LEARNING REPRESENTATIONS AS RESISTANCE STATES: STRUCTURE-AWARE NEUROMORPHIC SEQUENCE MODELING

Anonymous authors
Paper under double-blind review

ABSTRACT

We propose NeuSpell, a neuromorphic framework that learns representations as resistance states in a memristor crossbar array, offering a non-von Neumann alternative to token-based sequence modeling. Instead of digital tokenization and GPU-accelerated pipelines, structural components of input sequences are directly encoded as conductance dynamics, enabling massively parallel, in-memory inference with ultra-low latency (0.07 ms) and near-zero energy cost. Applied to Tibetan syllable recognition, a task where conventional models struggle due to morphological complexity and data scarcity, NeuSpell achieves 98.2% F1 on the SSC-TiM corpus and TUSA benchmark, outperforming state-of-the-art neural and large language models. Beyond this application, our results suggest that resistance-state dynamics can serve as a new foundation for structure-aware, token-free representation learning, opening a path toward efficient neuromorphic architectures that move beyond von Neumann computation.

1 INTRODUCTION

Large language models (LLMs) and other sequence modeling architectures have achieved remarkable progress across a wide range of natural language processing (NLP) tasks Achiam et al. (2023); Hurst et al. (2024); Guo et al. (2025); Liu et al. (2024); Touvron et al. (2023a;b); Grattafiori et al. (2024); Bai et al. (2023a;b; 2024); Yang et al. (2024). However, their reliance on von Neumann architectures, digital tokenization, and GPU-accelerated pipelines imposes fundamental limitations Bostrom & Durrett (2020); Jouppi et al. (2017); Han et al. (2016). Token-based representations inflate sequence length in morphologically rich languages, while memory bandwidth and energy constraints hinder real-time or edge deployment Kudo & Richardson (2018); Batsuren et al. (2022); Sze et al. (2017). As models continue to scale, the gap between algorithmic advances and hardware efficiency becomes increasingly pronounced, calling for alternatives that rethink how representations are learned and computed Arya et al. (2024).

In this paper, we argue that neuromorphic hardware offers a promising foundation for next-generation sequence modeling. Specifically, we explore *resistance states* in the *memristor crossbar array* Chua (1971); Jo et al. (2010b) as a new medium for representation learning. Instead of encoding linguistic units as discrete tokens for sequential processing, we directly map structural components into conductance dynamics, enabling massively parallel, in-memory inference. This design bypasses the data movement bottleneck of von Neumann systems and eliminates dependence on GPUs, while also providing an inductive bias well-suited for structure-aware tasks.

Among the many sequence modeling tasks, we focus on **Tibetan spelling error detection and correction**, a problem that epitomizes the challenges of morphologically rich and low-resource languages. Tibetan syllables are composed of hierarchical units (prefixes, roots, subscripts, superscripts, vowels, and suffixes) Zhang et al. (2021), making token-based models inefficient and error-prone, which is shown in Figure 1. Spelling errors are especially frequent in handwritten or digitized Tibetan texts, where a single misplaced component can change meaning entirely. Existing approaches, ranging from handcrafted rule-based systems Wang & Duola (2008); Huang & Da (2010); Zhu et al. (2013); wangdui et al. (2014); Brown (2024); Wan & He (2015) to deep learning models San et al. (2021); Jie et al. (2014); Zhu et al. (2014); Liu et al. (2017); Danzhaxi et al. (2020),

Tokenizer	Input	Words	Number of tokens after word segmentation	Output
Llama2	གཁའ་ཏེ་ཁ་བ་བདས་སྟུང་ན།	7	28 (4 times↑)	['▁', 'ག', 'ཁ', 'འ', '<0xE0>', '<0xBD>', '<0x8F>', 'ེ', 'ཏ', '<0xE0>', '<0xBD>', '<0x81>', 'ཁ', 'བ', 'འ', 'བ', 'ད', 'ས', 'ས', 'ྟ', 'ུ', 'ང', 'ན', '<0xE0>', '<0xBC>', '<0x8D>']
Llama3	གཁའ་ཏེ་ཁ་བ་བདས་སྟུང་ན།	7	43 (6 times↑)	['à½', 'Ĥ', 'à½', '£', 'à¼', 'Ï', 'à½', 'ı', 'à½', 'º', 'à¼', 'Ï', 'à½', 'ĝ', 'à¼', 'Ï', 'à½', 'ĸ', 'à¼', 'Ï', 'à½', 'ĸ', 'à½', 'ĸ', 'à½', '¦', 'à¼', 'Ï', 'à½', 'ĸ', 'à', '¾', '±', 'à½', '´', 'à½', 'Ĥ', 'à¼', 'Ï', 'à½', 'Ĵ', 'à¼', 'ĵ']

Figure 1: While we developed the speicalized LLM for Tibetan, existing LLM tokenizers such as those used in LLaMA2 and LLaMA3 severely over-segment Tibetan text. For an input of only seven words, LLaMA2 produces 28 tokens (a fourfold increase), while LLaMA3 generates 43 tokens (a sixfold increase). This excessive fragmentation inflates sequence length, increases computational and memory costs, and forces the model to operate on sub-symbolic fragments rather than linguistically meaningful units. As a result, semantic coherence is lost, and performance on downstream tasks such as spelling correction or translation is degraded. Beyond accuracy, this tokenization bias also exacerbates inefficiency and unfairness, as morphologically rich, low-resource languages like Tibetan face disproportionately higher computational burdens compared to high-resource languages.

struggle with either low robustness or prohibitive computational cost. These limitations make Tibetan spelling correction an ideal testbed for our framework: it requires *structure-aware representations*, yet demands *ultra-efficient inference* for real-world deployment in resource-constrained regions. By encoding syllable components directly as resistance states in a memristor crossbar, NeuSpell aligns naturally with the structural properties of Tibetan and provides a GPU-free solution to high-accuracy spelling error recognition.

In this paper, we make the following contributions:

- We introduce **NeuSpell**, a neuromorphic framework that learns *representations as resistance states* in a memristor crossbar array, offering a non-von Neumann alternative to token-based sequence modeling.
- We design a **structure-aware decomposition pipeline** that maps linguistic components into resistance-driven embeddings, enabling massively parallel, in-memory inference with ultra-low latency and near-zero power consumption.
- We construct and release **SSC-TiM**, the first component-level Tibetan medical corpus, and demonstrate that NeuSpell achieves state-of-the-art performance (**98.2% F1**) and efficiency, surpassing strong neural and large language models without relying on GPUs.

2 Related Work

2.1 Tibetan NLP

NLP for Tibetan has seen gradual progress but remains constrained by limited annotated resources Gao et al. (2025a;b); Huang et al. (2025) and the complexity of its syllabic structure Gao et al. (2025b). Early approaches relied heavily on rule-based systems or shallow neural architectures for segmentation, part-of-speech tagging, and spelling correction Wang & Duola (2008); Huang & Da (2010); Zhu et al. (2013); wangdui et al. (2014); Brown (2024); San et al. (2021); Jie et al. (2014); Zhu et al. (2014); Liu et al. (2017); Danzhaxi et al. (2020); Guo–cai–rang et al. (2021). More recent efforts introduced pretrained models for classification and error detection, but these systems inherit the inefficiencies of token-based representations, where the decomposition of syllables into long character sequences inflates input length and reduces robustness Liang et al. (2024); Khysru et al. (2021); Qi et al. (2024b;a). LLMs can partially address this through subword-level generalization Batsuren et al. (2022), yet their computational overhead and reliance on GPU acceleration make them impractical for deployment in resource-constrained environments. Overall, Tibetan NLP still lacks efficient, structure-aware approaches that align with the hierarchical nature of the language while remaining lightweight enough for real-world use.

2.2 MEMRISTOR

Memristors Chua (1971) have emerged as promising devices for in-memory computing, where storage and computation are co-located to overcome the bottleneck of von Neumann architectures. Their nonlinear resistance switching has already been exploited in neuromorphic circuits and vision tasks, demonstrating ultra-low power and high parallelism Jo et al. (2010b). More broadly, memristor-based designs have been considered for general-purpose learning systems, though their application to language processing remains largely unexplored Zhang et al. (2023); Jo et al. (2010a); Pershin (2019); Kahale & Tannir (2022). In the context of NLP, most existing methods continue to rely on digital architectures accelerated by GPUs, leaving an open question of how neuromorphic hardware can support sequence modeling San et al. (2021); Jie et al. (2014); Zhu et al. (2014); Hurst et al. (2024); Achiam et al. (2023). Our work contributes to this gap by reformulating Tibetan syllable recognition as a resistance-state computation problem, thereby integrating linguistic structure with the native properties of memristor crossbar arrays. This is the first work that introduces a neuromorphic framework for Tibetan syllable structure correction, eliminating the reliance on GPUs and token-based representations.

3 METHODOLOGY

According to the ideal memristor suggested by Chua Chua (1971), a time-invariant memristor can be represented as Equation 1 and Equation 2:

$$\frac{dw}{dt} = f(w, v) \tag{1}$$

$$I(t) = E(w, v) \cdot v(t) \tag{2}$$

Let w denote the internal state variable of the memristor, defined as the ratio of the doped region's thickness to the total device thickness. The voltage and current across the device are represented by $v(t)$ and $I(t)$, respectively, while $E(w, v)$ denotes the conductance, and $f(w, v)$ captures the dynamics of w. Under the cubic flux-controlled memristor model , the charge q is a single-valued function of the magnetic flux ϕ. Expanding $q(\phi)$ via Taylor series yields:

$$q(\phi) = \sum_{n=0}^{\infty} a_n \phi^n = \alpha\phi + \beta\phi^2 + \gamma\phi^3 + o\left(\phi^3\right) \tag{3}$$

where α, β and γ are constants. The device conductance $E(\phi)$ can be represented as Equation 4:

$$E(\phi) = \alpha + 2\beta\phi + 3\gamma\phi^2 \tag{4}$$

According to the relationship between current i, voltage v, magnetic flux ϕ, time t and device conductance $E(\phi)$, the flux ϕ and current i can be computed as Equation 5 and Equation 6:

$$i(t) = \left(\alpha + 2\beta\phi + 3\gamma\phi^2\right) v(t) \tag{5}$$

$$\phi(t) = \begin{cases} \phi_0 + \int_0^t v(\tau)d\tau, v \geq v_{pos} \text{ or } v \leq v_{neg} \\ \phi_0 + \int_0^{t-d\tau} v(\tau)d\tau, v_{neg} < v < v_{pos} \end{cases} \tag{6}$$

where v_{pos} and v_{neg} are threshold of magnetic flux. When voltage $v \in [v_{neg}, v_{pos}]$, the conductance of memristor is stable. Based on it, we develop the NeuSpell, shown in Figure 2. After inputting a syllable, its length is calculated and checked to ensure it falls within the valid range. Depending on the length, the syllable is evaluated against specific structural combinations of elements such as prefixes, root letters, subscripts, superscripts, vowels, suffixes, and feature sets, along with positional conditions that dictate component requirements. If no red parts are present, non-Tibetan clearing and tokenization are used to pre-process Tibetan text.

NeuSpell uses individual memristors at the intersections of word and bit lines, leveraging their unique properties . A voltage v_i adjusts the resistance of a target memristor, while a read voltage

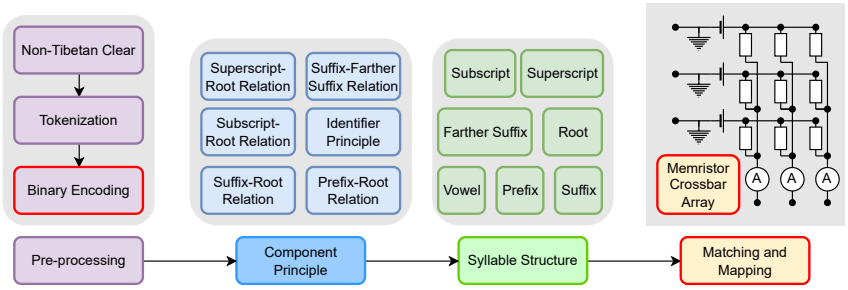


Figure 2: Diagram of the NeuSpell

v_{read}, below the v_{off} or v_{on} thresholds, prevents altering other memristors in the same row or column, which is used for parallel bitwise matching in Tibetan syllable component recognition.

To simplify processing, a single array matches all components. The array has 50 columns (30 consonants and 20 numerals, treating numerals as root letters) and encodes each Tibetan character into 8-bit binary, represented by 8 memristors per row. Thus, the array consists of 50×8 memristors. Based on Tibetan character, the binary codes of the three candidate characters are set to '1011001', '01010101', and '11101111', respectively. The binary code of the character being matched is '1011001'. According to the theory and experiment, the character should correctly match the first candidate character.

All in all, NeuSpell involves the following steps:

1) Sequentially select Tibetan characters from the syllables and identify candidate characters for mapping.

2) Apply voltage pulses v_{read} with magnitudes v_{off} and v_{on} on each row to represent the binary code of the target Tibetan character. A high voltage v_0 represents logical 0, while a low voltage v_1 represents logical 1.

3) Configure the candidate characters in the crossbar array by setting the memristor resistances R to R_{OFF} (logical 0) or R_{ON} (logical 1).

4) Place an ammeter at the end of each column to measure the column current I_{col_i} flowing through the memristors in the ith column.

5) For a match at the jth bit, the current I_{ij} through the corresponding memristors must satisfy:

$$I_{ij} = \frac{v_{read}}{R} = \frac{v_0}{R_{OFF}} = \frac{v_1}{R_{ON}} \tag{7}$$

6) If selected character matches ith in candidate characters, the current value in corresponding column should satisfy:

$$I_{col_i} = \sum_{j=1}^{8} \frac{v_{read}}{R} = 8 \times \frac{v_0}{R_{OFF}} \tag{8}$$

4 Dataset & Implementation

4.1 Dataset

4.1.1 SSC-TiM

We selected these two ancient books: rGyud bZhi[1] (46%) and Crystal Materia[2] (54%), because modern Tibetan, influenced by the Internet, has undergone significant changes, with its grammar

[1]ISBN: 7532305317
[2]ISBN: 9787225067841

becoming closer to Mandarin and losing many distinctive features of Tibetan culture and language. These 5000 text data contain long (51%, Appendix A Figure 8) and short sentences (49%, Appendix A Figure 7). And they all use ancient Tibetan, not Mandarin or vernacular. All data have been verified by Tibetan experts and their institutions. This is the first Tibetan medical dataset that preserves the ancient grammar of the Tibetan language, making it suitable for benchmarks in NLP tasks like machine translation and report generation. For more details, please refer to Appendix A.

4.1.2 TUSA

The Tibetan University Sentiment Analysis (TUSA) Zhang et al. (2024) containing 10,000 sentences, is used for pre-training transformer-based models in Tibetan sentiment analysis. For spelling correction, the training data is sourced from the Tibetan text corpus released by Tibet University , originally used for Tibetan news classification and comprising over 5.7 million sentences, this corpus serves as the basis for constructing the spelling correction training set. From this corpus, 50,000 high-quality sentences are selected. To assess model performance in real-world scenarios, an additional evaluation set of 1,000 erroneous sentences collected from the web is included.

4.2 Implementation

4.2.1 Hyperparameter

A total of 400 memristors are arranged in a crossbar array. The threshold voltages are configured with an off-state voltage (v_{off}) of 0.3 V and an on-state voltage (v_{on}) of -0.3 V. For binary encoding, the voltage levels are set as $v_0 = 0.2\,\text{V}$ and $v_1 = 0.08\,\text{V}$. The resistance values of the memristors are $R_{OFF} = 2500\,\Omega$ in the high-resistance state and $R_{ON} = 100\,\Omega$ in the low-resistance state. Other device parameters are defined as follows: the nonlinear ion drift coefficients are $\alpha_{off} = 1$ and $\alpha_{on} = 3$; the window function parameters are $k_{off} = 80$ and $k_{on} = -120$; and the internal state variable boundaries are set to $x_{off} = 0$ and $x_{on} = 1$. Finally, the number of training epochs is 5.

4.2.2 Simulation

Before the main experiments, we simulate the pattern-matching capability of a memristor crossbar array. A simplified task with three candidate characters is constructed using an 8×3 array of 24 memristors (Appendix B Figure 9), with ammeters recording column currents. Under a dynamic voltage $V(t) = A\sin(\omega t)$, the device exhibits typical I–V hysteresis (Appendix B Figure 11), where larger amplitudes enlarge the loop and higher frequencies compress it due to limited ion migration. When the voltage pattern of the correct character is applied, measured currents match theoretical values (Equations 7–8), confirming accurate recognition. Sudden current drops (Appendix B Figures 10a–10c) further demonstrate fast resistive switching, a key property for neuromorphic computing and memory.

4.2.3 Evaluation Matrix

We use Precision, Recall, and F1-score as evaluation metrics. Precision measures the proportion of correctly predicted positive samples among all predicted positives, while Recall reflects the proportion of correctly predicted positives among all actual positives. The F1-score, as the harmonic mean of Precision and Recall, provides a balanced assessment, particularly useful in scenarios with imbalanced datasets. The best results are highlighted in bold, and the second-best are underlined.

5 Experimental Result

5.1 Experimental Setup

As noted earlier, a memristor's resistance can represent logical 0 or 1, enabling binary operations. To utilize memristors effectively, Tibetan characters must be converted into binary form. In Unicode, Tibetan characters are assigned hexadecimal codes from 0F00 to 0FDA. We extract the last two digits of these codes and convert them into 8-bit binary. Figure 3 illustrates the binary encoding for consonants.

Character	Code	Character	Code	Character	Code	Character	Code	Character	Code
ཀ	01000000	ཁ	01000001	ག	01000010	ང	01000100	ཅ	01000101
ཆ	01000110	ཇ	01000111	ཉ	01001001	ཏ	01001111	ཐ	01010000
ད	01010001	ན	01010011	པ	01010100	ཕ	01010101	བ	01010110
མ	01011000	ཙ	01011001	ཚ	01011010	ཛ	01011011	ཝ	01011101
ཞ	01011110	ཟ	01011111	འ	01100000	ཡ	01100001	ར	01100010
ལ	01100011	ཤ	01100100	ས	01100110	ཧ	01100111	ཨ	01101000

Figure 3: Binary code of consonant.

The abbreviations for the components of Tibetan characters are as follows: Pre.=prefix letter; Sup.=superscript letter; Ro.=root letter; Sub=subscript letter; Vow.=vowel; Suf.=suffix letter; FS.=farther suffix letter.

5.2 Baseline Model

5.2.1 Lightweight Neural Model

We choose RNN Li et al. (2018), LSTM Hochreiter & Schmidhuber (1997), CNN-LSTM Vinyals et al. (2015b), CNN-RNN Vinyals et al. (2015a), Bi-RNN Schuster & Paliwal (1997), Bi-LSTM Schuster & Paliwal (1997); Hochreiter & Schmidhuber (1997), BERT Vaswani et al. (2017); Devlin et al. (2019), RoBERTa Liu et al. (2019), Transformer Vaswani et al. (2017), Mamba Gu & Dao (2023), and CINO Yang et al. (2022) as baseline models on the check task, optimized by the tool Optuna[3] . These models are trained and tested using ten-fold cross validation, with training epochs set to 500.

5.2.2 Large Language Model

As shown in Appendix C Table 7, we evaluate several open-source models, including Qwen-2.5 Bai et al. (2023a;b; 2024); Yang et al. (2024), DeepSeek Guo et al. (2025); Liu et al. (2024), and the LLaMA-3.1 series (8B, 70B, 405B) Grattafiori et al. (2024), via their official APIs or platforms. We also include closed-source systems such as the GPT family Hurst et al. (2024); Achiam et al. (2023), Claude-3.5-Sonnet Anthropic (2024), and Gemini-1.5-Flash Team (2024). For open-source models, we apply parameter-efficient fine-tuning with LoRA Hu et al. (2022) and supervised fine-tuning Ouyang et al. (2022), while proprietary models are adapted through prompt-based tuning and instruction engineering. To further assess representation choices, we compare token-based, subword-based, and hybrid input strategies. Token-based inputs provide semantic granularity but suffer from out-of-vocabulary issues. Subword segmentation improves coverage and generalization. The hybrid approach combines both, balancing semantic integrity with robustness, which is particularly beneficial for morphologically rich languages such as Tibetan.

5.2.3 Hardware

The hardware specifications for training and testing them include 2 Tesla V100 GPUs (2 x 32GB), 64GB Of RAM, 8 CPU cores per node, and a total of 6 nodes.

5.3 Comparative Experiment

As shown in Table 1, pretrained models such as RoBERTa and Mamba achieve competitive results, with underlined values denoting the second-best scores. NeuSpell consistently outperforms all baselines, reaching 100% Precision and the highest F1-scores on both datasets, thereby establishing its effectiveness in Tibetan spelling correction. We further conducted efficiency experiments across models. While conventional approaches require extensive data conversion and approximately two days for training, testing, and deployment, NeuSpell directly encodes Tibetan syllables as resistance states, eliminating these overheads and enabling a significantly more efficient pipeline.

As shown in Figure 4a, NeuSpell demonstrates stable performance as the sample size increases, achieving 100% precision, 98.6% recall, and 98.2% F1-score, which highlights its high accuracy in detecting syllable errors. We further evaluate 1,312 high-frequency syllables, including 200 with

[3]https://optuna.org/

Table 1: Comparison with NLP Models of Check Results, Inference Time & GPU Utilization

Model	SSC-TiM			TUSA			Efficiency & Hardware Utilization Statistics			
	Precision	Recall	F1-score	Precision	Recall	F1-score	Inference Time (ms)	Parameters (M)	FLOPs	GPU Utilization
RNN	96.01	95.89	95.56	97.14	97.54	96.73	18 ms	11M	87	Low
LSTM	96.07	96.01	95.87	97.22	96.52	96.42	22 ms	14M	115	Low
CNN-RNN	96.23	96.08	95.99	96.97	96.18	96.57	35 ms	20M	184	Medium
CNN-LSTM	97.03	97.11	96.96	97.39	96.88	97.08	50 ms	25M	254	Medium
Bi-RNN	97.39	97.61	97.13	97.51	96.93	97.22	40 ms	28M	302	Medium
Bi-LSTM	97.39	97.61	97.13	97.24	96.72	96.97	55 ms	32M	350	High
BERT-char	95.93	96.01	95.98	96.32	95.87	96.09	67 ms	66M	450	High
BERT	97.20	97.95	97.10	97.59	97.16	97.37	76 ms	110M	1164	Very High
Transformer	97.52	98.24	97.17	97.46	97.03	97.24	112 ms	125M	1800	Very High
Mamba	97.13	97.89	96.93	98.49	98.20	98.34	108 ms	113M	1760	High
CINO	97.99	98.01	97.01	98.17	97.88	98.02	134 ms	150M	2143	Very High
RoBERTa	99.53	98.51	97.16	98.82	97.64	98.22	126 ms	355M	2379	High
NeuSpell	**100.0**	**98.58**	**98.20**	**100.0**	**99.71**	**99.35**	**0.07 ms**	**0M**	**0.0045**	None

(a) Statistics of Check Result

(b) Statistics of Recognition Accuracy

Figure 4: Performance Details of NeuSpell on SSC-TiM

two components, 300 each with three, four, and five components, 200 with six components, and 12 with seven components. As shown in Figure 4b, NeuSpell performs best on three- and four-component syllables, with only a slight decrease in accuracy for seven-component syllables. Overall, the average accuracy remains consistently high at around 98.5

Table 2: Check Results & Inference time of LLMs

LLM	Version	Input	SSC-TiM			TUSA			Inference Time	
			Precision	Recall	F1-score	Precision	Recall	F1-score	Single Word	Multi-word (5)
Claude	3-5-sonnet	Token	63.27	63.07	62.79	61.74	61.08	60.97	650ms	1020ms
		Subword	74.37	73.87	73.51	77.72	77.14	76.93	670ms	1060ms
		Hybird	77.32	76.91	76.58	79.35	78.88	78.65	700ms	1100ms
Gemini	1.5-flash	Token	67.24	66.85	66.57	68.96	68.12	67.78	480ms	750ms
		Subword	69.34	68.91	68.62	69.58	68.97	68.72	500ms	790ms
		Hybird	71.24	70.78	70.45	72.39	71.85	71.61	530ms	830ms
DeepSeek	V3	Token	78.96	78.42	78.19	79.47	78.66	78.31	320ms	520ms
		Subword	83.17	82.55	82.31	84.93	84.21	83.86	340ms	550ms
		Hybird	86.29	85.79	85.54	86.18	85.63	85.41	370ms	580ms
	R1	Token	78.15	77.53	77.20	77.28	76.63	76.28	300ms	490ms
		Subword	79.36	78.74	78.41	78.15	77.56	77.21	320ms	510ms
		Hybird	79.29	78.61	78.26	78.27	77.61	77.28	350ms	540ms
LlaMA	3.1-8B	Token	69.42	68.77	68.41	70.15	69.53	69.12	180ms	290ms
		Subword	71.35	70.82	70.45	72.04	71.47	71.11	190ms	310ms
		Hybird	73.28	72.67	72.30	74.12	73.44	73.05	210ms	330ms
	3.1-70B	Token	78.05	77.39	77.01	78.61	77.98	77.59	370ms	600ms
		Subword	80.44	79.87	79.48	81.02	80.36	79.91	390ms	630ms
		Hybird	82.11	81.56	81.18	82.65	82.04	81.67	420ms	670ms
	3.1-405B	Token	83.18	82.63	82.25	83.72	83.14	82.75	500ms	820ms
		Subword	85.07	84.51	84.12	85.56	84.95	84.61	520ms	850ms
		Hybird	86.35	85.76	85.38	86.79	86.15	85.79	550ms	890ms
Qwen	2.5-7b	Token	66.48	65.93	65.57	67.12	66.41	66.03	190ms	300ms
		Subword	68.32	67.79	67.45	69.08	68.46	68.15	200ms	320ms
		Hybird	70.05	69.51	69.16	70.92	70.25	69.84	220ms	340ms
	2.5-32b	Token	72.38	71.82	71.44	73.05	72.42	72.05	330ms	520ms
		Subword	74.27	73.71	73.32	74.98	74.37	74.01	350ms	550ms
		Hybird	75.83	75.29	74.92	76.42	75.79	75.44	370ms	580ms
	2.5-72b	Token	76.41	75.87	75.52	77.04	76.41	76.05	460ms	740ms
		Subword	78.63	78.06	77.71	79.12	78.53	78.19	480ms	770ms
		Hybird	80.07	79.51	79.14	80.61	80.01	79.67	510ms	810ms
GPT	3.5-turbo	Token	56.35	55.68	55.36	57.89	56.95	56.40	420ms	680ms
		Subword	59.97	59.21	58.83	61.42	60.74	60.30	440ms	710ms
		Hybird	61.24	60.49	60.09	62.75	62.03	61.60	470ms	740ms
	4O	Token	78.97	78.13	77.68	80.12	79.24	78.89	550ms	880ms
		Subword	83.41	82.57	82.06	84.36	83.61	83.18	570ms	910ms
		Hybird	85.63	84.87	84.40	86.28	85.71	85.33	600ms	950ms
	O1-mini	Token	41.27	40.63	40.29	42.33	41.55	41.14	290ms	460ms
		Subword	42.33	41.72	41.37	43.45	42.70	42.26	310ms	480ms
		Hybird	46.95	46.21	45.75	47.96	47.18	46.81	330ms	500ms
	O1	Token	39.84	39.10	38.72	41.01	40.20	39.76	320ms	510ms
		Subword	42.34	41.68	41.27	43.33	42.61	42.18	340ms	540ms
		Hybird	43.99	43.21	42.78	45.02	44.20	43.79	370ms	570ms
	O3	Token	53.14	52.31	51.88	54.66	53.84	53.39	430ms	700ms
		Subword	55.67	54.93	54.48	57.18	56.45	56.01	450ms	730ms
		Hybird	56.02	55.26	54.84	57.53	56.79	56.36	480ms	770ms
	NeuSpell		**100.0**	**98.58**	**98.20**	**100.0**	**99.71**	**99.35**	0.07ms	0.16ms

As shown in Table 2, we measured inference time on single Tibetan words and five-word phrases. NeuSpell achieved markedly faster speeds than LLMs by encoding weights directly at the physical level, integrating computation and storage, and avoiding data conversion and multi-layer processing. Among LLMs, results were similar across the LlaMA, GPT, and DeepSeek series, with LlaMA-3.1-405B ranking second overall. Also, for the recognition of syllable, as shown in Figure 5, NeuSpell still leads the rest of the LLMs with the highest score.

(a) LLMs (Token)

(b) LLMs (Subword)

Figure 5: Tibetan Syllable Component Recognization: Memristor v.s. LLMs

5.4 Ablation Experiment

To evaluate the contribution of each module in *NeuSpell* and the impact of physical parameters on performance, we conduct a series of ablation experiments in three categories: (1) Structural Component Removal; (2) Hardware-level Parameter Variation and (3) Syllable-level Input Reduction.

5.4.1 Component-Level Ablation

we first investigate the impact of key structural alignment rules on recognition accuracy. The results are summarized in Table 3.

Table 3: Component-wise ablation of the structural pipeline.

Model Variant	Precision	Recall	F1-Score
No Prefix-Root Check	98.42	96.18	97.28
No Component Legality Check	96.25	95.73	95.98
Random Binary Assignment	71.84	68.52	70.14
NeuSpell	100.0	98.58	98.20

Removing prefix-root alignment and component legality constraints both led to performance degradation, with random binary mapping completely destroying structural integrity.

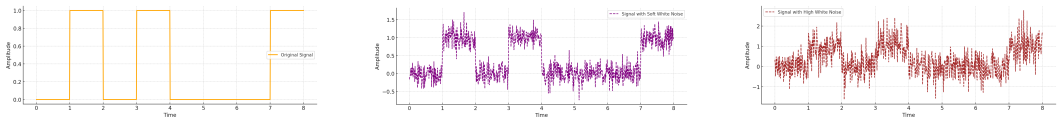
5.4.2 Memristor Parameter Sensitivity

we evaluate how physical memristor parameters affect recognition accuracy and current stability. Specifically, we vary the on/off resistance states and threshold voltages.

Table 4: Ablation of Hardware Parameters

Parameter Variant	F1-Score	Noise Level	Failure Mode
$R_{\text{ON}} = 100\Omega$ (default)	98.20	Low	Stable
$R_{\text{ON}} = 150\Omega$	95.66	Moderate	Current Drift
No v_{off} Threshold	93.48	High	Spurious Activation
Low Ambient Temperature	96.72	Low	Delayed Switching

We observe that incorrect voltage thresholds or resistance mismatches significantly increase current noise and lead to recognition instability. Figure 6 illustrates three forms of current trajectories for the binary pattern `01010001` under different resistance noise levels.



(a) Ideal (b) Soft (c) Strong

Figure 6: Three forms of current of 01010001. (The corresponding letter can be found in Figure 3)

5.4.3 Syllable Component Reduction

To assess the necessity of full syllable decomposition, we run NeuSpell on subsets of components. Results are shown in Table 5.

Table 5: Recognition Performance with Reduced Syllable Components

Input	Precision	Recall	F1-Score
Full 7-component Syllables	100.0	98.58	98.20
Without Farther Suffix	100.0	96.01	97.94
Prefix Only	93.24	89.51	91.34
Root + Vowel Only	90.12	86.74	88.39

These results confirm that both structural alignment and full syllable representation are essential to achieving state-of-the-art recognition in Tibetan.

5.5 Supplementary Experiment

To further assess robustness, we compare NeuSpell with a traditional rule-based method for Tibetan syllable validation (Appendix E). As shown in Table 6, the rule-based approach achieves perfect accuracy on clean, structurally valid syllables but fails on handwritten data due to its sensitivity to variations and lack of error tolerance. To reflect real-world conditions, we constructed an additional test set of messy handwritten syllables. On this benchmark, NeuSpell maintains strong performance with an F1-score of 97.09%, substantially outperforming the rule-based baseline.

Table 6: Performance Comparison between Rule-based method and NeuSpell

Method	Standard: SSC-TiM			Handwriting		
	Precision	Recall	F1-Score	Precision	Recall	F1-Score
Rule-Based	100.0	100.0	100.0	15.63	9.82	12.00
NeuSpell	100.0	98.58	98.20	97.28	97.13	97.09

6 Conclusion

We introduced NeuSpell, a neuromorphic framework for syllable structure recognition that eliminates dependence on GPUs and conventional von Neumann architectures. By encoding linguistic components as resistance patterns within a memristor crossbar array, the system achieves parallel, hardware-native inference without relying on digital tokenization or sequential pipelines. To support evaluation, we developed SSC-TiM, a component-level corpus that includes both printed and handwritten Tibetan inputs. Experiments demonstrate that NeuSpell delivers near-perfect accuracy on controlled data and strong robustness on handwritten samples, surpassing both rule-based systems and human annotators. These results highlight a new direction for structure-aware AI, showing that neuromorphic computation can provide an efficient, GPU-free alternative for real-world, low-resource language scenarios.

7 Limitation

While NeuSpell shows strong performance, it has several limitations. Our experiments focus on Tibetan syllables, so generalization to other languages remains untested. The memristor simulation is limited to small-scale arrays and idealized settings, which may differ from real hardware. Moreover, deployment within practical NLP systems will require further integration efforts. Future work will extend evaluation to multilingual tasks and larger physical devices.

Ethics Statement

This work focuses on Tibetan syllable structure correction using a neuromorphic framework. The datasets used are collected with proper consent, containing no personally identifiable or sensitive information. Our approach is intended to support low-resource language processing and cultural preservation. Potential risks include biased performance when applied to other languages or misuse of technology in unintended contexts. We encourage responsible use and emphasize that further evaluation is necessary before deployment in real-world applications.

Reproducibility Statement

We provide detailed descriptions of our datasets, model architectures, training procedures, and evaluation metrics in the main paper and appendix. The SSC-TiM dataset introduced in this work, along with preprocessing scripts, will be released upon publication. Implementation details, including hyperparameters, optimization settings, and hardware specifications, are documented in Appendix C. The memristor crossbar simulations are fully described in Appendix B with corresponding equations and parameters. These resources ensure that all reported results can be independently reproduced.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

Anthropic. Claude 3.5 sonnet model card addendum. `https://www.anthropic.com/news/claude-3.5-sonnet`, 2024.

Aman Arya, Anas Awadalla, Rohan Bhalerao, Daniil Belenko, Stella Biderman, Weijia Chen, Subhabrata Dey, Rishubh Dey, Nelson Elhage, Leo Gao, et al. Aya model: An instruction finetuned open-access multilingual language model. *arXiv preprint arXiv:2402.07827*, 2024.

Junjie Bai, Yuxiao Bai, Yu Chen, Yuxiao Gu, Jian Jin, Xiaodong Li, Zheng Lin, Zihan Liu, Yong Lu, Kun Sun, et al. Qwen technical report. *arXiv preprint arXiv:2309.16609*, 2023a.

Yuxiao Bai, Junjie Bai, Yu Chen, Yuxiao Gu, Jian Jin, Xiaodong Li, Zheng Lin, Zihan Liu, Yong Lu, Kun Sun, et al. Qwen-1.5: A scalable and accessible large language model family. *arXiv preprint arXiv:2310.18112*, 2023b.

Yuxiao Bai, Junjie Bai, Yu Chen, Yuxiao Gu, Jian Jin, Xiaodong Li, Zheng Lin, Zihan Liu, Yong Lu, Kun Sun, et al. Qwen2 technical report. *arXiv preprint arXiv:2407.10671*, 2024.

Khuyagbaatar Batsuren et al. Are subwords enough? tokenization in morphologically rich languages. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pp. 10200–10215, 2022.

Kaj Bostrom and Greg Durrett. Byte pair encoding is suboptimal for language model pretraining. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 461–466, 2020.

Collin J. Brown. Improved neural word segmentation for standard Tibetan. In Atul Kr. Ojha, Sina Ahmadi, Silvie Cinková, Theodorus Fransen, Chao-Hong Liu, and John P. McCrae (eds.), *Proceedings of the 2nd Workshop on Resources and Technologies for Indigenous, Endangered and Lesser-resourced Languages in Eurasia (EURALI) @ LREC-COLING 2024*, pp. 12–17, Torino, Italia, May 2024. ELRA and ICCL.

Leon Chua. Memristor—the missing circuit element. *IEEE Transactions on Circuit Theory*, 18(5): 507–519, 1971.

HUA Danzhaxi, CAI Zhijie, and BAN Mabao. A tc_lstm based method for tibetan spelling check. 34:50–55, 2020.

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*, pp. 4171–4186, 2019.

Fan Gao, Cheng Huang, Nyima Tashi, Yutong Liu, Xiangxiang Wang, Thupten Tsering, Ban Ma-bao, Renzeg Duojie, Gadeng Luosang, Rinchen Dongrub, et al. Tibstc-cot: A multi-domain instruction dataset for chain-of-thought reasoning in language models. *arXiv preprint arXiv:2508.01977*, 2025a.

Fan Gao, Cheng Huang, Nyima Tashi, Xiangxiang Wang, Thupten Tsering, Ban Ma-bao, Renzeg Duojie, Gadeng Luosang, Rinchen Dongrub, Dorje Tashi, et al. Tlue: A tibetan language understanding evaluation benchmark. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2025b.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

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

Dongsheng Guo, Deyi Yang, Hao Zhang, Jie Song, Rui Zhang, Ran Xu, Qi Zhu, Shizhe Ma, Peng Wang, Xiang Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

HUA Guo–cai–rang, Secha Jia, BAN Ma–bao, and CAI Rang–jia. Error correction of tibetan verbs based on deep learning. In *2021 2nd International Conference on Artificial Intelligence and Computer Engineering (ICAICE)*, pp. 221–226, 2021.

Song Han, Xingyu Liu, Huizi Mao, Jing Pu, Ardavan Pedram, Mark Horowitz, and William Dally. Eie: Efficient inference engine on compressed deep neural network. In *Proceedings of the 43rd Annual International Symposium on Computer Architecture (ISCA)*, pp. 243–254, 2016.

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

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2022.

Cheng Huang, Fan Gao, Yutong Liu, Nyima Tashi, Xiangxiang Wang, Thupten Tsering, Ma bao Ban, Renzeg Duojie, Gadeng Luosang, Rinchen Dongrub, Dorje Tashi, Xiao Feng, Hao Wang, and Yongbin Yu. Tib-stc: A large-scale structured tibetan benchmark for low-resource language modeling. 2025.

Heming Huang and Feipeng Da. Discussion on collation of tibetan syllable. In *2010 International Conference on Asian Language Processing*, pp. 35–38, 2010.

Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, et al. Gpt-4o system card. *arXiv preprint arXiv:2410.21276*, 2024.

Zhu Jie, Li Tianrui, and Liu Shengjiu. The algorithm of spelling check based on tsrm. 28:92–98, 2014.

Sung Hyun Jo, Ting Chang, Idongesit Ebong, Bhavitavya B Bhadviya, Pinaki Mazumder, and Wei Lu. Nanoscale memristor device as synapse in neuromorphic systems. *Nano letters*, 10(4):1297–1301, 2010a.

Sung Hyun Jo, Ting Chang, Ifeanyi Ebong, Bharat P. Bhadviya, Pinaki Mazumder, and Wei Lu. Nanoscale memristor device as synapse in neuromorphic systems. *Nano Letters*, 10(4):1297–1301, 2010b.

Norman Jouppi et al. In-datacenter performance analysis of a tensor processing unit. In *Proceedings of the 44th Annual International Symposium on Computer Architecture (ISCA)*, pp. 1–12, 2017.

Theresa Kahale and Dani Tannir. Memristor modeling using the modified nodal analysis approach. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 41(4):1191–1195, 2022. doi: 10.1109/TCAD.2021.3068098.

Kuntharrgyal Khysru, Di Jin, Yuxiao Huang, Hui Feng, and Jianwu Dang. A tibetan language model that considers the relationship between suffixes and functional words. *IEEE Signal Processing Letters*, 28:459–463, 2021.

Taku Kudo and John Richardson. Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pp. 66–71, 2018.

Shuai Li, Wanqing Li, Chris Cook, Ce Zhu, and Yanbo Gao. Independently recurrent neural network (indrnn): Building a longer and deeper rnn. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5457–5466. IEEE, 2018.

Yatao Liang, Hui Lv, Yan Li, La Duo, Chuanyi Liu, and Qingguo Zhou. Tibetan-bert-wwm: A tibetan pretrained model with whole word masking for text classification. *IEEE Transactions on Computational Social Systems*, 11(5):6268–6277, 2024.

An Liu, Bo Feng, Bo Xue, Bin Wang, Bo Wu, Chao Lu, Chen Zhao, Cheng Deng, Chen Zhang, Cheng Ruan, et al. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*, 2024.

Huidan Liu, Jinling Hong, Minghua Nuo, and Jian Wu. Statistics and analysis on spell errors of tibetan syllables based on a large scale web corpus. 31:61–70, 2017.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1093–1104, 2019.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *arXiv preprint arXiv:2203.02155*, 2022.

Yuriy V. Pershin. A demonstration of implication logic based on volatile (diffusive) memristors. *IEEE Transactions on Circuits and Systems II: Express Briefs*, 66(6):1033–1037, 2019. doi: 10.1109/TCSII.2018.2873635.

Jinhu Qi, Shuai Yan, Wentao Zhang, Yibo Zhang, Zirui Liu, and Ke Wang. Research on tibetan tourism viewpoints information generation system based on llm. In *2024 12th International Conference on Intelligent Computing and Wireless Optical Communications (ICWOC)*, pp. 35–41. IEEE, 2024a.

Jinhu Qi, Shuai Yan, Yibo Zhang, Wentao Zhang, Rong Jin, Yuwei Hu, and Ke Wang. Rag-optimized tibetan tourism llms: Enhancing accuracy and personalization. *arXiv preprint arXiv:2408.12003*, 2024b.

Maocuo San, Zhijie Cai, Rangzhuoma Cai, and Jizhaxi Dao. Analysis on types of spelling errors in true tibetan characters. *MATEC Web of Conferences*, 336:06019, 01 2021. doi: 10.1051/matecconf/202133606019.

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

Vivienne Sze, Yu-Hsin Chen, Tien-Ju Yang, and Joel Emer. Efficient processing of deep neural networks: A tutorial and survey. *Proceedings of the IEEE*, 105(12):2295–2329, 2017.

Google DeepMind Gemini Team. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. Technical Report / Model Card, 2024. Available online.

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*, 2023a.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023b.

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.

Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. Show and tell: A neural image caption generator. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3156–3164, Boston, MA, USA, 2015a. IEEE. doi: 10.1109/CVPR.2015.7298935.

Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. Show and tell: A neural image caption generator. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3156–3164, 2015b.

Fucheng Wan and Xiangzhen He. Tibetan syntactic parsing based on syllables. In *International Congress of Mathematicans*, 2015.

Weilan Wang and Duola. Formation of standard tibetan syllables and comparison as well as analysis of the statistical results. In *2008 IEEE Conference on Cybernetics and Intelligent Systems*, pp. 379–384, 2008.

Bianba wangdui, Zhuoga, Chen Yanli, and Wu Qiang. Study on recog nition algorithms for tibetan construction elements. 49:104–111, 03 2014.

An Yang, Bo Yang, Bo Zhang, Bo Hui, Bin Zheng, Bowen Yu, Chengyu Li, Deren Liu, Fei Huang, Hongyi Wei, et al. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2024.

Sen Yang, Yiming Cui, Zhiyang Chen, Wanxiang Che, and Ting Liu. Cino: A pre-trained language model for chinese minority languages. In *Proceedings of the 29th International Conference on Computational Linguistics (COLING)*, pp. 1574–1586, 2022.

Jing Zhang, Hang Ren, Jin Yang, Nuo Qun, and Shengqiao Ni. Sentiment analysis of tibetan short texts based on graph neural network with keyphrases integration. pp. 474–480, 07 2024. doi: 10.1109/PRML62565.2024.10779689.

Linghua Zhang, Jingjin Zhu, and Lumu Man. Eastward development of tibetans in china: A gravity-based approach with an application to population. In *Proceedings of the 2021 4th International Conference on Geoinformatics and Data Analysis*, ICGDA '21, pp. 1–6, New York, NY, USA, 2021. Association for Computing Machinery. ISBN 9781450389341.

Wenbin Zhang, Peng Yao, Bin Gao, Qi Liu, Dong Wu, Qingtian Zhang, Yuankun Li, Qi Qin, Jiaming Li, Zhenhua Zhu, et al. Edge learning using a fully integrated neuro-inspired memristor chip. *Science*, 381(6663):1205–1211, 2023.

J. Zhu, T. Li, S. Ge, Q. Ren, and S. Qiao. Tibetan syllable rule model and applications. 49:68–74, 01 2013.

J. Zhu, T. Li, and S. Liu. An approach for tibetan text automatic proofreading and its system design. 50:142–148, 01 2014. doi: 10.13209/j.0479-8023.2014.001.

A Appendix A: Details of SSC-TiM

A.1 rGyud bZhi

rGyud bZhi, also called "Four Medical Tantras," is a foundational classic of Tibetan medicine, written in classical Tibetan with rhymed, obscure sentences suited to its rhythm. It consists of four parts: rTsa rGyud (6 chapters), covering human physiology, pathology, diagnosis, and treatment from a Tibetan medical philosophy perspective; bShad rGyud (31 chapters), elaborating on human anatomy, causes of illness, health maintenance, drug properties, diagnostic methods, and treatment principles; Man ngag rGyud (92 chapters), focusing on specific treatments for various clinical diseases; and Phyi ma rGyud (27 chapters), detailing diagnostic techniques, drug formulations, and external therapies.

A.2 Crystal Materia

Crystal Materia is divided into two parts: the upper part, written in verse, summarizes the effects of each medicine, while the lower part, written in narrative text, details the source, production environment, nature, taste, effects, and precautions of each drug. Although the rGyud bZhi theoretically proposed six flavors, eight properties, and seventeen effects without associating them with specific drugs, Crystal Materia Medica provides concrete details on each medicine, enriching Tibetan pharmacology and guiding its practical application. Together, these two works complement each other.

Language	Corpus
Tibetan	ཁྱི་ཀྲུགས་མགྲིན་བཀག་ཀླི་སྟོང་ཐུར་མ་འཕང་།
Chinese	扳起颈项在空嗓穿刺
Tibetan	རྣག་བཅད་ཆུག་པ་དབྱུང་དང་ཐྲུང་གསོོར་གཞུག།
Chinese	清热、干黄水、消肿
Tibetan	ཀུ་ཏྲའི་ previous རྒྱ་པས་གདོན་ནད་སེལ།
Chinese	乌鸦肉具治疗魔病的功效。

Figure 7: Details of SSC-TiM (Short Sentences)

Language	Corpus
Tibetan	ཕྱེ་མ་ལའང་སེ་འབྲུ་དང་ད་ཚིས་ཀྱིས་གཅོ་བྱས་པ་དང་ཞལ་སྨན་ཀྱི་སྟོང་བ་སྨྱར་ལ།
Chinese	散剂用石榴和小叶杜鹃为主的散剂和灰药的方药内服

Figure 8: Details of SSC-TiM (Long Sentences)

B Appendix B: Details of Simulation

B.1 Memristor Crossbar Array

The binary codes of the three candidate characters are set to '1011001', '01010101', and '11101111', respectively. The binary code of the character being matched is '1011001'. According to the theory and experiment, the character should correctly match the first candidate character, shown in Figure 9.

B.2 Memristor Property

Figure 10 presents the current vs. time behavior recorded by three different ampere meters. In the Figure 10a, Figure 10b and Figure 10c, all three graphs display a sudden drop in current at a

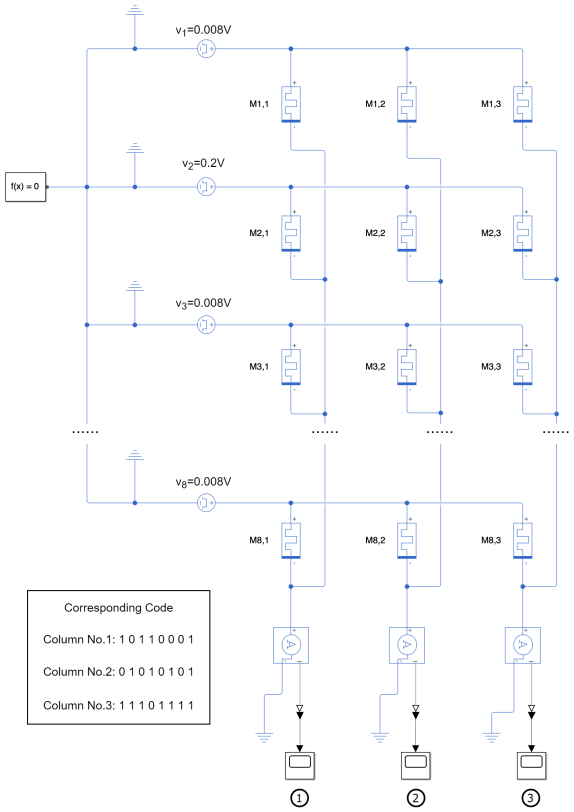
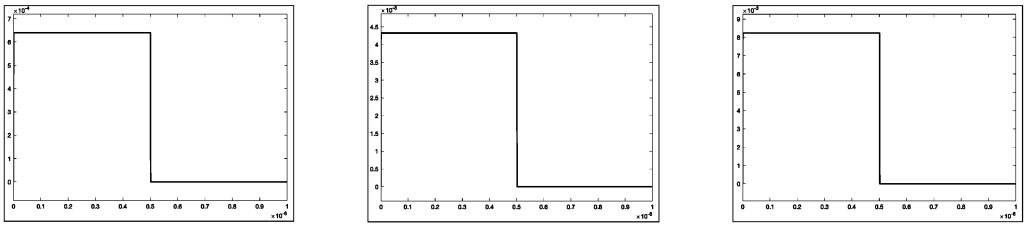


Figure 9: Circuit diagram of a memristor crossbar array, with the binary codes of candidate characters displayed in the lower-left corner.

critical moment, which is indicative of the memristor's rapid switching behavior. The consistency across these three measurements suggests that the memristor undergoes a sharp transition in current as it switches between different states. This switching characteristic is fundamental to memristor operation and is essential for applications such as synaptic simulation, non-volatile memory storage, and neuromorphic computing.

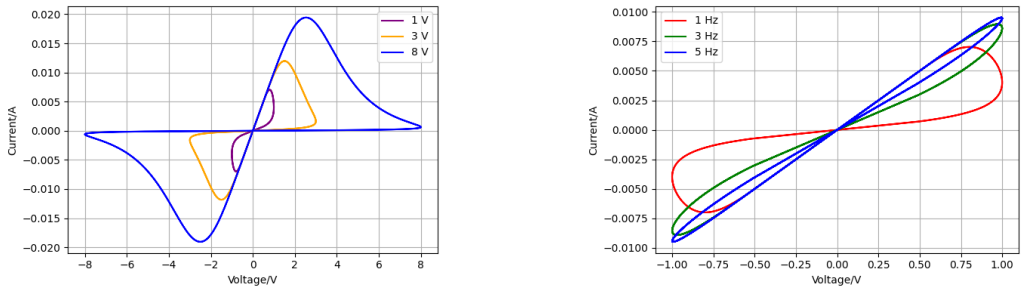


(a) The 1st Column (b) The 2nd Column (c) The 3rd Column

Figure 10: Current vs. Time Graph of 3 Ampere Meter

As shown in Figure 11, it shows the voltage-current (I-V) characteristics of the memristor under different conditions. In the Figure 11a, the I-V curves are shown for different voltage amplitudes (1 V, 3 V, and 8 V), where higher voltage amplitudes result in a larger hysteresis loop. This indicates that increasing the voltage amplitude enhances the nonlinear behavior of the memristor, activating stronger internal charge redistribution and resistive switching. In the Figure 11b, the I-V curves are plotted for different frequencies (1 Hz, 3 Hz, and 5 Hz), where the area of the hysteresis loop decreases as the frequency increases. This suggests that at higher frequencies, the memristor's

resistive switching weakens and approaches linear behavior due to the inability of ion migration to keep pace with the rapid changes in the electric field.



(a) Different frequency ω

(b) Different amplitude A

Figure 11: Voltage-ampere Characteristic Curve of Memristor

C Appendix C: Hyperparameters of Baseline Model

Table 7 summarizes the hyperparameter settings for all large language models evaluated in our experiments. Each model is configured with specific values for temperature, top-p, and streaming mode. For example, most models use fixed temperature values (e.g., 1.0 for Claude-3-5-sonnet, GPT-3.5-turbo, and GPT-4o, 0.6 for the LlaMA-3.1 family), while top-p varies across models (e.g., 0.95 for Gemini-1.5-flash, 0.9 for LlaMA-3.1). Streaming mode is disabled in most cases, except for certain GPT variants (O1-mini, O1, O3) and DeepSeek-R1, where it is enabled. These standardized configurations ensure a consistent and transparent comparison across different LLM baselines.

LLM	Temperature	Top_p	Stream
Claude-3-5-sonnet	1.0	None	False
Gemini-1.5-flash	None	0.95	False
DeepSeek-V3	1.0	None	False
DeepSeek-R1	1.0	None	True
LlaMA-3.1-8B	0.6	0.9	False
LlaMA-3.1-70B	0.6	0.9	False
LlaMA-3.1-405B	0.6	0.9	False
Qwen-2.5-7b	0.7	0.8	False
Qwen-2.5-32b	0.7	0.8	False
Qwen-2.5-72b	0.7	0.8	False
GPT-3.5-turbo	1.0	1.0	False
GPT-4O	1.0	1.0	False
GPT-O1-mini	1.0	1.0	True
GPT-O1	1.0	1.0	True
GPT-O3	1.0	1.0	True

Table 7: Hyperparameters of LLM

D Appendix D: Experiental Results

Syllables with the four modern Tibetan vowels were accurately recognized in Figure 12, while those with Sanskrit-transliterated vowels and identifiers were not. This limitation is due to the algorithm recognizing only modern Tibetan vowels. Expanding the vowel set to include Sanskrit-transliterated vowels would resolve this issue.

Syllable	Pre.	Sup.	Ro.	Sub.	Vow.	Suf.	FS.
སཁམས	ས	-	ཁ	-	-	ས	-
མང	-	-	ང	-	-	ང	-
གཱི	-	-	ག	-	◌ཱི	-	-
ཤུལ	-	-	ཤ	-	◌ུ◌	ལ	-
ལས	-	-	ལ	-	-	ས	-
ཀྱང	-	-	ཀ	◌ྱ◌	-	ང	-
ཤུལ	-	-	ཤ	-	◌ུ◌	ལ	-
ཁམས	-	-	ཁ	-	-	ས	ས
གཞན	ག	-	ཞ	-	-	ན	-
དུ	-	-	ད	-	◌ུ◌	-	-
མཆོད	མ	-	ཆ	-	◌ོ◌	ད	-
ཡ	-	-	ཡ	-	-	-	-
ཐོབ	-	-	ཐ	-	◌ོ◌	བ	-
ཤོར	-	-	ཤ	-	◌ོ◌	ར	-
ཤུ	-	-	ཤ	-	◌ུ◌	-	-
དུ	-	-	ད	-	◌ུ◌	-	-
བྱིན	-	-	བ	◌ྱ◌	◌ི	ན	-
ཡ	-	-	ཡ	-	-	-	-
ཙམ	-	-	ཙ	-	-	ས	-
ཕུར	-	ར	ཕ	◌ུ◌	-	-	-
མཆོ	མ	-	ཆ	-	◌ོ◌	-	-
ཉི	-	-	ཉ	-	◌ི	-	-
ནང	-	-	ན	-	-	ང	-
དུ	-	-	ད	-	◌ུ◌	-	-
ག	-	-	ག	-	-	-	-
ལ	-	-	ལ	-	-	-	-
བྱིན	-	-	བ	◌ྱ◌	◌ི	ན	-
གསུངས	ག	-	ས	-	◌ུ◌	ང	ས
བརྒྱད	བ	ས	ཀ	◌ྱ◌	◌ི	ད	-
བརྒྱད	བ	ར	ག	◌ྱ◌	◌ུ◌	ད	-
བསྒམ	བ	ས	ག	◌ྱ◌	◌ུ◌	མ	-
བསྒབ	བ	ས	ག	◌ྱ◌	◌ུ◌	བ	-
བསྒབས	བ	ས	ག	◌ྱ◌	◌ུ◌	བ	ས
བསྒོག	བ	ས	ག	◌ྱ◌	◌ི	ག	ས
བསྒོས	བ	ས	ག	◌ྱ◌	◌ི	ས	ས
བསྒྲུབ	བ	ར	ག	◌ྱ◌	◌ུ◌	ས	ས

Figure 12: Recognization Result

Note: Pre.=prefix letter; Sup.=superscript letter; Ro.=root letter; Sub=subscript letter; Vow.=vowel; Suf.=suffix letter; FS.=farther suffix letter.

E Appendix E: Rule-based Mappping Algorithm

As shown in Figure 13, the rule-based Tibetan syllable recognition and spelling check method involves preprocessing input text, verifying recognition, performing dictionary-based spelling checks, and outputting results with corrections. The rule-based system checks for legal combinations of prefix, root, vowel, and suffix based on handcrafted linguistic rules and positional constraints. It outputs a binary classification indicating whether a syllable is structurally valid or not.

Note: Pre.=prefix letter; Sup.=superscript letter; Ro.=root letter; Sub=subscript letter; Vow.=vowel; Suf.=suffix letter; FS.=farther suffix letter.

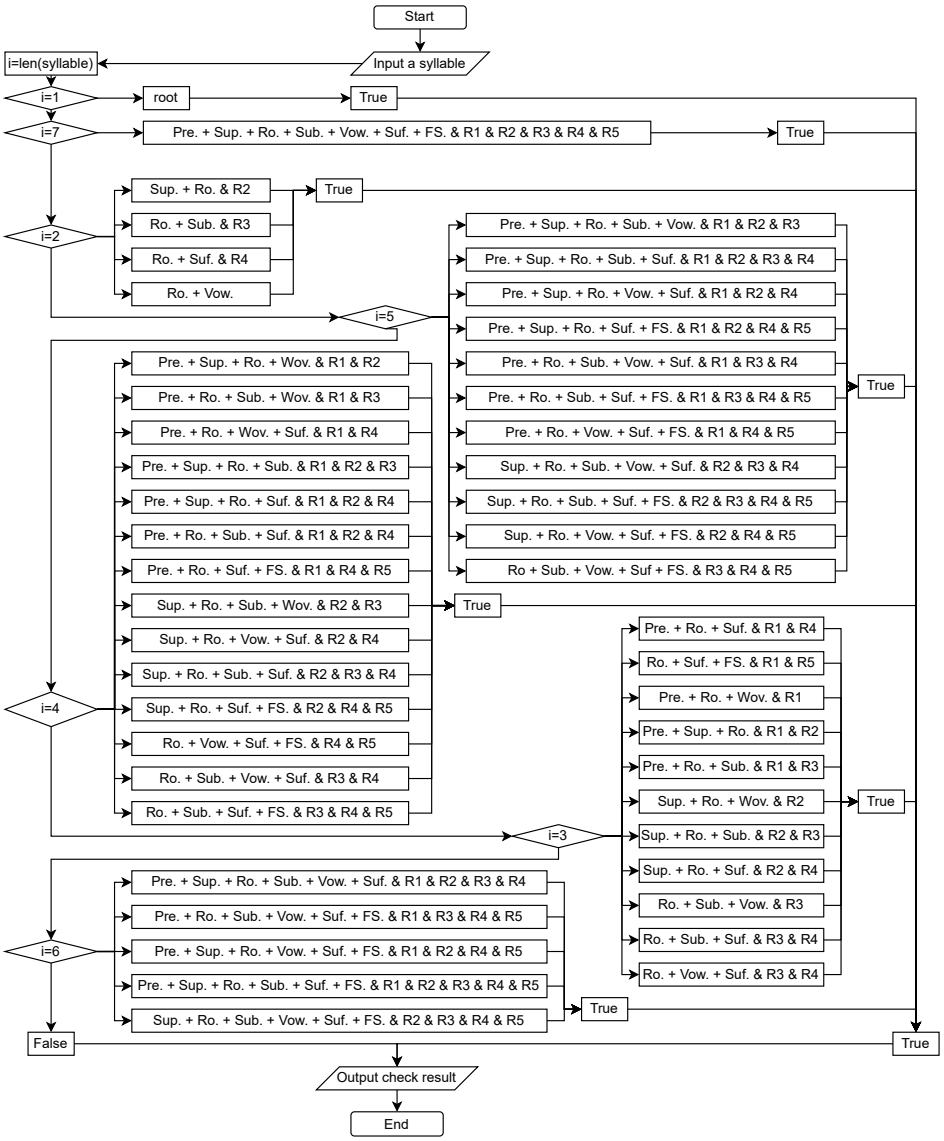


Figure 13: Flow chart of memristive computing-based Tibetan syllable recognization and spelling check method.