274	1.065	1.124	1.0x	1.0x	1.0x	1.0x	1.0x	1.0x	1.00x
DTP (Entropy)	1.138	1.218	1.083	0.949	1.276	1.072	1.123	4.1x	3.8x	4.1x	**4.1x**	3.6x	4.2x	3.98x
DTP (Unigram)	1.134	1.212	1.078	0.937	**1.270**	1.058	1.115	**5.0x**	**4.8x**	**5.0x**	2.1x	1.9x	4.0x	3.80x
MAGNET	1.132	1.212	1.079	0.936	1.281	1.060	1.117	4.6x	4.6x	4.7x	2.6x	**4.4x**	4.3x	4.20x
SPEAK	**1.120***	**1.208**	**1.068***	**0.932**	1.273	**1.049***	**1.108***	4.6x	**4.8x**	4.5x	3.8x	3.8x	**4.4x**	**4.32x**
Δ to MAGNET	-1.2%	-0.4%	-1.1%	-0.4%	-0.8%	-1.1%	-0.9%	0.0x	+0.2x	-0.2x	+1.2x	-0.6x	+0.1x	+0.12x

Table 1: Results of **SPEAK** against GToks. We report test bits per character (BPC), shortening factor (SF) and their mean values. Vanilla represents the vanilla transformer baseline in DTP's work. The symbol * marks statistical significance (i.e., $p < 0.05$) by two-sample t-test for BPC between **SPEAK** and MAGNET. Cyan numbers represent enhancement, while red ones indicate degradation.

Methods	Accuracy per Language (%)													
	ar	bg	de	el	en	es	fr	hi	ru	sw	tr	ur	vi	**Avg.**
ZeTT (SentencePiece)	67.9	73.9	74.1	71.4	81.1	76.2	74.7	67.7	70.7	62.3	68.7	63.2	73.9	71.2
DyTok (sampled)	66.5	74.1	74.5	71.6	84.3	77.0	75.9	64.9	72.7	58.8	66.5	65.1	73.7	71.2
DyTok (50%)	67.8	74.2	74.3	72.4	83.2	**78.3**	75.7	66.6	72.9	61.3	67.5	**66.4**	**75.0**	72.0
SPEAK	**69.3**	**75.9**	**75.3**	73.4	**84.9**	78.0	76.0	**68.9**	**73.5**	63.0	**72.5**	65.5	74.3	**73.1**
ΔAcc. to DyTok 50%	1.5%	1.7%	1.0%	1.0%	1.7%	-0.3%	0.3%	2.3%	0.6%	1.7%	5.0%	-0.9%	-0.7%	1.1%
ΔLen. to ZeTT	3.8%	-8.1%	-7.6%	1.2%	-3.2%	-4.6%	-3.2%	-8.6%	-11%	-8.3%	-11%	-5.8%	-1.1%	-5.3%

Table 2: Results of **SPEAK** against RToks on XNLI dataset. ΔAcc. is the absolute accuracy change between **SPEAK** with DyTok (50%). ΔLen. is the average decrease in token sequence length between **SPEAK** with DyTok (50%). Cyan numbers represent enhancement, while red ones indicate degradation.

assessment of cross-dataset generalizability in English scenarios. For comparison with GToks, we report bits-per-character (BPC) for performance and shortening factor (SF) for efficiency. For RToks, we use per-language accuracy on XNLI and F1-score on UNER.

Results are shown in Table 1, Table 2, and Appendix Table 6. Overall, our method demonstrates superior cross-lingual adaptability, consistently improving performance while maintaining competitive efficiency against both SOTA GToks and RToks. Specifically, our model achieves an average BPC of 1.108, establishing a new SOTA among GToks and significantly outperforming MAGNET ($p < 0.05$). While some baselines improve BPC at the cost of efficiency, our model attains an average SF of 4.32, remaining SOTA among all GToks in efficiency as well. Moreover, we achieve an average accuracy of 73.1% on XNLI and an average F1-score of 79.2% on UNER, consistently outperforming all compared tokenizers. As for efficiency, our method incurs a 5.3% efficiency increase relative to ZeTT (SentencePiece) on XNLI, and a significant 13.4% increase compared to ZeTT (original) on UNER. Further, on all English datasets, our model further shows consistent gains in both performance and efficiency, indicating strong cross-dataset generalization in English scenarios.

In summary, our entropy-aware tokenization enhances token-level representations and token understanding, thus yielding consistent performance gains without sacrificing efficiency. Additionally, we address some possible questions in Appendix D.

5.2 Ablation Analysis

We conduct ablation studies to validate the effectiveness of the two main contributions of our method, as well as the efficacy of the scaling factor and the optimization algorithm. We report results in Table 3 and Appendix Table 5.

Explicitly leverage history. The first contribution is the explicit leverage of historical tokenization results. We ablate our entropy-aware reset into three variants: uniform hard reset (Hard reset, Hard reset, –), uniform soft reset (Soft reset, Soft reset, 1.0), and uniform no reset (No reset, No reset, 0.0). Uniform hard reset discards all history after each token is determined, forcing every token boundary to be decided independently without using any historical tokenization results. In contrast, the latter two variants, as well as our model, leverage history to varying degrees.

High-entropy	Low-entropy	θ	text8				XNLI			
			BPC ↓	p value	SF ↑	p value	en	p value	Avg.	p value
Hard reset	**Soft reset**	**0.8**	**1.120**	–	**4.6x**	–	**84.9%**	–	**73.1%**	–
Hard reset	Hard reset	–	1.134	0.03*	4.1x	<0.01*	82.8%	<0.01*	71.0%	<0.01*
Soft reset	Soft reset	1.0	1.128	0.12	4.5x	0.11	84.3%	0.07	72.3%	0.04*
No reset	No reset	0.0	1.130	0.05*	4.0x	<0.01*	83.2%	<0.01*	71.2%	<0.01*
Hard reset	Soft reset	1.0	1.125	0.12	4.6x	0.11	84.9%	1.0	72.8%	0.21
Hard reset	Soft reset	0.6	1.127	0.13	4.4x	0.13	84.2%	0.05	72.3%	0.04*
Hard reset	Soft reset	0.4	1.132	0.07	4.1x	0.01*	83.7%	0.01*	71.9%	<0.01*
Hard reset	No reset	0.0	1.139	<0.01*	3.7x	<0.01*	82.9%	<0.01*	70.9%	<0.01*

Table 3: Results for ablation analysis. The symbol * marks statistical significance ($p < 0.05$) of two-sample t-test between ablated models with the original model. The first row represents the original model **SPEAK** with the optimal setting.

Dataset	Model	Tokenization time	Speedup	Inference time	Speedup	LM params	TK params	Ratio
wiki40b	MAGNET	7.3 ms / batch	-	10.6 ms / batch	–	34M	6M	15.0%
	SPEAK	6.8 ms / batch	+6.8%	9.7 ms / batch	+8.5%	34M	2M	5.5%
XNLI	ZeTT	5.7 ms / batch	-	15.3 ms / batch	–	270M	–	–
	SPEAK	4.9 ms / batch	+14.0%	14.1 ms / batch	+7.8%	270M	1M	0.4%

Table 4: Tokenization time, inference time and model parameters. We report, in order, tokenization time with relative speedup, language model inference time with relative speedup, language model (LM) parameters, tokenizer (TK) parameters, and the ratio of TK parameters in the overall framework. Note that tokenization and inference times are cross-lingual average values.

As a result, uniform hard reset yields the worst performance and efficiency, with significant drops (both $p < 0.05$) compared to our model, validating the effectiveness of explicitly leveraging historical tokenization results.

Selectively leverage history. The second contribution is the selective leverage of historical tokenization results. To evaluate this aspect, we compare our model with uniform soft reset and uniform no reset, which resemble gated and vanilla recurrent models, respectively. Note that both variants leverage all history without assessing contextual relevance. In contrast, our model selectively preserves relevant history while discarding irrelevant history through the entropy-aware reset mechanism.

The result shows that our method consistently outperforms all ablated variants. This indicates that, beyond explicitly leveraging historical tokenization results, selectively preserving relevant history while discarding irrelevant history can further enhance performance.

Scaled soft reset fow low-entropy tokens. Within our entropy-aware reset, we employ a scaled soft reset controlled by θ for low-entropy tokens to modulate the degree of historical information retention. To assess its necessity, we compare our setting ($\theta = 0.8$) against default soft reset ($\theta = 1$), no reset ($\theta = 0$), and other scaling values.

The result shows that our setting is optimal across all variants, indicating that adjusting historical retention could futher improve performance.

Optimization algorithm. Our optimization algorithm introduces two improvements: an extended membrane potential–driven loss beyond Cao et al. (2024), and MH sampling for momentum-based update. To validate their contributions, we ablate them with analysis and results reported in Appendix B and Table 5. We observe that both improve performance, confirming their effectiveness.

5.3 Tokenization Time, Inference Time and Model Parameters

To quantify the concrete efficiency of our method over other tokenizers, we measure tokenization time, language model inference time, and model parameters, using wiki40b and XNLI datasets. Both tokenization and inference times for each dataset are calculated in a cross-lingual averaging manner. We report results in Table 4.

For tokenization efficiency, our method presents a 6.8% reduction in per-batch tokenization time compared to MAGNET and a 14% speedup over ZeTT (SentencePiece). As for inference efficiency, our method achieves a 8.5% reduction in per-batch inference time compared to MAGNET and a 7.8% speedup over ZeTT (SentencePiece). Moreover,

the tokenizer introduced by our approach is highly parameter-efficient: it constitutes only 5.5% of the total model parameters in wiki40b experiments, substantially lower than MAGNET's 15.0%; on XNLI, the parameter overhead is negligible, contributing merely 0.4% to the overall model size.

These results highlight that our method delivers notable gains in efficiency without inflating the tokenizer's parameter footprint. Crucially, the performance improvements stem from architectural and methodological innovations rather than increased model parameters, demonstrating that our approach achieves efficiency without compromise.

6 Conclusion

We propose **SPEAK**, which integrate spiking neurons into a GTok to explicitly leverage historical tokenization results during token boundary decisions. We innovatively develop an entropy-aware reset mechanism, to selectively leverage historical tokenization results based on contextual relevance, which is determined by token-level entropy. We develop a momentum-based proxy supervision algorithm to optimize our model. Extensive experiments demonstrate superior cross-lingual adaptability with competitive performance and efficiency. Ablation analysis validates the effectiveness of our main contributions.

7 Limitations

Adaptability across language models. In this study, we evaluate our tokenizer on only two models, including one single mainstream LLM. Further experiments on other widely used language models are needed to fully assess its general applicability. Moreover, integrating our tokenizer typically requires training from scratch unless specialized techniques (e.g., hypernetworks) are employed to avoid full fine-tuning, which may limit accessibility to researchers with insufficient computational resources.

Scalability across model sizes. The largest model used in our experiments is XLM-R (base), which contains 270M parameters. Since we leverage hypernetworks and LoRA to avoid full fine-tuning, we do not evaluate larger variants of XLM-R (e.g., Large, XL, and XXL) due to the lack of corresponding trained hypernetworks. A feasible extension would be to train hypernetworks for these larger models following the code guidance of Minixhofer et al. (2024), enabling integration with our method.

Acknowledgements

This work was supported by the National Key R&D Program of China (No. 2025YFG0100700), in part by the National Natural Science Foundation of China (62376247, 62436008, 62334014), in part by Fundamental Research Funds for the Central Universities (226-2025-00057), in part by the grants from Key R&D Program of Zhejiang (2022C01048).

References

Shivam Agarwal, Zimin Zhang, Lifan Yuan, Jiawei Han, and Hao Peng. 2025. The Unreasonable Effectiveness of Entropy Minimization in LLM Reasoning. *Advances in Neural Information Processing Systems*.

Orevaoghene Ahia, Sachin Kumar, Hila Gonen, Valentin Hofmann, Tomasz Limisiewicz, Yulia Tsvetkov, and Noah A. Smith. 2024. MAGNET: Improving the Multilingual Fairness of Language Models with Adaptive Gradient-Based Tokenization. *Advances in Neural Information Processing Systems*.

Mehdi Ali, Michael Fromm, Klaudia Thellmann, Richard Rutmann, Max Lübbering, Johannes Leveling, Katrin Klug, Jan Ebert, Niclas Doll, Jasper Buschhoff, Charvi Jain, Alexander Weber, Lena Jurkschat, Hammam Abdelwahab, Chelsea John, Pedro Ortiz Suarez, Malte Ostendorff, Samuel Weinbach, Rafet Sifa, and 2 others. 2024. Tokenizer choice for llm training: Negligible or crucial? In *Findings of the Association for Computational Linguistics: NAACL*.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, and 12 others. 2020. Language Models are Few-Shot Learners. *Advances in Neural Information Processing Systems*.

A. N. Burkitt. 2006. A Review of the Integrate-and-fire Neuron Model: I. Homogeneous Synaptic Input. *Biological Cybernetics*.

Jiahang Cao, Mingyuan Sun, Ziqing Wang, Hao Cheng, Qiang Zhang, Shibo Zhou, and Renjing Xu. 2024. Spiking neural network as adaptive event stream slicer. *Advances in Neural Information Processing Systems*.

Yi Cao, Jinhao Liang, Tao Liu, Weihui Sang, Yang Gan, Honghong Li, Yue Wang, Zheng Ren, Yuan Yu, Zhou Xin, Yukang Chen, Xumeng Zhang, Du Xiang, and Qi Liu. 2025. Reward-modulated spike-timing-dependent plasticity in van der waals ferroelectric memtransistor for robotic recognition and tracking. *Science Bulletin*.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2020. Unsupervised cross-lingual representation learning at scale. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*.

Alexis Conneau, Ruty Rinott, Guillaume Lample, Adina Williams, Samuel Bowman, Holger Schwenk, and Veselin Stoyanov. 2018. XNLI: Evaluating cross-lingual sentence representations. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*.

Qingkai Fang, Yan Zhou, Shoutao Guo, Shaolei Zhang, and Yang Feng. 2025. LLaMA-omni 2: LLM-based real-time spoken chatbot with autoregressive streaming speech synthesis. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics*.

Sebastian Farquhar, Jannik Kossen, Lorenz Kuhn, and Yarin Gal. 2024. Detecting hallucinations in large language models using semantic entropy. *Nature*.

Darius Feher, Ivan Vulić, and Benjamin Minixhofer. 2025. Retrofitting Large Language Models with Dynamic Tokenization. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics*.

Richard Futrell and Michael Hahn. 2025. Linguistic structure from a bottleneck on sequential information processing. *Nature Human Behaviour*.

Saibo Geng, Nathan Ranchin, Yunzhen yao, Maxime Peyrard, Chris Wendler, Michael Gastpar, and Robert West. 2025. zip2zip: Inference-Time Adaptive Tokenization via Online Compression. *Advances in Neural Information Processing Systems*.

Nathan Godey, Roman Castagné, Éric de la Clergerie, and Benoît Sagot. 2022. MANTa: Efficient gradient-based tokenization for end-to-end robust language modeling. In *Findings of the Association for Computational Linguistics: EMNLP*.

Mandy Guo, Zihang Dai, Denny Vrandečić, and Rami Al-Rfou. 2020. Wiki-40B: Multilingual language model dataset. In *Proceedings of the Twelfth Language Resources and Evaluation Conference*.

W. K. Hastings. 1970. Monte Carlo sampling methods using Markov chains and their applications. *Biometrika*.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *ArXiv preprint*. ArXiv:2106.09685.

Yangfan Hu, Huajin Tang, and Gang Pan. 2023a. Spiking Deep Residual Networks. *IEEE Transactions on Neural Networks and Learning Systems*.

Yangfan Hu, Qian Zheng, Xudong Jiang, and Gang Pan. 2023b. Fast-SNN: Fast Spiking Neural Network by Converting Quantized ANN. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Eric Hunsberger and Chris Eliasmith. 2015. Spiking Deep Networks with LIF Neurons. *ArXiv preprint*. ArXiv:1510.08829.

Md. Mahadi Islam, Gustavo Aguilar, Prasanna Ponnusamy, C. S. Mathialagan, Chen Ma, and Chao Guo. 2022. A vocabulary-free multilingual neural tokenizer for end-to-end task learning. *ArXiv preprint*. ArXiv:2204.10815.

Eric Jang, Shixiang Gu, and Ben Poole. 2017. Categorical Reparameterization with Gumbel-Softmax. In *Proceedings of International Conference on Learning Representations*.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023. Mistral 7b. *ArXiv preprint*. ArXiv:2310.06825.

Taku Kudo. 2018. Subword regularization: Improving neural network translation models with multiple subword candidates. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*.

Taku Kudo and John Richardson. 2018. Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. *ArXiv preprint*. ArXiv:1808.06226.

Haoyan Luo and Lucia Specia. 2024. From understanding to utilization: A survey on explainability for large language models. *ArXiv preprint*. ArXiv:2401.12874.

De Ma, Xiaofei Jin, Shichun Sun, Yitao Li, Xundong Wu, Youneng Hu, Fangchao Yang, Huajin Tang, Xiaolei Zhu, Peng Lin, and Gang Pan. 2024. Darwin3: a large-scale neuromorphic chip with a novel ISA and on-chip learning. *National Science Review*.

Xiaoxiao Ma, Feng Zhao, Pengyang Ling, Haibo Qiu, Zhixiang Wei, Hu Yu, Jie Huang, Zhixiong Zeng, and Lin Ma. 2025. Towards Better & Faster Autoregressive Image Generation: From the Perspective of Entropy. *Advances in Neural Information Processing Systems*.

Chris J. Maddison, Andriy Mnih, and Yee Whye Teh. 2017. The Concrete Distribution: A Continuous Relaxation of Discrete Random Variables. In *Proceedings of International Conference on Learning Representations*.

Matt Mahoney. 2006. Large text compression benchmark.

Stephen Mayhew, Terra Blevins, Shuheng Liu, Marek Šuppa, Hila Gonen, Joseph Marvin Imperial, Börje F. Karlsson, Peiqin Lin, Nikola Ljubešić, LJ Miranda, Barbara Plank, Arij Riabi, and Yuval Pinter. 2024. Universal NER: A gold-standard multilingual named entity recognition benchmark. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Nicholas Metropolis, Arianna W. Rosenbluth, Marshall N. Rosenbluth, Augusta H. Teller, and Edward Teller. 1953. Equation of State Calculations by Fast Computing Machines. *The Journal of Chemical Physics*.

Benjamin Minixhofer, Edoardo M. Ponti, and Ivan Vulić. 2024. Zero-Shot Tokenizer Transfer. *Advances in Neural Information Processing Systems*.

Piotr Nawrot, Jan Chorowski, Adrian Lancucki, and Edoardo Maria Ponti. 2023. Efficient Transformers with Dynamic Token Pooling. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics*.

Piotr Nawrot, Szymon Tworkowski, Michał Tyrolski, Lukasz Kaiser, Yuhuai Wu, Christian Szegedy, and Henryk Michalewski. 2022. Hierarchical transformers are more efficient language models. In *Findings of the Association for Computational Linguistics: NAACL*.

Yerim Oh, Jun-Hyung Park, Junho Kim, SungHo Kim, and SangKeun Lee. 2025. Incorporating Domain Knowledge into Materials Tokenization. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics*.

Zexuan Qiu, Zijing Ou, Bin Wu, Jingjing Li, Aiwei Liu, and Irwin King. 2025. Entropy-Based Decoding for Retrieval-Augmented Large Language Models. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*.

Sarath Sivaprasad, Pramod Kaushik, Sahar Abdelnabi, and Mario Fritz. 2025. A theory of response sampling in LLMs: Part descriptive and part prescriptive. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics*.

Jianlin Su. 2023. Bytepiece: A more pure and effective tokenizer.

Yi Tay, Vinh Q. Tran, Sebastian Ruder, Jai Gupta, H. W. Chung, Dara Bahri, Zhen Qin, Steven Baumgartner, Cong Yu, and Donald Metzler. 2022. Charformer: Fast character transformers via gradient-based subword tokenization. In *Proceedings of International Conference on Learning Representations*.

Avijit Thawani, Saurabh Ghanekar, Xiaoyuan Zhu, and Jay Pujara. 2023. Learn your tokens: Word-pooled tokenization for language modeling. In *Findings of the Association for Computational Linguistics: EMNLP*.

Refael Tikochinski, Ariel Goldstein, and Roi Reichart. 2025. Incremental accumulation of linguistic context in artificial and biological neural networks. *Nature Communications*.

Dixuan Wang, Yanda Li, Junyuan Jiang, Zepeng Ding, Ziqin Luo, Guochao Jiang, Jiaqing Liang, and Deqing Yang. 2024. Tokenization matters! degrading large language models through challenging their tokenization. *ArXiv preprint*. ArXiv:2405.17067.

Shenzhi Wang, Le Yu, Chang Gao, Chujie Zheng, Shixuan Liu, Rui Lu, Kai Dang, Xionghui Chen, Jianxin Yang, Zhenru Zhang, Yuqiong Liu, An Yang, Andrew Zhao, Yang Yue, Shiji Song, Bowen Yu, Gao Huang, and Junyang Lin. 2025. Beyond the 80/20 Rule: High-Entropy Minority Tokens Drive Effective Reinforcement Learning for LLM Reasoning. *Advances in Neural Information Processing Systems*.

Xuan Wang, Sebastian Ruder, and Graham Neubig. 2021. Multi-view subword regularization. *ArXiv preprint*. ArXiv:2103.08490.

Yu Xiao, Yize Liu, Bihua Zhang, Peng Chen, Huaze Zhu, Enhui He, Jiayi Zhao, Wenju Huo, Xiaofei Jin, Xumeng Zhang, Hao Jiang, De Ma, Qian Zheng, Huajin Tang, Peng Lin, Wei Kong, and Gang Pan. 2025. Bio-plausible reconfigurable spiking neuron for neuromorphic computing. *Science Advances*.

Jiaqi Yan, Changping Wang, De Ma, Huajin Tang, Qian Zheng, and Gang Pan. 2025. Training High Performance Spiking Neural Network by Temporal Model Calibration. In *Proceedings of International Conference on Machine Learning*.

Sangwoong Yoon, Himchan Hwang, Dohyun Kwon, Yung-Kyun Noh, and Frank C. Park. 2024. Maximum Entropy Inverse Reinforcement Learning of Diffusion Models with Energy-Based Models. *Advances in Neural Information Processing Systems*.

Sheldon Yu, Yuxin Xiong, Junda Wu, Jingbo Shang, Julian McAuley, and 1 others. 2025. Explainable chain-of-thought reasoning: An empirical analysis on state-aware reasoning dynamics. In *Findings of the Association for Computational Linguistics: EMNLP*.

Jingyang Yuan, Huazuo Gao, Damai Dai, Junyu Luo, Liang Zhao, Zhengyan Zhang, Zhenda Xie, Yuxing Wei, Lean Wang, Zhiping Xiao, Yuqing Wang, Chong Ruan, Ming Zhang, Wenfeng Liang, and Wangding Zeng. 2025. Native sparse attention: Hardware-aligned and natively trainable sparse attention. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics*.

Mengyu Zheng, Hanting Chen, Tianyu Guo, Chong Zhu, Binfan Zheng, Chang Xu, and Yunhe Wang. 2024.

Enhancing Large Language Models through Adaptive Tokenizers. *Advances in Neural Information Processing Systems*.

Qian Zheng, Ming Chen, Sha Zhao, Shi Gu, Peng Lin, De Ma, Huajin Tang, and Gang Pan. 2026. S³: Spiking Neurons as an Isolating Segmenter for Brain Signal Decoding. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

A Discussions of Possible Issues

A.1 Spiking over Recurrent

This section explains why we choose spiking neural networks over recurrent neural networks as tokenizers, although both can explicitly leverage historical tokenization results. In summary, there are two main reasons:

- Spiking networks have a unique mechanism for utilizing historical information—namely, hard reset, which discards past states.
- Spiking networks allow flexible control over reset modes of each spiking neuron, enabling the combination of different historical utilization strategies, which cannot be achieved by recurrent networks.

Detailed explanations are as follows.

The spiking networks utilize the unique reset mechanism of spiking neurons to control the leverage of historical information. Specifically, hard reset clears all history, soft reset remains partial history, and no reset leverages all history. Within a spiking network, the reset mode for any specific neuron can be easily manipulated. For example, some spiking neurons adopt hard reset while some neurons adopt soft reset. Further, these reset configurations can be dynamically adjusted during training to identify the optimal combination.

In contrast, recurrent networks lack both this diverse control over historical information and the ability to combine different historical retention modes. Specifically, vanilla recurrent networks indiscriminately leverage all history, while more advanced gated variants (e.g., LSTM, GRU) selectively filter history via gating mechanisms. However, there is no one that would clear history, which is significant when removing irrelevant or even harmful history is required (e.g., "german" -> "german | ium" exemplified in Figure 1). Furthermore, the recurrence mode of recurrent units cannot be changed adaptively (e.g., change a vanilla unit to a gated unit), because they fundamentally use different update formula and rely on different learnable parameters.

A.2 Top 20% as High-Entropy Tokens

We follow Wang et al. (2025) and select the top 20% of tokens ranked by their entropy values as high-entropy tokens, with the remainder treated as low-entropy tokens. We qualitatively examine the resulting high-entropy tokens and compare them with those reported in Wang et al. (2025). We observe a strong alignment between our token-level entropy patterns and the findings of Wang et al. (2025). In addition, the entropy value of the 80th percentile is also close. Specifically, high-entropy tokens frequently function as logical connectors within or across sentences, such as "wait," "however," and "unless" (indicating contrast or shifts), "thus" and "also" (signaling progression or addition), and "since" and "because" (expressing causality). In contrast, low-entropy tokens are often word suffixes, source code fragments, or components of mathematical expressions, which exhibit high determinism. Furthermore, we perform a grid search over several ratios around 20% and empirically observe that 20% yields stable and favorable performance. Readers are referred to Wang et al. (2025) for a more detailed analysis.

A.3 Experiments on Decoder-Style LLMs

Our experiments do not include decoder-style LLMs such as LLaMA or Mistral, but this does not imply that the proposed method is incompatible with such models. Rather, applying our tokenizer to decoder-only LLMs requires a separate tokenizer pretraining stage.

Specifically, deploying SPEAK in decoder-style LLMs would involve pretraining the tokenizer on a large corpus to learn high-quality tokenization strategies, after which the tokenizer can be fixed and seamlessly integrated into standard decoder-only architectures. Once pretrained, the tokenizer can be evaluated on common benchmarks for decoder-style LLMs (e.g., MT-Bench) without modifying the downstream model.

Our paper focuses on establishing the framework, mechanism, and theoretical advantages of selective history leveraging. Extending the approach to large-scale decoder-only LLMs is primarily an engineering and computational effort, which we leave for future work.

A.4 Why Not Use Linguistic Boundaries

An alternative approach to identifying contextual boundaries would be to rely on discourse structure or syntactic boundaries. While these signals are intuitively appealing, they pose practical limitations.

First, generation entropy has been extensively validated across recent LLM studies as a strong indicator of uncertainty and reasoning divergence, making it a reliable signal for contextual relevance. Second, discourse or syntactic boundaries typically

require labeled data, external parsers, or additional models to infer such structures.

Introducing such prior knowledge or auxiliary components would reduce the generalizability of our framework and introduce additional dependencies. In contrast, entropy is model-intrinsic, task-agnostic, and universally available during inference, allowing our approach to remain simple, general, and self-contained.

B Details of the Optimization Algorithm

B.1 Formulation

MH sampling to refine proxy signal. To progressively optimize the proxy supervision signal $\hat{\mathbf{s}}_N$, we adopt a greedy strategy that prioritizes candidates yielding lower $\mathcal{L}_{\text{LM}}$. However, a standard greedy search may lead to premature convergence, yielding suboptimal performance. Inspired by the Metropolis–Hastings (MH) algorithm (Metropolis et al., 1953; Hastings, 1970), we introduce stochasticity into the greedy updates:

$$\hat{\mathbf{s}}_{N+1} = \begin{cases} \mathbf{s}_{N+1}, & u \leqslant \alpha\,, \\ \hat{\mathbf{s}}_N, & u > \alpha\,, \end{cases} \tag{15}$$

where

$$\alpha = \frac{\mathcal{L}_{\text{LM}}(\hat{\mathbf{s}}_N)}{\mathcal{L}_{\text{LM}}(\mathbf{s}_{N+1})} \quad \text{and} \quad u \sim \mathcal{U}(0,1). \tag{16}$$

Here, α is the acceptance ratio, and u is a uniform random number. $\mathcal{L}_{\text{LM}}(\mathbf{s})$ denotes the objective loss computed using spiking outputs $\mathbf{s}$ for tokenization.

Lower-loss candidates are always accepted (i.e., $\alpha > 1$), yielding a "greedy" update. Candidates with slightly higher loss may still be accepted "stochastically", enabling exploration and preventing premature convergence. Through this process, $\mathcal{T}$ is optimized to minimize $\mathcal{L}_{\text{LM}}$, effectively circumventing the gradient blockade between them.

Proxy signal initialization. Optimizing tokenizers often confronts unstability and occasional failure. Prior works encourage aligning with or starting with a well-established tokenizer (Nawrot et al., 2023; Zheng et al., 2024). In line with it, we initialize $\hat{\mathbf{s}}_1$ to be the tokenization outcomes of an exemplar tokenizer such as Unigram or XLMR's tokenizer, which ensures a reliable start.

B.2 Ablation analysis.

We conduct ablation studies on the XNLI dataset to evaluate two key components: (1) our extension to the membrane potential–driven loss, and (2) the use of MH sampling in place of the standard greedy strategy. Results are reported in Table 5. Specifically, we implement two variants: **SPEAK** w/o $\mathcal{L}_{\text{mem}}$ 2nd term, which uses the original loss from Cao et al. (2024), and **SPEAK** w/o MH sampling, which replaces MH sampling with the standard greedy strategy. The results clearly validate the effectiveness of both innovations.

Methods	**en**	p value	**Avg.**	p value
SPEAK	**84.9%**	–	**73.1%**	–
w/o $\mathcal{L}_{\text{LM}}$ 2$^{\text{nd}}$ term	83.0%	<0.01*	70.7%	<0.01*
w/o MH sampling	83.5%	0.01*	71.9%	0.04*

Table 5: Results for ablation analysis. The symbol * marks statistical significance ($p < 0.05$) of two-sample t-test between ablated models with the original model. The first row represents the original model **SPEAK**.

B.3 Possible questions.

We address some issues readers might have:

- Effectiveness: Our "stochastic", inspired by the MH algorithm, encourages exploration to escape local optima and avoid premature convergence for enhancing performance. Empirically, replacing stochastic-greedy with standard greedy shows a performance drop.
- Rejection rate: The rejection rate of our greedy strategy is dynamic throughout training. In early epochs, the rate is relatively low (around 10%), reflecting active exploration. In late epochs, the rate increases (around 90%), indicating a robust selection.
- Convergence stability: Our stochastic-greedy optimization is in spired by Metropolis-Hastings algorithm, so we have similar convergence stability as it.

C Hyperparameters and Network Configurations

Our configuration combines established defaults, library conventions, and empirical tuning. For reproducibility and alignment with common practice, several parameters adopt standard values: $\gamma = \delta = 1$, $u_{\text{th}} = 1$, $u_{\text{reset}} = 0$, and $\epsilon = 10^{-8}$. The SNN optimization follows the default settings of the `spikingjelly` package, including the use of a sigmoid surrogate gradient. We now discuss the remaining configurations separately for the two experimental frameworks.

Methods	Language F1-score (%)					
	en_ewt	de_pud	pt_bosque	pt_pud	ru_pud	**Avg.**
ZeTT (original)	80.9	**78.3**	80.8	82.3	68.4	78.1
DyTok (sampled)	81.3	76.3	78.5	80.2	67.1	76.7
DyTok (75%)	80.5	75.0	80.5	81.3	67.9	77.0
SPEAK	**81.8**	77.7	**82.8**	**83.6**	**69.9**	**79.2**
ΔF1 to ZeTT	0.9%	-0.6%	2.0%	1.3%	1.5%	1.1%
ΔLen. to ZeTT	-8.3%	-16.1%	-12.9%	-10.7%	-19.2%	-13.4%

Table 6: Results of **SPEAK** against RToks on UNER dataset. The results reported are on the validation split for ewt and bosque datasets, and test split for pud due to the availability.

C.1 GTok comparing experiments.

The network configuration of our SNN-based GTok closely follows that of DTP, including the architecture and dimensionality of the ContextEncoder and Linear_1. All other training details, such as the optimizer and learning-rate scheduler, also follow DTP's settings. The parameters α and β are not manually specified; instead, they are treated as learnable parameters and optimized jointly during training. This design choice is motivated by the fact that GTok experiments require training the entire language model, and empirically learning α and β yields better performance. As a result, GTok experiments require minimal hyperparameter search, and all configurations can be directly referenced in our released code.

C.2 RTok comparing experiments.

Unlike GTok, RTok experiments require a more involved configuration process. We adopt a two-stage tuning protocol to balance computational efficiency and performance.

Stage 1: Tokenizer tuning. Due to the high cost of full end-to-end evaluation, we first tune the tokenizer in isolation. Specifically, we train the tokenizer to mimic the segmentation produced by the default XLM-R tokenizer, and evaluate spike prediction accuracy, measured by the alignment between predicted and target spiking/non-spiking positions. This proxy task effectively captures the learning capacity and adaptability of the SNN.

We perform grid search over the following factors: the architecture of the ContextEncoder (Hourglass Transformer, Hourglass without multi-head attention, or a Linear–LayerNorm–GELU block); the design of Linear_1 (a Linear layer with or without $\tanh$ activation); the parameters $\alpha \in [0.8, 1.4]$ and $\beta \in [0.3, 0.8]$; and the character embedding dimension chosen from $\{64, 100, 128, 256, 512, 768, 1024\}$. The final configuration adopts a Linear–LayerNorm–GELU ContextEncoder, Linear–$\tanh$ for Linear_1, $\alpha = 1.2$, $\beta = 0.6$, and a character embedding dimension of 128. Representative results are reported in Table 7.

Stage 2: End-to-end tuning. A small number of parameters critically influence end-to-end training and therefore must be tuned within the complete framework. These include the soft reset scaling factor θ and the architecture of Linear_2. Based on limited trials, we set $\theta = 0.8$ and implement Linear_2 as a single Linear layer. The results from the search over θ are reported in the ablation study (Table 3).

D Possible Issues of Table Results

We address some possible issues regarding the results in Table 1, Table 2, and Table 6.

D.1 Obtaining results of compared tokenizers

GTok Experiments. During GTok experiments, since we follow DTP's experimental pipeline, we can directly copy the numbers presented in their paper (Nawrot et al., 2023). As for MAGNET, it is an improved version of DTP and adopts the Gumbel-Sigmoid strategy (one of multiple strategies proposed in DTP). Since MAGNET does not evaluate on the datasets in DTP's experimental pipeline, we reproduce its code and record the results until we achieve better performance than DTP (Gumbel).

RTok Experiments. During RTok experiments, since we follow DyTok's experimental pipeline, all DyTok results are directly copied from their work (Feher et al., 2025). Additionally, DyTok's experimental pipeline partially follows ZeTT's, sharing certain settings such as the language model and some datasets. Since both DyTok and ZeTT evaluate on XNLI using XLMR, we can copy ZeTT's results from their paper (Minixhofer et al., 2024), which trains language-specific monolingual tok-

enizers with a vocabulary size of 50k using SentencePiece (Kudo and Richardson, 2018). This ZeTT version is denoted as ZeTT (SentencePiece) in our table. However, ZeTT's original paper does not evaluate on UNER. However, in DyTok's paper, they apply ZeTT on top of XLMR's tokenizer, denoted as "original, HN" in their Table 3. Hence, we copy this result into our table for comparison, denoted as ZeTT (original).

Thus, a distinction should be noted: ZeTT (SentencePiece) and ZeTT (original) represent entirely different tokenization strategies, which may confuse readers regarding efficiency comparisons: why SPEAK vs. ZeTT (SentencePiece) shows only -5.3%, while SPEAK vs. ZeTT (original) shows a much larger -13.4%. We discuss this further in Appendix D.4.

D.2 Efficiency comparison

The performance comparison is clear, but the efficiency comparison may cause ambiguity.

GTok Experiment. Initially, we copy the baseline results, i.e., Vanilla Transformer ("Vanilla" in our Table 1), from the DTP's paper. This baseline acts as an anchor, such that all tokenizers should use their respective resultant token sequence lengths to compute the shorten factor (SF) in comparison with this baseline. However, we do not reproduce this baseline and therefore do not know the anchor token sequence length. In contrast, we reproduce DTP (Unigram) to measure its token sequence length while ensuring the reproduced performance closely matches their reported results. Hence, we can compute our SF relative to DTP (Unigram), and thereby infer the comparison with the baseline.

RTok Experiment. When comparing with ZeTT and DyTok, such complex dependencies are unnecessary. DyTok already provides a baseline through its "word-level tokenization". In their work, word-level tokenization means segmenting the character sequence word by word, without splitting any word into multiple parts. Since the number of words can be easily counted, we can, based on DyTok's reported efficiency gains, back-calculate the token sequence length of the original tokenization. Thus, we can compute the token sequence length of ZeTT (SentencePiece) using results reported in their paper, and further compare it with our model. Moreover, the token sequence length of ZeTT (original) is identical to that of the original tokenization, so we can also make direct comparisons.

D.3 Efficiency comparison with ZeTT not DyTok

As stated in Appendix D.2, DyTok's results reported in their paper and included in our tables are based on word-level tokenization. Hence, their efficiency is inherently the highest possible. If we achieved higher efficiency than DyTok, it would imply merging multiple words into a single token, which cannot occur in the 16 languages included in our paper (though it might happen for Chinese characters). In fact, DyTok is a method that gradually reduces token sequence length and ultimately converges to word-level tokenization. However, DyTok only provides a plot of this gradual process, with explicit numerical results reported only for the final word-level tokenization. Therefore, we only include these final numbers in our tables. As for why we do not report intermediate results from this process: to avoid concerns about number fabrication, we rely on directly copied results from their paper rather than reproductions. Consequently, efficiency comparison with DyTok is meaningless, whereas comparison with ZeTT remains highly informative.

D.4 Reason that SPEAK vs. ZeTT (original) is much higher than SPEAK vs. ZeTT (SentencePiece)

As raised in the final part of Appendix D.1, a discrepancy warrants explanation.

From the discussion in Apendix D.2, we now know that ZeTT (SentencePiece) results are directly copied from their paper. Thus, ZeTT (SentencePiece) already achieves a certain level of efficiency, as reported in their tables. Consequently, comparisons against it cannot yield large improvements like the -10%∼-30% reported in DyTok's paper. Indeed, on XNLI, ZeTT (SentencePiece) achieves an average reduction of -13.5%, while DyTok achieves -22.5%. Our model achieves -18.1% relative to the original tokenization, which translates to a relative -5.3% improvement over ZeTT (SentencePiece), exactly as reported in our table. The calculation is:

$$\frac{(1-18.1\%)-(1-13.5\%)}{1-13.5\%} = -5.3\% \quad (17)$$

In contrast, the comparison with ZeTT (original) is directly against the original tokenization, yielding -13.4%. This number is on the same scale as the aforementioned -18.1%, not relative to an already

efficient tokenizer. This detailed calculation explains the observed difference in reported numbers.

E Isolating vs. Non-Isolating Tokenizer

In this section, we aim to theoretically demonstrate the advantage of isolation, showing that our method, which isolates irrelevant historical information, outperforms other tokenizers that leverage all historical information.

We define a tokenizer $\mathcal{T}$ as a causal decision function:

$$\mathcal{T} : \bigcup_{t=1}^{T} \mathbb{R}^t \to \{0,1\}, \quad b_t = \mathcal{T}(c_{1:t}), \tag{18}$$

which sequentially processes the character sequence $\mathbf{c}$ and outputs a binary token boundary $b_t \in \{0,1\}$ at each position t. The boundary decision at position t depends only on the observed prefix $c_{1:t}$, ensuring strict sequential causality. D represents the character embedding dimension.

We propose isolating the temporal influence of historical token features and tokenization decisions upon every high-entropy token. The following proposition formalizes the advantage of such isolation.

Proposition 1. *We consider a high-entropy token $t_k = \{c_{\text{start}}, \ldots, c_{\text{end}}\}$, and $c_{\text{start}-1}$ is a character activating a token boundary (i.e., $b_{\text{start}-1} = 1$). We have:*

$$\mathbb{P}\big(\mathcal{T}(c_{\text{start:end}}) = 1\big) \geqslant \mathbb{P}\big(\mathcal{T}(c_{1:\text{end}}) = 1\big), \tag{19}$$

The probability that an isolating tokenizer activates a token boundary at end *is no less than that of a non-isolating tokenizer that conditions on the full history. That is, clearing historical states preceding a high-entropy token does not degrade but generally improves the reliability of boundary detection.*

Proof. Assume $\mathcal{T}$ is a fixed sequential neural network (e.g., RNN, SNN) that processes an input sequence $\mathbf{c} = (c_1, \ldots, c_L)$ and outputs a token boundary decision at the final step:

$$\mathcal{T}(\mathbf{c}) = \psi(h_L) \in \{0,1\},\ h_\ell = f(h_{\ell-1}, c_\ell), \tag{20}$$

where f is the recurrent update and ψ maps the final hidden state h to $\{0,1\}$. Typically, $h_0 = \mathbf{0}$.

Now consider the high-entropy token $t_k = \{c_{\text{start}}, \ldots, c_{\text{end}}\}$. Define two input sequences to the same tokenizer $\mathcal{T}$:

- **Isolating input**: $\mathbf{c}^{\text{iso}} = c_{\text{start:end}}$ (length $L = \text{end} - \text{start} + 1$).
- **Non-isolating input**: $\mathbf{c}^{\text{full}} = c_{1:\text{end}}$ (length end).

Let h^{iso} and h^{full} denote the hidden states of $\mathcal{T}$ at the final step for these two inputs:

$$h^{\text{iso}} = f^{(L)}(\mathbf{0}, c_{\text{start:end}}), \tag{21}$$

$$h^{\text{full}} = f^{(\text{end})}(\mathbf{0}, c_{1:\text{end}}), \tag{22}$$

where $f^{(n)}$ denotes n recursive applications of f.

Due to the property of high entropy, the token t_k suggests minimal or no semantic or syntactic continuity between preceding and subsequent content, or we can say nearly independent. Hence, the distribution of $c_{\text{start:end}}$ approximately remains whether or not it is preceded by other data. However, the hidden state h^{full} depends on $c_{1:\text{start}-1}$ through the recurrence:

$$h^{\text{full}} = f^{(L)}\big(\underbrace{f^{(\text{start}-1)}(\mathbf{0}, c_{1:\text{start}-1})}_{h_{\text{prev}}},\ c_{\text{start:end}}\big). \tag{23}$$

Since h_{prev} is a deterministic function of somewhat irrelevant, independent data with $c_{\text{start:end}}$, it acts as an **uninformative and potentially misleading** initial state for processing the current segment. In contrast, the isolating version starts from the canonical initial state $\mathbf{0}$.

Now, the token boundary at end corresponds to a change in temporal patterns (e.g., linguistic properties) after end. To detect this, the tokenizer $\mathcal{T}$ must assess whether the observed sequence $c_{\text{start:end}}$ is internally stable. This assessment is optimal when the recurrent state is initialized to a known, neutral state (e.g., $\mathbf{0}$), as in the isolating case.

Because h_{prev} is independent of $c_{\text{start:end}}$ but arbitrary in value, it introduces variance without signal into h^{full}. To formulate this, for any measurable decision function ψ, we have:

$$\mathbb{P}\big(\psi(h^{\text{iso}}) = 1\big) \ \geqslant\ \mathbb{P}\big(\psi(h^{\text{full}}) = 1\big). \tag{24}$$

Therefore,

$$\mathbb{P}\big(\mathcal{T}(c_{\text{start:end}}) = 1\big) \geqslant \mathbb{P}\big(\mathcal{T}(c_{1:\text{end}}) = 1\big), \tag{25}$$

which completes the proof.

Linear_1	ContextEncoder	α	β	Char Dim	Spike Acc	Non-spike Acc	Acc Sum
Baseline and linear/context variants (α=1.0, β=0.5, dim=512)							
no tanh	1 layer	1.0	0.5	512	76.8	83.3	160.1
no tanh	hourglass noatt	1.0	0.5	512	71.3	83.0	154.3
tanh	1 layer	1.0	0.5	512	77.0	83.3	160.3
tanh	hourglass noatt	1.0	0.5	512	76.8	83.2	160.0
Effect of α (β=0.5, dim=512, tanh, 1-layer)							
tanh	1 layer	0.8	0.5	512	64.07	86.5	150.57
tanh	1 layer	1.0	0.5	512	77.0	83.3	160.3
tanh	1 layer	1.2	0.5	512	82.24	81.21	163.45
tanh	1 layer	1.3	0.5	512	83.41	79.93	163.34
tanh	1 layer	1.4	0.5	512	84.1	78.5	162.6
Effect of β (α=1.2, dim=512, tanh, 1-layer)							
tanh	1 layer	1.2	0.3	512	73.46	83.89	157.35
tanh	1 layer	1.2	0.4	512	81.89	81.41	163.3
tanh	1 layer	1.2	0.5	512	82.24	81.21	163.45
tanh	1 layer	1.2	0.6	512	82.69	80.96	163.65
tanh	1 layer	1.2	0.7	512	82.88	80.73	163.61
tanh	1 layer	1.2	0.8	512	83.17	80.42	163.59
Character embedding dimension sweep (tanh, α=1.2, β=0.6, 1-layer)							
tanh	1 layer	1.2	0.6	32	83.24	81.01	164.25
tanh	1 layer	1.2	0.6	64	83.11	81.01	164.12
tanh	1 layer	1.2	0.6	100	79.72	81.15	160.87
tanh	1 layer	1.2	0.6	128	83.4	81.03	164.43
tanh	1 layer	1.2	0.6	256	82.91	81.03	163.94
tanh	1 layer	1.2	0.6	512	82.69	80.96	163.65
tanh	1 layer	1.2	0.6	768	55.26	46.05	101.31
tanh	1 layer	1.2	0.6	1024	55.26	46.05	101.31
Character embedding dimension sweep (no tanh, α=1.2, β=0.6, 1-layer)							
no tanh	1 layer	1.2	0.6	32	82.95	80.91	163.86
no tanh	1 layer	1.2	0.6	64	82.64	80.92	163.56
no tanh	1 layer	1.2	0.6	100	79.72	81.15	160.87
no tanh	1 layer	1.2	0.6	128	83.21	80.91	164.12
no tanh	1 layer	1.2	0.6	256	81.97	80.94	162.91
no tanh	1 layer	1.2	0.6	512	82.77	80.9	163.67
no tanh	1 layer	1.2	0.6	768	83.04	80.9	163.94
no tanh	1 layer	1.2	0.6	1024	83.14	80.91	164.05
Context ablation (tanh, α=1.2, β=0.6, dim=512)							
tanh	1 layer	1.2	0.6	512	82.69	80.96	163.65
tanh	hourglass noatt	1.2	0.6	512	81.91	80.79	162.7
tanh	hourglass	1.2	0.6	512	80.79	81.06	161.85
Context ablation (tanh, α=1.2, β=0.6, dim=128)							
tanh	1 layer	1.2	0.6	128	83.4	81.03	164.43
tanh	hourglass noatt	1.2	0.6	128	80.8	80.34	161.14
tanh	hourglass	1.2	0.6	128	82.17	81.54	163.71

Table 7: Hyperparameter tuning results.