Training-Free ANN-to-SNN Conversion for High-Performance Spiking Transformer

Jingya Wang*, Xin Deng*, Wenjie Wei, Dehao Zhang, Shuai Wang, Qian Sun, Jieyuan Zhang, Hanwen Liu, Ning Xie, Malu Zhang†

***Abstract*—Leveraging the event-driven paradigm, Spiking Neural Networks (SNNs) offer a promising approach for constructing energy-efficient Transformer architectures. Compared to directly trained Spiking Transformers, ANN-to-SNN conversion methods bypass the high training costs. However, existing methods still suffer from notable limitations, failing to effectively handle nonlinear operations in Transformer architectures and requiring additional fine-tuning processes for pre-trained ANNs. To address these issues, we propose a high-performance and training-free ANN-to-SNN conversion framework tailored for Transformer architectures. Specifically, we introduce a Multi-basis Exponential Decay (MBE) neuron, which employs an exponential decay strategy and multi-basis encoding method to efficiently approximate various nonlinear operations. It removes the requirement for weight modifications in pre-trained ANNs. Extensive experiments across diverse tasks (CV, NLU, NLG) and mainstream Transformer architectures (ViT, RoBERTa, GPT-2) demonstrate that our method achieves near-lossless conversion accuracy with significantly lower latency. This provides a promising pathway for the efficient and scalable deployment of Spiking Transformers in real-world applications.**

***Index Terms*—Spiking Neural Networks, ANN-to-SNN Conversion, Spiking Transformer, Few Spikes Neuron.**

I. Introduction

SPIKING Neural Networks (SNNs) have garnered widespread attention due to their sparse spike-driven computing paradigm [1], [2]. Unlike traditional Artificial Neural Networks (ANNs) transmit information via continuous-valued activations, SNNs employ sparse binary spikes as information carriers, thereby offering superior energy efficiency for resource-limited devices [3], [4]. Recently, the SNNs community has incorporated high-performance Transformer [5] architectures into SNNs to leverage both the powerful representation capabilities of Transformer and the inherent energy efficiency of the SNNs [6]–[8]. These methods provide significant performance improvements and broaden the application scenarios of SNNs.

To obtain Transformer-based SNNs, two approaches exist: direct training (DT) [9], [10] and ANN-to-SNN (A2S) conversion [11], [12]. DT employs surrogate gradients [13] and backpropagation through time (BPTT), but exhibits inaccurate gradient approximation and $\mathcal{T}\times$ training overhead. In contrast, A2S converts pre-trained ANNs to SNNs, maintaining inference efficiency while avoiding training overhead. A2S methods [14], [15] achieve lossless conversion in convolutional structures by establishing an equivalence between RELU activations and firing rates [16], [17]. However, Transformer conversion remains challenging due to complex nonlinear operations.

Nonlinear operations in Transformer architectures can be divided into two categories: single-variable operations (e.g., GELU, Tanh) and multi-variable operations (e.g., variable-variable floating-point (FP) multiplications, LayerNorm). These operations lead to notable performance degradation [17]. SpikeZIP-TF [18] replaces single-variable functions and applies fine-tuning to adjust pre-trained ANN weights, though at the cost of additional training overhead. STA [19] approximates all nonlinear functions using group operators, but this approach incurs increased inference latency. Therefore, it is necessary to design a high-performance conversion that balances training cost and inference efficiency.

Inspired by the temporal coding mechanisms observed in biological neurons, Few-spikes (FS) neurons [20] are proposed. Unlike rate-coded approaches, it leverages spike timing and patterns to represent neuronal activation states. This characteristic enables near-lossless fitting for single-variable operations. However, vanilla FS neurons suffer from significant performance degradation in large-scale deep networks. Moreover, their design based on single-input fitting limits their ability to handle the multi-variable operations in Transformer architectures.

In this paper, we propose a training-free A2S conversion framework tailored for Transformer architectures, enabling near-lossless conversion without modifying the weights of pre-trained ANNs. We begin by theoretically and empirically analyzing the key challenges associated with FS neuron-based conversion, particularly in approximating nonlinear operations. To overcome these challenges, we introduce a novel Multi-Basis Exponential Decay (MBE) neuron, which effectively approximates various nonlinear operations. Built upon MBE neurons, our method achieves both low inference latency and near-lossless Transformer conversion. The main contributions of this paper are as follows:

1) We systematically analyze the incompatibility between FS neurons and Transformer architectures, focusing on excessive dependence on initialization (EDI) and global suboptimality (GSO) problems, which hinder convergence and degrade conversion performance.
2) We introduce a MBE neuron with exponential decay

*These authors contributed equally to this work.

†Corresponding author.

All authors are with the Department of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu 610054, China.

Corresponding author: Malu Zhang (email: maluzhang@uestc.edu.cn).

strategy and multi-basis encoding method. The decay strategy enables multi-resolution representations, while multi-basis encoding enhances the near-lossless approximation of diverse nonlinear operations.

3) We propose an A2S framework based on MBE neurons that efficiently approximates various nonlinear operations in Transformer architectures, including variable-variable FP multiplications, GELU, Softmax, and LayerNorm. It achieves near-lossless conversion with fast inference and require no training on source ANNs.

4) Extensive experiments on diverse tasks (CV, NLU, NLG) and architectures (ViT, RoBERTa, GPT-2) demonstrate that our method achieves near-lossless conversion with reduced latency, yielding competitive results among existing Transformer-based A2S methods.

II. Related work

Existing conversion paradigms can be categorized into two types based on whether extra training on the source ANNs is required: training-dependent and training-free. ***Training-dependent conversion*** typically requires additional ANNs training based on theoretical ANN-SNN equivalence. Clip-Floor-Shift activations [16] replace ReLU to better match spiking neuron behavior. A two-phase training [21] first optimizes ANN weights, then adjusts neuron thresholds, while activation range constraints and membrane potential initialization [22] help further reduce conversion error. [23] introduce a two-stage approach to handle quantization, pruning, and residual errors. [24] propose a unified framework treating spike-rate mapping as a differentiable problem. For Transformer architectures, a QANN is integrated with spatiotemporal properties of spiking neurons for lossless conversion [18]. While effective, these methods introduce computational overhead through additional source ANNs training.

Training-free conversion directly transforms pre-trained ANNs via structure and parameter reuse without fine-tuning. Early works align SNN thresholds with ANN activations via threshold balancing [11], [12], but longer timesteps are required. Signed spiking neurons [25] support dynamic vision data. In [19], training-free Transformer conversion is achieved using universal group operators and spatial rectification self-attention. SpikedAttention [26] enables spike-driven Transformer A2S via trace-based matrix multiplication and winner-take-all spike shifting, yet retains non-spiking LayerNorm. ECMT [27] reduces latency in Transformer SNNs, yet still relies on floating-point operations. While retraining is avoided, these methods often require long timesteps or retain non-spiking components, limiting SNNs' energy efficiency.

III. Preliminary & Problem Analysis

A. Preliminary

Inspired by authentic electrophysiological characteristics of human brain neurons, FS neuron is proposed to address the inherent trade-off between spike count and accuracy in A2S [20]. By introducing learnable parameters including neuron threshold, membrane potential reset value, and spike intensity, FS neuron approximates ANN activation functions with very few spikes. Mathematically, the membrane potential of FS neuron can be computed as:

$$u[t+1] = u[t] - r[t] \cdot s[t], \tag{1}$$

where $t \in [0, T-1]$ denotes timesteps, $u[t]$ represents the membrane potential, $s[t] \in \{0, 1\}$ is the binary spike, and $r[t]$ is the learned reset value. The membrane potential has no decay, and its initial value is typically set to the input from the previous layer, i.e., $u[0] = x$. When the membrane potential exceeds a learned threshold, a spike is generated, which is described as follows:

$$s[t] = \mathcal{H}\left(\left(x - \sum_{t'=0}^{t-1} r[t'] \cdot s[t']\right) - V_{th}[t]\right), \tag{2}$$

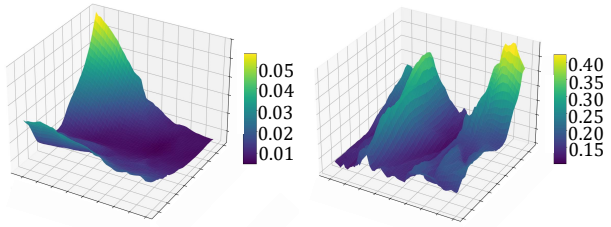
where $\mathcal{H}(\cdot)$ is the Heaviside step function and $V_{th}[t]$ is the learned firing threshold. After each spike emission, the reset mechanism is invoked to update the membrane potential, as described in Eq.(1). Noteworthy, the spike $s[t]$ emitted by FS neuron at time t is amplified by learned spike intensity $d[t]$. After T timesteps, the neuron integrates the weighted spike train and forwards it to the next layer:

$$\hat{f}(x) = \sum_{t=0}^{T-1} d[t]s[t] \approx f(x), \tag{3}$$

where $\hat{f}(x)$ is the approximation value produced by FS neuron and $f(x)$ is the output of activation function in ANNs.

B. Problem Analysis

We conduct a systematic analysis of FS neurons to understand their limitations in Transformer-based A2S tasks.



(a) Binary initialization (b) Random initialization

Fig. 1. Loss landscape of FS neurons in approximating the ReLU activation function. The 3D landscape of $\mathcal{L}_{Binary}$ and $\mathcal{L}_{Random}$ from two different initialization.

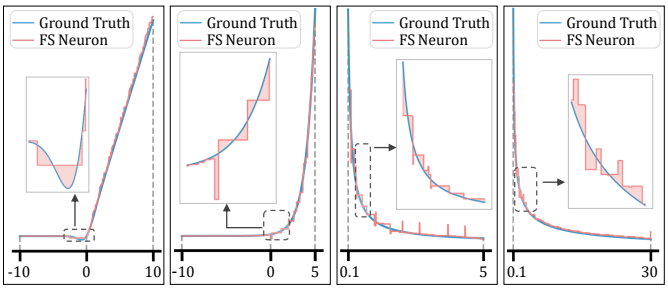
Approximation Error Analysis To qualitatively analyze how the FS neuron design impacts performance, following [19], we consider errors from three sources: insufficient sampling, limited parameterization, and spiking quantization. This yields the following error bound:

Theorem 1 (FS Neuron Error Bound)**.** *Let $f : [a, b] \to \mathbb{R}$ be a target activation function, and let $\hat{f}_T^{(M)}(x)$ denote the output of an FS neuron with T timesteps trained on M samples. Then the total approximation error satisfies:*

$$\varepsilon \leq \mathcal{O}\Bigg(\underbrace{\sqrt{\frac{T \log T \log M}{M}}}_{\textit{Empirical Gap}} + \underbrace{\frac{\mathcal{L}_f |y|_{\max}}{T}}_{\textit{Parametric Gap}} + \underbrace{\frac{\|d\|_1}{T}}_{\textit{Quantization Gap}}\Bigg),$$

where $\varepsilon = \mathbb{E}[|f(x) - \hat{f}_T^{(M)}(x)|]$, $\mathcal{L}_f$ *is the Lipschitz constant of* f, $|y|_{\max}$ *the maximum value of* $f(x)$, *and* $\|d\|_1$ *is the* L_1*-norm of spike intensities. Proof in Appendix A.*

1) Excessive Dependence on Initialization: From Theorem 1, the quantization gap $\frac{\|d\|_1}{T}$ depends on spike intensities $d[t]$, which are determined through optimization from initialization. Poor initialization may produce suboptimal $d[t]$ that increase $\|d\|_1$, thereby enlarging the approximation error. We conduct experiments to validate this: random initialization yields $\text{MSE}_{\text{random}} = 1.5 \times 10^{-3}$, while binary initialization ($V_{\text{th}}[t] = r[t] = d[t] = 2^{T-t}$) achieves $\text{MSE}_{\text{binary}} = 9.4 \times 10^{-5}$, representing a 93.29% performance degradation under random initialization. As shown in Fig. 1, binary initialization results in a smoother and more stable optimization landscape, while random initialization leads to multiple local minima. This further corroborates the sensitivity of FS neurons to initialization quality.



(a) GELU (b) e^x (c) $1/x$ (d) $1/\sqrt{x}$

Fig. 2. Critical interval approximation failure of nonlinear operations in Transformer using FS neurons.

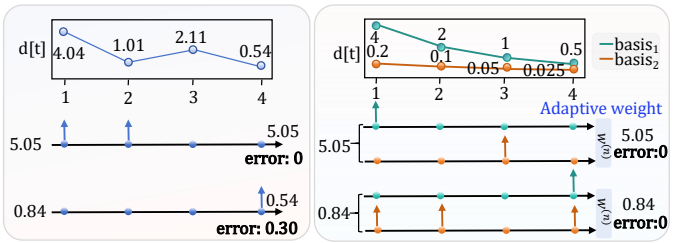
2) Global Suboptimality: GSO stems from the parametric gap term $\frac{\mathcal{L}_f |y|_{\max}}{T}$ in Theorem 1, revealing FS neurons' limitations when approximating functions with non-uniform complexity. Transformer nonlinearity inputs concentrate near zero (Statistical analysis in Appendix C), where activation functions such as GELU and SiLU have high curvature and large local Lipschitz constants $\mathcal{L}_f^{\text{local}} \gg \mathcal{L}_f$. This creates an amplified local gap $\varepsilon_{\text{param}}^{\text{local}} = \mathcal{O}(\mathcal{L}_f^{\text{local}} |y|_{\max}^{local} / T)$, which dominates the approximation error and highlights the inadequacy of uniform time allocation. Experimental results support this analysis: as shown in Fig. 2, although the overall approximation error is small, the fitting degrades significantly in the near-zero region where outputs vary most rapidly. This region is crucial for Transformer performance, and the poor local approximation dominates overall behavior (FS-based Transformer conversion results in Appendix D).

IV. Method

In this section, we first propose the MBE neuron, which employs exponential decay parameter update strategy and multi-basis encoding method to map activation values at multiple resolutions. Based on the MBE neuron, we further design an A2S conversion framework to overcome key challenges for Transformer architectures, achieving near-lossless conversion that is training-free and low-latency.

A. Multi-Basis Exponential Decay Neuron

To address the limited expressiveness of FS neurons, we propose a novel MBE neuron with an exponential decay parameter update strategy and multi-basis encoding scheme. Unlike FS neurons requiring learning multiple parameters $(d[t], r[t], V_{\text{th}}[t])$ per timestep, MBE neurons only learn decay rates and discrete timesteps, reducing parameter overhead and gradient instability (Details in Appendix E).



(a) FS neuron (b) MBE neuron with 2 basis

Fig. 3. FS neuron and MBE neuron encoding methods.

The exponential decay mechanism enables progressive refinement from coarse to fine resolution, allowing adaptive approximation of functions with varying granularity and rapid focus on limited value ranges over shorter timesteps, as illustrated in Fig. 3. The parameter update is defined as:

$$Para(\tau_n, t) = \alpha \cdot exp\left(-\frac{t\Delta t}{\tau_n}\right), \tag{4}$$

where $Para(\tau_n, t)$ represents the parameter value at time t, α is a hyperparameter typically set to the target function's maximum value, Δt is the discrete timestep, and $\tau_n \in \{\tau_{d_n}, \tau_{r_n}, \tau_{V_{th_n}}\}$ are the corresponding decay rates.

To further enhance the representational capacity for approximating diverse nonlinear functions, we introduce multi-basis encoding as the second core feature. As shown in Fig.4, each MBE neuron comprises n basis components contributing distinct functional components to the overall response. The dynamics of the n-th basis are defined as:

$$u_n[t+1] = u_n[t] - s_n[t] \cdot r_n[t], \tag{5}$$

$$s_n[t] = \mathcal{H}\big(u_n[t] - V_{thn}[t]\big), \tag{6}$$

$$o_n[t+1] = o_n[t] + s_n[t] \cdot d_n[t], \tag{7}$$

where $u_n[t]$ is the membrane potential of basis n at time t, $o_n[t]$ is the accumulated output, $d_n[t]$ denotes the spike intensity, $r_n[t]$ is the reset value, $V_{thn}[t]$ is the firing threshold, and $\mathcal{H}(\cdot)$ is the Heaviside step function. When $u_n[t] \geq V_{thn}[t]$, the neuron fires and emits a spike with intensity $d_n[t]$ while reducing the membrane potential by $r_n[t]$.

Finally, all basis outputs are weighted by a w to form $\hat{f}(x)$, passed to the next layer to approximate $f(x)$:

$$\hat{f}(x) = \sum_{n=1}^{N} w^{(n)} \cdot \mathrm{o}^{(n)}(T) \approx f(x). \tag{8}$$

Approximation Error Analysis Following the same analytical framework in Theorem 1, we derive error bounds for MBE neurons, accounting for the representational capacity:

Theorem 2 (MBE Neuron Error Bounds)**.** *For a target activation function* $f : [a, b] \to \mathbb{R}$ *and MBE neuron output* $\hat{f}_{N,T}^{(M)}(x)$

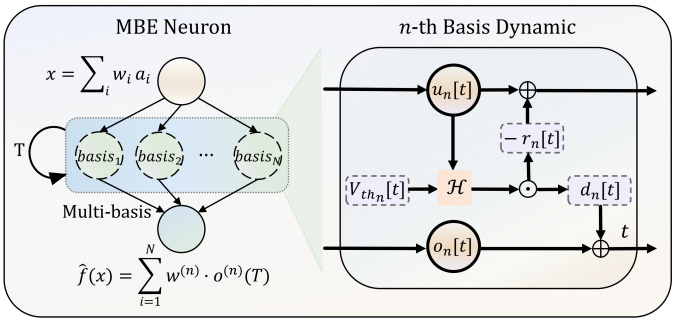


Fig. 4. Multi-basis exponential decay neuron.

with N basis components, T timesteps trained on M samples, the total approximation error satisfies:

$$\varepsilon^* \leq O\left(\underbrace{\sqrt{\frac{N\log(N)\log M}{M}}}_{\textit{Empirical Gap}} + \underbrace{\frac{\mathcal{L}_f|y|_{max}}{NT}}_{\textit{Parametric Gap}} + \underbrace{\frac{\|w\|_1|\alpha|\tau_{max}}{T\Delta t}}_{\textit{Quantization Gap}}\right),$$

where $\varepsilon^* = \mathbb{E}[|f(x) - \hat{f}^{(M)}_{N,T}(x)|]$, $\mathcal{L}_f$ is the Lipschitz constant of f, $|y|_{max}$ is the maximum absolute value of the target function, $\|w\|_1$ is the L_1 norm of basis weights, $|\alpha|$ is the positive scaling parameter, $\tau_{max} = \max_n\{\tau_{d_n}, \tau_{r_n}, \tau_{V_{th_n}}\}$ is the maximum time constant, and Δt is the discrete timestep. The proof is provided in Appendix B.

The theoretical analysis shows MBE advantages and guides our conversion framework through three key insights:

Basis component design: The Parametric Gap $\mathcal{O}(1/NT)$ outperforms FS neurons' $\mathcal{O}(\mathcal{L}_f|y|_{\max}/T)$ by enabling multi-basis encoding to adaptively focus on high-curvature regions, solving the GSO problem without extra timesteps.

Initialization stability: The Quantization Gap $\frac{\|w\|_1|\alpha|\tau_{max}}{T\Delta t}$ mitigates the EDI problem through controlled scaling via $|\alpha|$ and time constant effects via $\tau_{max}/\Delta t$, enabling initialization-stability parameter updates.

Hyperparameter determination: The $1/NT$ scaling reveals a principled trade-off: increasing basis components N can effectively compensate for limited timesteps T, thereby achieving a balance between accuracy and low latency.

B. Conversion Framework

Based on MBE neurons, we propose a framework for Transformer-to-SNN conversion. As shown in Fig. 5, by decomposing non-spiking components (nonlinear activations, FP multiplication, Softmax, LayerNorm) into basic functions and designing corresponding MBE neurons for equivalent approximation, we enable spiking-based representation.

Approximation of Nonlinear Activation Functions

To enable accurate and efficient spike-based approximation of nonlinear activation functions such as GELU and Tanh in Transformers, we employ MBE neurons with $N = 4$ basis components (in Eq.(8)). Since LayerNorm compresses activations into a narrow range, we constrain the GELU input domain to $(-120, 10)$ to enhance both approximation accuracy and training stability. Within this interval, we uniformly sample $M = 10{,}000$ points from the target function $f(x)$ and compute the spike-based output $\hat{f}(x)$ over T timesteps.

Approximation of Floating-Point Multiplication

The FP multiplication operands x_1 and x_2 are transformed through approximate identity mapping by the MBE neuron MBE_{Id}, yielding the spike-train representations:

$$x \approx MBE_{Id}(x) = \sum_{t=0}^{T-1} d[t]s[t], \quad x \in \{x_1, x_2\}. \tag{9}$$

Then, the multiplication of x_1 and x_2 is expressed as:

$$x_1 \cdot x_2 \approx \sum_{i=0}^{T-1}\sum_{j=0}^{T-1} d[i]d'[j] \cdot s[i]s'[j]. \tag{10}$$

We represent the spike sequences of two MBE neurons as vectors $\mathbf{s} = \{s[t]\}_{t=0}^{T-1}$ and $\mathbf{s}' = \{s'[t]\}_{t=0}^{T-1}$ respectively, and denote their corresponding intensity sequences as vectors $\mathbf{d} = \{d[t]\}_{t=0}^{T-1}$ and $\mathbf{d}' = \{d'[t]\}_{t=0}^{T-1}$. Based on these temporal vectors, we construct the intensity matrix $\mathbf{D} \in \mathbb{R}^{T\times T}$ and the spike matrix $\mathbf{S} \in \{0,1\}^{T\times T}$ as follows:

$$\mathbf{D} = \mathbf{d}^\top\mathbf{d}', \qquad \mathbf{S} = \mathbf{s}^\top\mathbf{s}'. \tag{11}$$

To achieve multiplication in the form of spikes, we employ the Hadamard product followed by summation. Consequently, Eq.(10) can be expressed as:

$$x_1 \cdot x_2 \approx \sum_{i=0}^{T-1}\sum_{j=0}^{T-1} \mathbf{D}_{ij} \cdot \mathbf{S}_{ij} = \sum_{i=0}^{T-1}\sum_{j=0}^{T-1} (\mathbf{D} \odot \mathbf{S})_{ij}, \tag{12}$$

where $\mathbf{D}_{ij} = d[i]d'[j]$ and $\mathbf{S}_{ij} = s[i]s'[j]$. The intensity matrix $\mathbf{D}$ can be precomputed and shared across the entire network. With the binary matrix $\mathbf{S}$, the entire FP multiplication approximation process maintains its spike-driven characteristics (Details in Appendix. F.1). By replacing FP multiplications in self-attention with our spike-driven operations, we enable fully spike-based attention matrix computation.

Approximation of Softmax

The $\text{Softmax}(x_i) = \frac{e^{x_i}}{\sum_j e^{x_j}}$, where i ,j denote the indices of elements in the sequence. Since softmax depends on all inputs, it cannot be directly implemented by a single MBE neuron. As shown in Fig. 5(b), we decompose the softmax computation into three components: exponential (e^x), reciprocal ($1/x$), and FP multiplication, each approximated by MBE neurons.

For the term e^{x_i}, we apply the change-of-base formula to decompose it into integer and decimal components as:

$$e^{x_i} = 2^{x_i\cdot\log_2 e} = 2^{\lfloor x_i\cdot\log_2 e\rfloor} \cdot 2^{x_i - \lfloor x_i\cdot\log_2 e\rfloor}, \tag{13}$$

where $\lfloor\cdot\rfloor$ denotes the floor operation. The decimal component $2^{x_i-\lfloor x_i\cdot\log_2 e\rfloor}$ is approximated by MBE neurons, and the integer exponent $2^{\lfloor x_i\cdot\log_2 e\rfloor}$ is achieved by hardware-efficient addition [28]. For the reciprocal term $1/\sum_j e^{x_j}$, we follow the IEEE 754 standard [29] to extract the exponent E and mantissa M through the corresponding bits of $\sum_j e^{x_j}$ (i.e., $\sum_j e^{x_j} = M \cdot 2^E$). The reciprocal $1/M$ is directly approximated using an MBE neuron, and the final result is obtained as $1/\sum_j e^{x_j} = 2^{-E}/M$.

Combining the approximations of e^{x_i} and $1/\sum_j e^{x_j}$, together with our FP multiplication approximation scheme, we

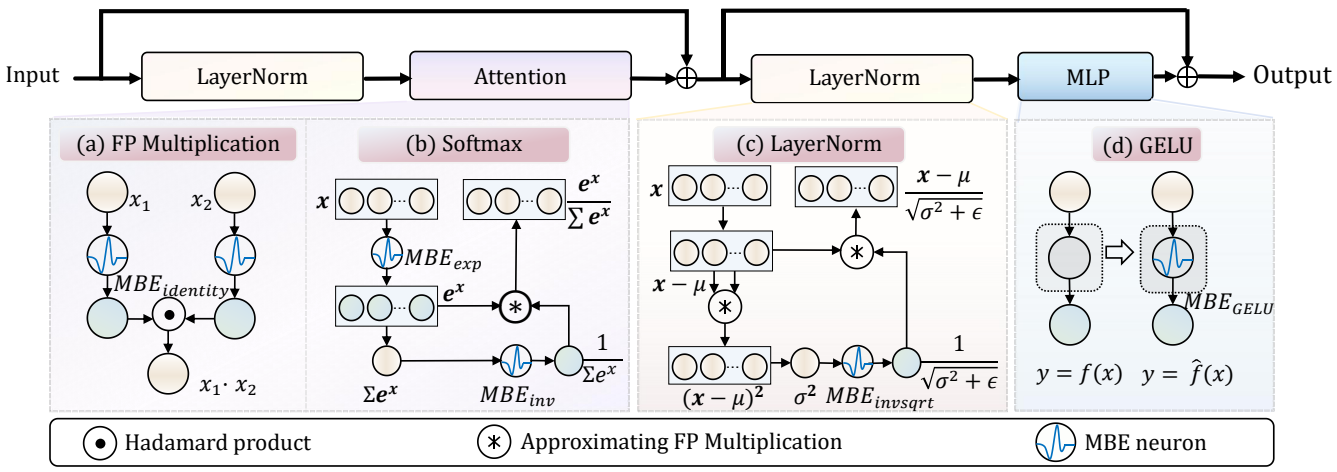


Fig. 5. Overview of our framework, which depicts the approximations of FP multiplication, Softmax, LayerNorm, and GELU.

compute the final softmax output in a spike-based manner (Detailed Softmax algorithm table in Appendix F.2).

Approximation of LayerNorm

The $\text{LN}(x_i) = \gamma \cdot \frac{x_i - \mu}{\sqrt{\sum(x_i-\mu)^2/n+\epsilon}} + \beta$, where μ is the input mean, n is the number of elements, ϵ is a stability constant, and γ, β are learnable parameters. As shown in Fig. 5(c), we also decompose LayerNorm into three spike-unfriendly operations: squaring (x^2), reciprocal square root ($1/\sqrt{x}$), and FP multiplication, each approximated by MBE neurons.

For the squared term $\sum(x_i - \mu)^2$, we adopt the proposed FP multiplication approximation with the intensity matrix $\mathbf{D}$ pre-scaled by $1/n$ to directly yield $(x_i - \mu)^2/n$.

For the inverse square root $1/\sqrt{\sum(x_i - \mu)^2/n + \epsilon}$, we decompose the variance term into exponent E and mantissa M. We adjust E and M based on E's parity to ensure $-E/2$ is an integer. The term $1/\sqrt{M}$ is approximated directly using MBE neuron, and the inverse square root is obtained by multiplying with $2^{-E/2}$, similar to the Softmax approximation. Finally, we multiply $(x_i - \mu)$ with the approximated $1/\sqrt{\sum(x_i - \mu)^2/n + \epsilon}$ using our FP multiplication approximation to obtain the LayerNorm output.

Based on the aforementioned conversion of all non-spike-friendly components into equivalent spiking forms, we replace the corresponding modules to construct the Transformer-based SNN without modifying the original network parameters. Detailed pseudocode is in Appendix F.3.

V. Experiments

A. Experimental Setup

We conduct experiments on a GPU (RTX 4090) environment using the PyTorch framework, evaluating both pre-trained Vision Transformer (ViT) models, including ViT-Base-Patch16 (ViT-B/16) [40] and ViT-Medium-Patch16-Reg4-Gap-256 (ViT-M/16) [41], as well as CNN models, including VGG16 [42] and ResNet34 [43] on the ImageNet dataset [44]. To validate the generalizability of our method across different language domains and tasks, we adapt RoBERTa [45] for natural language understanding (NLU) tasks and employ GPT-2 [46] for natural language generation (NLG) task evaluation (Experimental and implementation details in Appendix G.1).

B. Comparative Study

a) ***Comparison on CV****:* We evaluate the performance of our conversion framework by applying it to convert both ViT and CNN architectures. As shown in Tab. I, ViT-B/16 and ViT-M/16 achieve conversion losses of 0.44% and 0.64% respectively—significantly lower than most existing methods. The framework attains 85.31% accuracy for ViT and 75.57% for CNN, outperforming other SNNs. While ECMT achieves competitive accuracy at shorter timesteps, it retains FP multiplication in its expectation compensation module. In contrast, our A2S framework maintains crucial spike-based property, thereby avoiding the high energy consumption typically associated FP multiplication during inference. Furthermore, the training-free nature of our approach enables conversion with minimal computational overhead, eliminating the need for extensive retraining for source ANNs. Our method successfully accomplishes A2S conversion without requiring additional training of the original network, preserves the essential spiking properties, delivers superior performance at short timesteps, and achieves optimal results across all evaluation dimensions.

b) ***Comparison on NLU****:* To evaluate our method on NLU tasks, we conduct experiments on four benchmark datasets including SST-2, SST-5, MR, and Subj, following standard experimental settings from studies [18], [47]. As shown in Table II, our method achieves competitive performance across all datasets, outperforming existing methods on both RoBERTa-Base (125M) and RoBERTa-Large (355M). On SST-2, it attains 95.98% accuracy with $T = 16$, representing only 0.24% degradation from the original ANN (96.22%), while outperforming SpikeZIP-TF [18] by 2.19% and reducing timesteps by 87.5%. Similarly, on MR with RoBERTa-Base, our method achieves 89.00% accuracy at $T = 16$, exceeding SpikeZIP-TF by 2.87% with 75% fewer timesteps.

c) ***Comparison on CV****:* We evaluate the performance of our Spike-F^3 framework by applying it to convert both ViT-B/16 and ViT-M/16 architectures. The experiment results

TABLE I
IMAGENET PERFORMANCE COMPARISON.

Category	Method	Arch.	Spike-Driven	Training-Free	Param[M]	Timesteps	ANN	Acc.[%](Δ)
DT	DSR [30]	ResNet-18	✔	-	12	50	-	67.74
	TET [31]	SEW-ResNet-34	✘	-	22	4	-	68.00
	AT-SNN [32]	ResNet-104	✘	-	45	4	-	77.08
	Spikingformer [33]	-4-384-400E	✔	-	66	4	-	75.85
	SDTv1 [34]	-8-768	✔	-	66	4	-	77.07
	Spikeformer [35]	-7L/3×2×4	✘	-	38	4	-	78.31
	QKFormer [36]	HST-10-768	✔	-	65	4	-	84.22
	SDTv3 [8]	E-SpikeFormer	✔	-	173	8	-	85.10
A2S	QCFS [16]	VGG-16	✔	✘	138	64	74.92	72.85(-2.07)
	QFFS [37]	VGG-16	✔	✘	138	8	73.08	73.10(+0.02)
	SNM [25]	ResNet-18	✔	✔	12	64	73.18	71.50(-1.68)
	QCFS [16]	ResNet-34	✔	✘	22	64	74.23	72.35(-1.88)
	AdaFire [38]	ResNet-34	✔	✔	22	8	75.66	72.96(-2.70)
	ECL [39]	ResNet-34	✔	✘	22	16	74.36	72.37(-1.99)
	Ours	VGG16	✔	✔	138	10	73.37	72.61 (-0.76)
		ResNet-34	✔	✔	22	10	76.31	**75.57** (-0.74)
	ECMT [27]	ViT-L/16	✘	✔	307	12	84.88	84.71(-0.17)
	STA [19]	ViT-B/32	✘	✔	86	256	83.60	82.79(-0.81)
	SpikeZIP-TF [18]	SViT-L-32Level	✔	✘	304	64	85.41	83.82(-1.59)
	SpikedAttention [26]	Swin-T(BN)	✘	✔	28	48	79.30	77.20(-2.10)
	Ours	ViT-B/16	✔	✔	86	16	83.44	83.00(-0.44)
		ViT-M/16	✔	✔	64	16	85.95	**85.31**(-0.64)

demonstrate that our Spike-F^3 method achieves SOTA performance across multiple metrics. As illustrated in Tab. I, the conversion loss for the ViT-B/16 and ViT-M/16 models is measured at 0. 44% and 0. 64%, respectively, significantly lower than most existing methods, while achieving high accuracy 85.31% at T=16. Although ECMT [27] achieves good accuracy at shorter timesteps, it keeps FP multiplication in the expectation compensation module for matrix product. In contrast, converted SNNs based on Spike-F^3 maintain crucial spike-based property, thereby avoiding the high energy consumption typically associated FP multiplication during inference. Furthermore, the training-free nature of our approach enables conversion with minimal computational overhead, eliminating the need extensive retraining for source ANNs. Our method successfully accomplishes nearly lossless Transformer A2S conversion while avoiding retraining of the original ANN. It effectively preserves essential spiking neural dynamics, demonstrates superior inference accuracy under low timesteps conditions, and ultimately achieves SOTA performance across all comprehensive evaluation metrics.

TABLE II
NLU PERFORMANCE COMPARISON

Category	Model	Param	SST-2	SST-5	MR	Subj	T
ANN	Roberta [45]	125	94.49	55.46	89.39	96.45	1
		355	96.22	59.37	91.36	97.50	1
DT	Spikeformer [35]	110	81.55	42.02	79.38	91.80	4
	SpikeBERT [48]	109	85.39	46.11	80.69	93.00	4
	SpikeGPT [47]	45	80.39	37.69	69.23	88.45	50
A2S	SpikeZIP-TF [18]	125	92.81	52.71	86.13	95.55	64
		355	93.79	56.51	89.28	96.70	128
	Ours	125	93.46	55.11	89.00	96.30	16
		355	**95.98**	**58.31**	**90.96**	**97.45**	16

TABLE III
NLG PERFORMANCE COMPARISON. BOTH WIKI-2 AND WIKI-103 USE PERPLEXITY. ↓ INDICATES LOWER IS BETTER.

Category	Model	Param [M]	Wiki-2 ↓	Wiki-103 ↓	T
ANN	GPT-2 [46]	346	22.34	22.65	1
DT	SpikeGPT [47]	216	18.01	39.75	1024
A2S	Spike-F^3 (Ours)	346	22.69 (+0.35)	23.41 (+0.76)	16

*d) **Comparison on NLU**:* To assess the effectiveness of Spike-F^3 in NLU tasks, we conduct experiments on four benchmark datasets including SST-2, SST-5, MR, and Subj, following standard experimental settings from previous studies [18], [47]. As shown in Tab. II, Spike-F^3 achieves SOTA performance across all evaluated datasets, consistently outperforming existing methods on both RoBERTa-Base (125M) and RoBERTa-Large (355M). On SST-2, it achieves 95.98% accuracy with T=16, resulting in only 0.24% conversion loss compared to the original ANN (96.22%). Compared to SpikeZIP-TF [18], Spike-F^3 improves accuracy by 2.19% and reduces timesteps by 87.5%. In the lightweight setting, Spike-F^3 achieves 89.00% accuracy on MR using RoBERTa-Base with T=16, surpassing SpikeZIP-TF by 2.87% while reducing timesteps by 75%. These results confirm the effectiveness of Spike-F^3 in training-free A2S for NLU tasks.

*e) **Comparison on NLG**:* To demonstrate the applicability of our method to NLG tasks, we evaluate its performance on WikiText-2 and WikiText-103. As shown in Tab. III, our method achieves 22.69 perplexity on WikiText-2 with only T=16 timesteps, incurring merely 0.35% conversion loss compared to GPT-2. More significantly, on WikiText-103, it attains 23.41 perplexity—a substantial 41.1% improvement over directly-trained SpikeGPT (39.75) while using dramati-

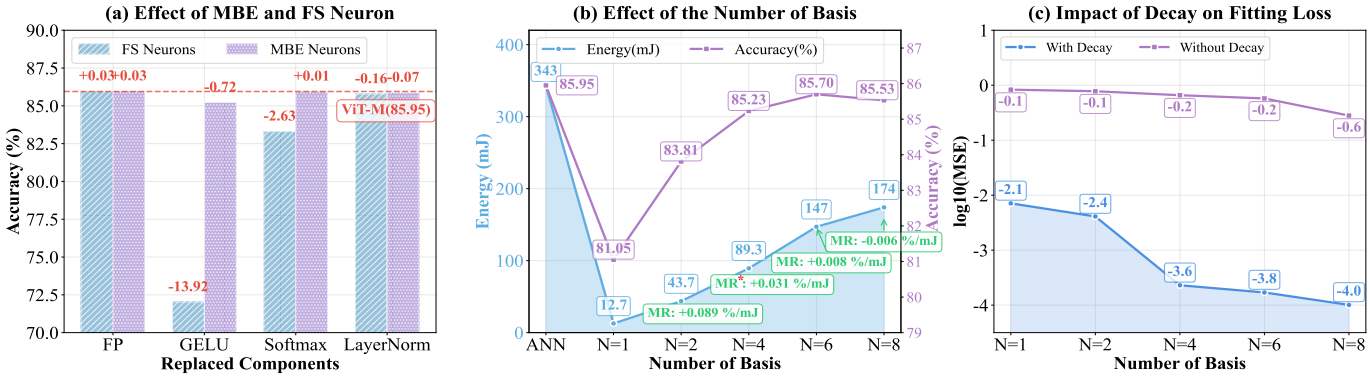


Fig. 6. (a) Effect of MBE and FS neuron on components in Transformer, FP in (a) means floating multiplication. (b) Impact of MBE neurons with varying basis counts on energy consumption and accuracy when replacing GELU. MR represents marginal rate of change between accuracy and energy consumption. (c) Effect of the basis and decay. Loss comparison between models with (blue) and without (red) decay mechanism across basis numbers N=1,2,4,6,8. Y-axis shows log MSE loss (lower is better).

cally fewer timesteps (T=16 vs. T=1024), closely approaching the original GPT-2 performance and demonstrating superior efficiency for long-context generation.

C. Ablation Study

To compare MBE neurons and FS neurons, we evaluate their performance on key Transformer components. For GELU and FP multiplication, direct replacement is employed. As shown in Fig. 6(a), MBE neurons surpass FS neurons by 13.20% in GELU conversion and achieve improvements in Softmax and LayerNorm, demonstrating superior approximation ability. In Fig. 6(b), increasing N leads to more accurate function approximations by MBE neurons. MBE neurons achieve high performance (e.g., less than 1% conversion loss when N $\geq$ 4) while offering significantly higher energy efficiency than ANNs. N=4 achieves the optimal accuracy-MR trade-off, representing the knee point where marginal gains plateau. The synergistic effect between exponential decay and multi-basis encoding is demonstrated in Fig. 6(c). Incorporating decay consistently reduces MSE by 1–2 orders of magnitude across all basis numbers, with significant performance improvements. This reveals that exponential decay creates a multiplicative synergy with multi-basis encoding, enabling exceptional approximation precision unattainable through basis expansion alone.

TABLE IV
PERFORMANCE ACROSS VARIOUS TIMESTEPS AND MODELS. FOR WIKI-103, PERPLEXITY (↓) IS USED; LOWER IS BETTER.

Dataset	Model	ANN	Timestep (T)			
			8	10	12	16
ImageNet	ViT-B/16	83.44	0.12	79.96	82.79	**83.00**
	ViT-M/16	85.95	1.17	21.99	84.43	**85.31**
MR	Roberta-B	89.39	50.17	71.30	88.55	**89.00**
	Roberta-L	91.36	49.66	68.62	90.52	**90.96**
Wiki-103 ↓	GPT-2	22.65	41072	11992	33.56	**23.41**

We evaluate ImageNet, MR, and Wiki-103 to analyze timestep requirements for A2S conversion across vision and language tasks. As shown in Tab. IV, our method achieves about 1% conversion loss for ViT, RoBERTa, and GPT-2 at T=16, with near-optimal performance at T=12, demonstrating efficient progressive encoding. Notably, ViT-M/16 shows a 21.99% accuracy drop at T=10 vs. ViT-B/16, primarily due to its wider identity mapping range [0,62] causing amplified approximation errors under limited timesteps.

VI. Energy Estimation

Unlike ANNs where energy depends on floating-point operations (FLOPs), the energy cost of SNNs is dominated by synaptic operations (SOPs). Following [49], the energy ratio between SNNs and ANNs is:

$$\frac{E_{\text{SNN}}}{E_{\text{ANN}}} = \frac{\text{SOPs} \cdot E_{\text{AC}}}{\text{FLOPs} \cdot E_{\text{MAC}}}, \tag{14}$$

where E_{MAC} = 4.6pJ, E_{AC} = 0.9pJ. For ViT-M/16, we measure the firing rates η of nonlinear operations across all layers. Taking GELU as an example, the ANN implementation requires 70 FLOPs, while the MBE neuron achieves η=38.22% under N=4 and T=16. The synaptic operations achieve energy consumption of 13.7% according to $T * N * \eta * E_{AC}$. Other operations yield greater energy savings owing to lower MBE firing rates. Complete results and detailed firing rates are provided in Appendix G.4.

VII. Conclusion

This paper proposes an efficient Spiking Transformers conversion method that requires no additional training on the source ANNs. We first theoretically and experimentally identify two key challenges in using FS neurons to approximate nonlinear operations: excessive dependence on initialization (EDI) and global suboptimality (GSO) problems. To address these issues, the MBE neuron employs an exponential decay strategy and a multi-basis encoding method to more accurately approximate the nonlinear operations in Transformer architecture. Based on the MBE neuron, a general ANN-to-SNN conversion framework is developed, which supports spiking nonlinear activation functions, spiking FP multiplications, spiking Softmax, and spiking LayerNorm. Extensive experiments on

various models (CNN, ViT, RoBERTa, GPT-2) across tasks (CV, NLU, NLG) confirm that the proposed method achieves near-lossless conversion. Therefore, our method provides a promising approach for the efficient and scalable deployment of Transformer-based SNNs in real-world applications.

REFERENCES

[1] Wolfgang Maass, "Networks of spiking neurons: the third generation of neural network models," *Neural networks*, vol. 10, no. 9, pp. 1659–1671, 1997.

[2] Eugene M Izhikevich, "Simple model of spiking neurons," *IEEE Transactions on neural networks*, vol. 14, no. 6, pp. 1569–1572, 2003.

[3] Kaushik Roy, Akhilesh Jaiswal, and Priyadarshini Panda, "Towards spike-based machine intelligence with neuromorphic computing," *Nature*, vol. 575, no. 7784, pp. 607–617, 2019.

[4] Jilin Zhang, Dexuan Huo, Jian Zhang, Chunqi Qian, Qi Liu, Liyang Pan, Zhihua Wang, Ning Qiao, Kea-Tiong Tang, and Hong Chen, "22.6 anpi: A 28nm 1.5 pj/sop asynchronous spiking neural network processor enabling sub-o. 1 μj/sample on-chip learning for edge-ai applications," in *2023 IEEE International Solid-State Circuits Conference (ISSCC)*. IEEE, 2023, pp. 21–23.

[5] 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*, vol. 30, 2017.

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

[7] Man Yao, Jiakui Hu, Tianxiang Hu, Yifan Xu, Zhaokun Zhou, Yonghong Tian, Bo Xu, and Guoqi Li, "Spike-driven transformer v2: Meta spiking neural network architecture inspiring the design of next-generation neuromorphic chips," *arXiv preprint arXiv:2404.03663*, 2024.

[8] Man Yao, Xuerui Qiu, Tianxiang Hu, Jiakui Hu, Yuhong Chou, Keyu Tian, Jianxing Liao, Luziwei Leng, Bo Xu, and Guoqi Li, "Scaling spike-driven transformer with efficient spike firing approximation training," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2025.

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

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

[11] Peter U Diehl, Daniel Neil, Jonathan Binas, Matthew Cook, Shih-Chii Liu, and Michael Pfeiffer, "Fast-classifying, high-accuracy spiking deep networks through weight and threshold balancing," in *2015 International joint conference on neural networks (IJCNN)*. ieee, 2015, pp. 1–8.

[12] Bodo Rueckauer, Iulia-Alexandra Lungu, Yuhuang Hu, Michael Pfeiffer, and Shih-Chii Liu, "Conversion of continuous-valued deep networks to efficient event-driven networks for image classification," *Frontiers in neuroscience*, vol. 11, pp. 682, 2017.

[13] Emre O Neftci, Hesham Mostafa, and Friedemann Zenke, "Surrogate gradient learning in spiking neural networks: Bringing the power of gradient-based optimization to spiking neural networks," *IEEE Signal Processing Magazine*, vol. 36, no. 6, pp. 51–63, 2019.

[14] Zecheng Hao, Tong Bu, Jianhao Ding, Tiejun Huang, and Zhaofei Yu, "Reducing ann-snn conversion error through residual membrane potential," in *Proceedings of the AAAI conference on artificial intelligence*, 2023, vol. 37, pp. 11–21.

[15] Yangfan Hu, Qian Zheng, Xudong Jiang, and Gang Pan, "Fast-snn: Fast spiking neural network by converting quantized ann," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 45, no. 12, pp. 14546–14562, 2023.

[16] Tong Bu, Wei Fang, Jianhao Ding, PengLin Dai, Zhaofei Yu, and Tiejun Huang, "Optimal ann-snn conversion for high-accuracy and ultra-low-latency spiking neural networks," *arXiv preprint arXiv:2303.04347*, 2023.

[17] Yuhang Li, Shikuang Deng, Xin Dong, Ruihao Gong, and Shi Gu, "A free lunch from ann: Towards efficient, accurate spiking neural networks calibration," in *International conference on machine learning*. PMLR, 2021, pp. 6316–6325.

[18] Kang You, Zekai Xu, Chen Nie, Zhijie Deng, Qinghai Guo, Xiang Wang, and Zhezhi He, "Spikezip-tf: Conversion is all you need for transformer-based snn," *arXiv preprint arXiv:2406.03470*, 2024.

[19] Yizhou Jiang, Kunlin Hu, Tianren Zhang, Haichuan Gao, Yuqian Liu, Ying Fang, and Feng Chen, "Spatio-temporal approximation: A training-free snn conversion for transformers," in *The Twelfth International Conference on Learning Representations*, 2024.

[20] Christoph Stöckl and Wolfgang Maass, "Optimized spiking neurons can classify images with high accuracy through temporal coding with two spikes," *Nature Machine Intelligence*, vol. 3, no. 3, pp. 230–238, 2021.

[21] Jianhao Ding, Zhaofei Yu, Yonghong Tian, and Tiejun Huang, "Optimal ann-snn conversion for fast and accurate inference in deep spiking neural networks," *arXiv preprint arXiv:2105.11654*, 2021.

[22] Tong Bu, Jianhao Ding, Zhaofei Yu, and Tiejun Huang, "Optimized potential initialization for low-latency spiking neural networks," in *Proceedings of the AAAI conference on artificial intelligence*, 2022, vol. 36, pp. 11–20.

[23] Ziming Wang, Shuang Lian, Yuhao Zhang, Xiaoxin Cui, Rui Yan, and Huajin Tang, "Towards lossless ann-snn conversion under ultra-low latency with dual-phase optimization," *arXiv preprint arXiv:2205.07473*, 2022.

[24] Haiyan Jiang, Srinivas Anumasa, Giulia De Masi, Huan Xiong, and Bin Gu, "A unified optimization framework of ann-snn conversion: towards optimal mapping from activation values to firing rates," in *International Conference on Machine Learning*. PMLR, 2023, pp. 14945–14974.

[25] Yuchen Wang, Malu Zhang, Yi Chen, and Hong Qu, "Signed neuron with memory: Towards simple, accurate and high-efficient ann-snn conversion.," in *IJCAI*, 2022, pp. 2501–2508.

[26] Sangwoo Hwang, Seunghyun Lee, Dahoon Park, Donghun Lee, and Jaeha Kung, "Spikedattention: Training-free and fully spike-driven transformer-to-snn conversion with winner-oriented spike shift for softmax operation," *Advances in Neural Information Processing Systems*, vol. 37, pp. 67422–67445, 2024.

[27] Zihan Huang, Xinyu Shi, Zecheng Hao, Tong Bu, Jianhao Ding, Zhaofei Yu, and Tiejun Huang, "Towards high-performance spiking transformers from ann to snn conversion," in *Proceedings of the 32nd ACM International Conference on Multimedia*, 2024, pp. 10688–10697.

[28] Zhikai Li and Qingyi Gu, "I-vit: Integer-only quantization for efficient vision transformer inference," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023, pp. 17065–17075.

[29] William Kahan, "Ieee standard 754 for binary floating-point arithmetic," *Lecture Notes on the Status of IEEE*, vol. 754, no. 94720-1776, pp. 11, 1996.

[30] Qingyan Meng, Mingqing Xiao, Shen Yan, Yisen Wang, Zhouchen Lin, and Zhi-Quan Luo, "Training high-performance low-latency spiking neural networks by differentiation on spike representation," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2022, pp. 12444–12453.

[31] Shikuang Deng, Yuhang Li, Shanghang Zhang, and Shi Gu, "Temporal efficient training of spiking neural network via gradient re-weighting," *arXiv preprint arXiv:2202.11946*, 2022.

[32] Man Yao, Guangshe Zhao, Hengyu Zhang, Yifan Hu, Lei Deng, Yonghong Tian, Bo Xu, and Guoqi Li, "Attention spiking neural networks," *IEEE transactions on pattern analysis and machine intelligence*, vol. 45, no. 8, pp. 9393–9410, 2023.

[33] Chenlin Zhou, Liutao Yu, Zhaokun Zhou, Zhengyu Ma, Han Zhang, Huihui Zhou, and Yonghong Tian, "Spikingformer: Spike-driven residual learning for transformer-based spiking neural network," *arXiv preprint arXiv:2304.11954*, 2023.

[34] Man Yao, Jiakui Hu, Zhaokun Zhou, Li Yuan, Yonghong Tian, Bo Xu, and Guoqi Li, "Spike-driven transformer," *Advances in neural information processing systems*, vol. 36, pp. 64043–64058, 2023.

[35] Yudong Li, Yunlin Lei, and Xu Yang, "Spikeformer: A novel architecture for training high-performance low-latency spiking neural network," *arXiv preprint arXiv:2211.10686*, 2022.

[36] Chenlin Zhou, Han Zhang, Zhaokun Zhou, Liutao Yu, Liwei Huang, Xiaopeng Fan, Li Yuan, Zhengyu Ma, Huihui Zhou, and Yonghong Tian, "Qkformer: Hierarchical spiking transformer using qk attention," *arXiv preprint arXiv:2403.16552*, 2024.

[37] Chen Li, Lei Ma, and Steve Furber, "Quantization framework for fast spiking neural networks," *Frontiers in Neuroscience*, vol. 16, pp. 918793, 2022.

[38] Ziqing Wang, Yuetong Fang, Jiahang Cao, Hongwei Ren, and Renjing Xu, "Adaptive calibration: A unified conversion framework of spiking neural networks," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2025, vol. 39, pp. 1583–1591.

[39] Chang Liu, Jiangrong Shen, Xuming Ran, Mingkun Xu, Qi Xu, Yi Xu, and Gang Pan, "Efficient ann-snn conversion with error compensation learning," *arXiv preprint arXiv:2506.01968*, 2025.

[40] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al., "An image is worth 16x16 words: Transformers for image recognition at scale," *arXiv preprint arXiv:2010.11929*, 2020.

[41] Timothée Darcet, Maxime Oquab, Julien Mairal, and Piotr Bojanowski, "Vision transformers need registers," *arXiv preprint arXiv:2309.16588*, 2023.

[42] Karen Simonyan and Andrew Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[43] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[44] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE conference on computer vision and pattern recognition*. Ieee, 2009, pp. 248–255.

[45] 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," *arXiv preprint arXiv:1907.11692*, 2019.

[46] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al., "Language models are unsupervised multitask learners," *OpenAI blog*, vol. 1, no. 8, pp. 9, 2019.

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

[48] Changze Lv, Tianlong Li, Jianhan Xu, Chenxi Gu, Zixuan Ling, Cenyuan Zhang, Xiaoqing Zheng, and Xuanjing Huang, "Spikebert: A language spikformer trained with two-stage knowledge distillation from bert," 2023.

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

[50] Peter L Bartlett, Nick Harvey, Christopher Liaw, and Abbas Mehrabian, "Nearly-tight vc-dimension and pseudodimension bounds for piecewise linear neural networks," *Journal of Machine Learning Research*, vol. 20, no. 63, pp. 1–17, 2019.

[51] Vladimir N Vapnik, "The nature of statistical learning," *(No Title)*, 1998.

[52] Ronald A DeVore and George G Lorentz, *Constructive approximation*, vol. 303, Springer Science & Business Media, 1993.

APPENDIX

A. Proof for Theorem 1

Theorem 1 (FS Neuron Error Bound)**.** *For a target activation function $f : [a, b] \to \mathbb{R}$ and FS neuron output $\hat{f}_T^{(M)}(x)$ with T timesteps trained on M samples, the total approximation error satisfies:*

$$\varepsilon \leq O\left(\underbrace{\sqrt{\frac{T\log T\log M}{M}}}_{\textit{Empirical Gap}} + \underbrace{\frac{\mathcal{L}_f|y|_{max}}{T}}_{\textit{Parametric Gap}} + \underbrace{\frac{\|d\|_1}{T}}_{\textit{Quantization Gap}}\right),$$

where $\varepsilon = \mathbb{E}[|f(x) - \hat{f}_T^{(M)}(x)|]$, $\mathcal{L}_f$ is the Lipschitz constant, $|y|_{max}$ is the maximum function value, and $\|d\|_1$ is the L_1 norm of spike intensities.

Proof. We decompose the total error into three gaps:

- The **Empirical Gap** between f and the trained FS neuron $\hat{f}_T^{(M)}$, caused by limited data M and model complexity scaling with T;
- The **Parametric Gap** between f and the optimal FS representation $\hat{f}_T$, due to finite time steps and limited expressiveness;
- The **Quantization Gap** between $\hat{f}_T$ and the final discrete spike-based implementation, caused by temporal quantization and spike binarization.

a) Empirical Gap.: We analyze the VC-dimension of the FS neuron function class. Although FS neurons have $3T$ learnable parameters, their function form is constrained by the spike generation mechanism.

Lemma 1 ([50])**.** *For deep neural networks with arbitrary piecewise linear activation function where W is the number of weights and L is the number of layers, its VC-dimension is bounded by $\Omega(WL\log(W/L))$ and $\mathcal{O}(WL\log(W))$.*

Applying the lemma with $W = 3T$ and $L = 1$, thus $d_{VC} = \mathcal{O}(W\log W) = \mathcal{O}(T\log T)$. According to the standard generalization bound [51], the empirical error over a training set of M samples satisfies:

$$\begin{aligned}\varepsilon_{\text{emp}} &= O\left(\sqrt{\frac{\text{VC}(\mathcal{F}_T)\log\frac{M}{\text{VC}(\mathcal{F}_T)}}{M}}\right)\\ &= O\left(\sqrt{\frac{T\log T\log M}{M}}\right).\end{aligned} \tag{A.1}$$

b) Parametric Gap.: We adapt the classical piecewise constant approximation theory:

Lemma 2 ([52])**.** *For a Lipschitz continuous function f on $[a, b]$ with Lipschitz constant $\mathcal{L}_f$, the optimal piecewise constant approximation error with T segments is bounded by $O(\mathcal{L}_f(b - a)/T)$.*

For FS neurons, the spike intensities $d[t]$ must be selected to span the target function range $[y_{\min}, y_{\max}]$. The discrete spike representation introduces a quantization error that is proportional to the function amplitude. As a result, the overall approximation error can be bounded as follows:

$$\varepsilon_{\text{param}} = O\left(\frac{\mathcal{L}_f|y|_{\max}}{T}\right). \tag{A.2}$$

c) Quantization Gap.: The quantization error is caused by representing the continuous-time FS neuron model in discrete time. In the ideal continuous-time case, the output would be:

$$f_{\text{ideal}}(x) = \int_0^T \sum_{t=0}^{T-1} d[t]\cdot s_t(\tau)\,d\tau, \tag{15}$$

where $s_t(\tau)$ is the continuous-time spike indicator function. The discrete-time approximation is:

$$\hat{f}_T(x) = \sum_{t=0}^{T-1} d[t]\cdot s[t], \tag{16}$$

where $s[t] \in \{0, 1\}$ at discrete timestep t.

Each time step introduces an error of at most $|d[t]|/T$ due to temporal discretization. Summing over all T steps gives:

$$\varepsilon_{\text{quant}} \leq \sum_{t=0}^{T-1}\frac{|d[t]|}{T} = \frac{\|d\|_1}{T}. \tag{A.3}$$

Combining Eqs. (A.1)–(A.3) completes the proof. □

B. Proof for Theorem 2

Theorem 2 (MBE Neuron Error Bounds)**.** *For a target activation function $f : [a, b] \to \mathbb{R}$ and MBE neuron output $\hat{f}_{N,T}^{(M)}(x)$ with N basis components, T timesteps trained on M samples, the total approximation error satisfies:*

$$\varepsilon^* \leq O\Bigg(\underbrace{\sqrt{\frac{N\log(N)\log M}{M}}}_{\textit{Empirical Gap}} + \underbrace{\frac{\mathcal{L}_f|y|_{max}}{NT}}_{\textit{Parametric Gap}} + \underbrace{\frac{\|w\|_1|\alpha|\tau_{max}}{T\Delta t}}_{\textit{Quantization Gap}}\Bigg),$$

where $\varepsilon^* = \mathbb{E}[|f(x) - \hat{f}_{N,T}^{(M)}(x)|]$, $\mathcal{L}_f$ is the Lipschitz constant of f, $|y|_{max}$ is the maximum absolute value of the target function, $\|w\|_1$ is the L_1 norm of basis weights, $|\alpha|$ is the scaling parameter, $\tau_{max} = \max_n\{\tau_{d_n}, \tau_{r_n}, \tau_{V_{th_n}}\}$ is the maximum time constant, and Δt is the discrete timestep. □

Proof. Consistent with the FS neuron error analysis, we decompose the total error into three gaps.

a) Empirical Gap.: We analyze the VC-dimension of the function class defined by the MBE neuron. An MBE neuron with N basis components contains $5N$ learnable parameters: three time constants ($\tau_{d_n}, \tau_{r_n}, \tau_{V_{th_n}}$), one discretization step (Δt_n), and one basis weight $w^{(n)}$ per component. Despite the parameter count, the function class is constrained by the structure of the multi-basis spike generation mechanism.

Since the MBE neuron has 5N learnable parameters with constrained function form, applying the Lemma 1 with $W = 5N$ and $L = 1$, we obtain $d_{VC} = \mathcal{O}(N \log N)$.

$$\begin{aligned}\varepsilon_{\text{emp}} &= O\left(\sqrt{\frac{\text{VC}(\mathcal{F}_{N,T})\log\frac{M}{\text{VC}(\mathcal{F}_{N,T})}}{M}}\right) \\ &= O\left(\sqrt{\frac{N\log N\log M}{M}}\right).\end{aligned} \tag{B.1}$$

b) Parametric Gap.: We extend the classical piecewise constant approximation theory to multi-basis representation. The MBE neuron output is given by:

$$\hat{f}_{N,T}(x) = \sum_{n=1}^{N} w^{(n)} \cdot o^{(n)}(T), \tag{17}$$

where each $o^{(n)}(T)$ represents the accumulated output of the n-th basis component.

Applying the classical piecewise constant approximation theory from the previous lemma, the multi-basis structure enables each basis to contribute distinct functional components, effectively increasing the approximation resolution. With N basis components, each optimized independently with T time steps, the combined representation provides enhanced approximation capability compared to single-basis approaches. For MBE neurons, the spike intensities must span the target function range $[y_{\min}, y_{\max}]$, and the discrete multi-basis representation introduces quantization errors proportional to the function amplitude. The multi-basis enhancement yields:

$$\varepsilon_{\text{param}} = O\left(\frac{\mathcal{L}_f|y|_{\max}}{NT}\right). \tag{B.2}$$

c) Quantization Gap.: The quantization error arises from the discrete-time implementation of the continuous-time MBE neuron model with exponential leaky parameters. In the ideal continuous-time setting, the output would be:

$$f_{\text{ideal}}(x) = \sum_{n=1}^{N} w^{(n)} \int_0^T \sum_{t=0}^{T-1} d_n[t] \cdot s_{n,t}(\tau)\, d\tau, \tag{18}$$

where $s_{n,t}(\tau)$ is the continuous-time spike indicator function for the n-th basis at time t. The discrete-time approximation is:

$$\hat{f}_{N,T}(x) = \sum_{n=1}^{N} w^{(n)} \sum_{t=0}^{T-1} d_n[t] \cdot s_n[t], \tag{19}$$

where $s_n[t] \in \{0, 1\}$ is the binary spike output of the n-th basis at discrete timestep t, and $d_n[t] = \alpha \cdot \exp(-t\Delta t/\tau_{d_n})$ follows the exponential parameter update.

For each basis component n, the temporal discretization introduces a quantization error of:

$$\varepsilon_{quant,n} \leq \frac{1}{T}\sum_{t=0}^{T-1}|d_n[t]| = \frac{|\alpha|}{T}\sum_{t=0}^{T-1}\exp(-t\Delta t/\tau_{d_n}). \tag{20}$$

The geometric series sum satisfies:

$$\sum_{t=0}^{T-1}\exp(-t\Delta t/\tau_{d_n}) = \frac{1-\exp(-T\Delta t/\tau_{d_n})}{1-\exp(-\Delta t/\tau_{d_n})} \leq \frac{\tau_{d_n}}{\Delta t}. \tag{21}$$

Combining all basis components with their respective weights:

$$\varepsilon_{\text{quant}} \leq \sum_{n=1}^{N}|w^{(n)}| \cdot \frac{|\alpha|\tau_{d_n}}{T\Delta t} \leq \frac{\|w\|_1|\alpha|\tau_{max}}{T\Delta t}, \tag{B.3}$$

where $\tau_{max} = \max_n\{\tau_{d_n}, \tau_{r_n}, \tau_{V_{th_n}}\}$.

Combining Eqs. (B.1)–(B.3) completes the proof. □

C. Input Distribution of Nonlinear Operations

To provide empirical evidence on the GSO problem, we sample the ImageNet dataset and perform statistical analysis to obtain empirical distributions of inputs to various nonlinear operations. As shown in Fig. 7, both ViT-M/16 and ViT-B/16 architectures exhibit strongly non-uniform input concentration patterns for key nonlinear components, i.e., GELU in MLP and $1/\sqrt{x}$ in LayerNorm. Our measurements show that most of the input values are concentrated in the interval near 0, a critical region where both operations exhibit the largest gradient volatility. GELU exhibits a clear inflection point near x=0. Similarly, $1/\sqrt{x}$ operations exhibit exponential gradient variation as the input approaches zero.

The concentration of input features in high-gradient regions exposes fundamental limitations in the FS neuron's nonlinear approximation. While piecewise linear approximation performs adequately in smooth regions, it systematically underperforms in these critical zones. As shown in Fig. 7, high-gradient regions—despite contributing disproportionately to approximation error—receive insufficient approximation resources, confirming the inherent GSO problem in naive FS neuron implementations.

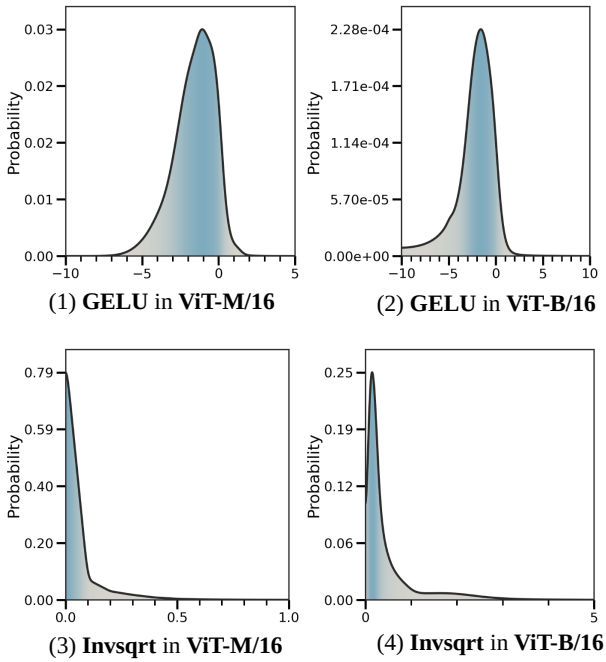


Fig. 7. Input distributions of nonlinear operations GELU and invsqrt in ViT-M/16 and ViT-B/16.

D. FS-based Transformer Conversion

Although FS neurons have shown strong approximation capabilities for simple univariate activation functions, significant performance degradation is observed when directly replacing nonlinear activation functions in Transformer architectures with FS neurons. As shown in Tab. V, directly substituting nonlinear operations in Transformers, including GELU in MLPs, e^x and $1/x$ in Softmax, and $1/\sqrt{x}$ in LayerNorm, using FS neurons with T = 16 configured for minimal approximation loss results in complete network failure, except in the case of GELU. Notably, this error persists and cannot be resolved by increasing the timesteps in SNNs.

TABLE V
ACCURACY LOSS OF VIT-M/16 ON IMAGENET WHEN REPLACING NONLINEARITIES WITH FS NEURONS. FAIL: ACC. 0.10%.

Nonlinear operation	Interval	MSE	Acc loss (%)
$GELU$	[-120,10]	$4.31E-3$	13.20
e^x (Softmax)	[-20,5]	1.75	**Fail**
$1/x$ (Softmax)	[0.1, 10]	$1.01E-3$	**Fail**
$1/\sqrt{x}$ (LayerNorm)	[0.1, 30]	$3.23E-5$	**Fail**

E. Comparison with FS Neuron

E.1 The Count of Learnable Parameters During the process of approximating nonlinear operations using neurons, the FS neuron requires independent updates of three parameters at each time step: the spike firing intensity $d[t]$, reset magnitude $r[t]$, and firing threshold $V_{\text{th}}[t]$. Consequently, the total number of learnable parameters for an FS neuron is $3\mathcal{T}$, which increases linearly with the number of timesteps.

In contrast, for our proposed MBE neuron, we define N as the number of basis. Each basis requires learning five parameters: the corresponding leaky rates τ_{d_n}, τ_{r_n}, $\tau_{V_{\text{th}n}}$, the discretized timestep Δt, and the weight w. Notably, the initial values of the spike intensity $d_n[0]$, reset magnitude $r_n[0]$, and firing threshold $V_{\text{th}_n}[0]$ are determined by the hyperparameter α, the discretized timestep Δt, and the leaky rates τ_{d_n}, and therefore do not require learning. Subsequent updates at each time step follow an exponential decay scheme, eliminating the need for additional learnable parameters.

TABLE VI
COMPARISON OF LEARNABLE PARAMETERS WITH FS AND MBE NEURONS. T: NUMBER OF TIMESTEPS; N: NUMBER OF BASIS.

Operation	FS Neuron	MBE Neuron	Reduction
Parameters	$3T$	$5N$	–
GELU	48	20	58.3%
Tanh	48	20	58.3%
Softmax (e^x)	48	40	16.7%
Softmax ($\frac{1}{x}$)	48	40	16.7%
LayerNorm ($\frac{1}{\sqrt{x}}$)	48	40	16.7%

As shown in Tab. VI, the total number of learnable parameters for an MBE neuron is $5N$, independent of the number of timesteps. In our setting, we fix $T = 16$, and choose $N = 4$ for approximating nonlinear activation functions and $N = 8$ for other nonlinear operations. Accordingly, a single FS neuron requires 48 parameters, whereas an MBE neuron requires only 20 or 40 parameters for the tasks, demonstrating a clear reduction in parameter count.

E.2 Approximation of Nonlinear Operations

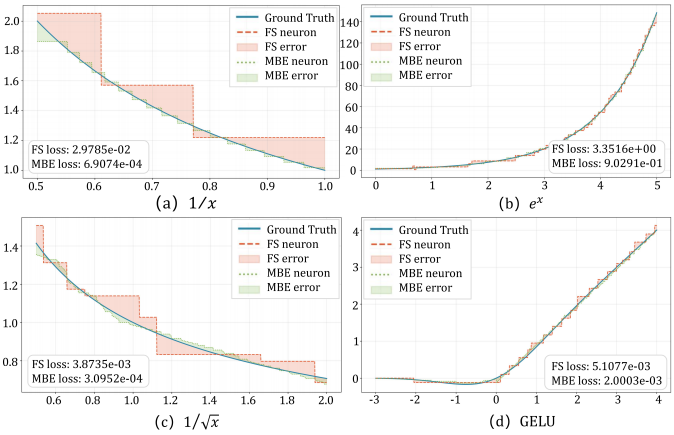


Fig. 8. Comparison of FS and MBE neurons in approximating various nonlinear functions.

To more clearly compare our method with FS neurons in approximating nonlinear operations, we evaluate a series of representative functions, including SiLU, $1/x$, e^x, $\sqrt{1/x}$ and GELU. For the SiLU function, we follow the same evaluation protocol as used in the FS paper, measuring approximation errors separately in key and non-key regions. The comparison results are summarized in Tab. VII, while the results for the other functions are shown in Fig. 8.

F. Conversion Implementation via MBE Neurons

F.1 MBE Neuron for FP Multiplication

TABLE VII
ACCURACY COMPARISON BETWEEN FS AND MBE NEURON IN APPROXIMATING THE SILU FUNCTION ACROSS DIFFERENT INTERVALS. LOWER VALUES INDICATE BETTER APPROXIMATION.

Interval	$[-8, -2]$	$[-2, 2]$	$[2, 12]$
FS	0.0064	0.0023	0.0064
MBE (Ours)	6.97×10^{-5}	0.0006	0.0048
Improvement	(-98.91%)	(-73.91%)	(-25.00%)

Initially, we perform random sampling on the dataset to statistically determine the value ranges of the two activation values involved in floating-point multiplication. As illustrated in the upper portion of Fig. 9, we use MBE neurons to approximate an identity mapping over this range and convert the involved floating-point values into spike-weighted summation forms. The approximate product of the weighted sums is given by:

$$x_1 \cdot x_2 \approx \sum_{i=0}^{T-1} d[i]s[i] \cdot \sum_{j=0}^{T-1} d'[j]s'[j], \tag{22}$$

where $d[t]$ and $d'[t]$ denote the spike intensities at time step t, and $s[t]$ and $s'[t]$ represent the spike events at time step t. Since the intensity values are in floating-point format, directly multiplying the spike-weighted sums still entails floating-point arithmetic, which is not spike-friendly. To address this, we equivalently transform the multiplication expression to decouple the spike terms from the floating-point components. Specifically, we reformulate Eq.(22) as:

$$x_1 \cdot x_2 \approx \sum_{i=0}^{T-1} d[i]d'[i] \cdot \sum_{j=0}^{T-1} s[j]s'[j] \tag{23}$$

$$= \sum_{i=0}^{T-1}\sum_{j=0}^{T-1} d[i]d'[j] \cdot s[i]s'[j]. \tag{24}$$

Based on Eq. 24, the factors of the accumulation terms can be represented as distinct vectors, which are formally defined as follows:

$$\mathbf{i} = [i[0], i[1], \ldots, i[T-1]], \quad \text{where} \quad \mathbf{i} \in \{\mathbf{d}, \mathbf{d}', \mathbf{S}, \mathbf{S}'\}, \tag{25}$$

where $\mathbf{S}$ and $\mathbf{S}'$ represent the temporal spike train of MBE neuron while $\mathbf{d}$ and $\mathbf{d}'$ denote the corresponding spike intensity. As illustrated at the bottom of Fig. 9, the intensity matrix $\mathbf{D}$ is obtained through the outer product of two neuron-specific constant intensity vectors. Similarly, the binary spike matrix $\mathbf{S}$ is formed in an analogous manner and can be expressed as:

$$\mathbf{D} = \mathbf{d}^\top \mathbf{d}', \qquad \mathbf{S} = \mathbf{s}^\top \mathbf{s}'. \tag{26}$$

Subsequently, we compute the element-wise Hadamard product of $\mathbf{D}$ and $\mathbf{S}$, followed by summation to yield the approximated floating-point multiplication result:

$$x_1 \cdot x_2 \approx \sum_{i=0}^{T-1}\sum_{j=0}^{T-1} \mathbf{D}_{\text{ij}} \cdot \mathbf{S}_{\text{ij}} = \sum_{i=0}^{T-1}\sum_{j=0}^{T-1} (\mathbf{D} \odot \mathbf{S})_{ij}. \tag{27}$$

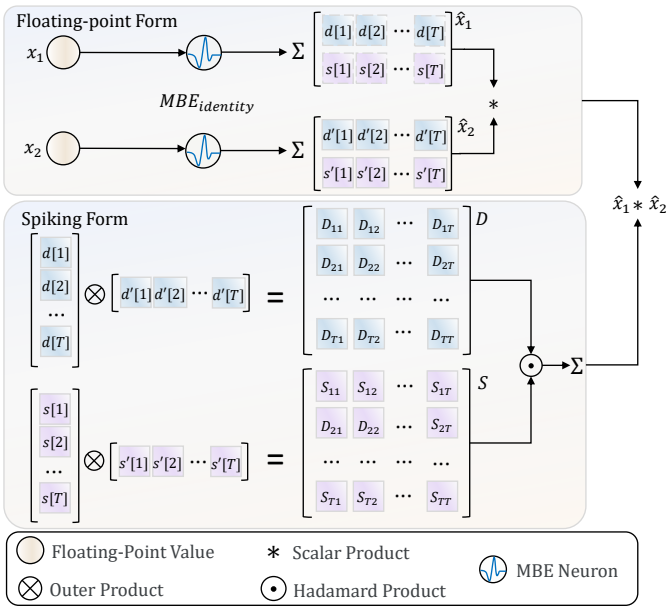


Fig. 9. Approximation of Floating-Point Multiplication Using MBE Neurons. Top: Direct approximation of floating-point multiplication. Bottom: Spike-based approximation reformulated as a Hadamard product.

The spike-driven nature of this process is ensured through two key mechanisms: intensity matrix construction and spike-based operation.

*a) **Intensity matrix construction**:* The weight matrix depends solely on the neuronal spike intensity $d[t]$, which is determined during training. Consequently, $\mathbf{D}$ becomes an input-independent quantity that can be precomputed and stored prior to inference.

*b) **Spike-based operation**:* The Hadamard product involves only floating-point numbers and spikes, eliminating any floating-point-to-floating-point operations.

By acquiring MBE neuron spike intensities, we construct intensity matrix $\mathbf{D}$, while spike matrix $\mathbf{S}$ is obtained through T-step AND operations in inference. Floating-point multiplication is approximated via efficient Hadamard product summation. When activation values originate directly from MBE neurons, Eq. 26 enables direct matrix construction without additional spike-generating neurons.

F.2 MBE Neuron for Softmax As depicted in Tab VIII, the process begins with raw input tokens x_i where $i \in [1, K]$ that undergo five key computational steps. First, the exponential operation e^{x_i} is reformulated using the change-of-base formula $e^{x_i} = 2^{x_i \log_2 e}$ and approximated by MBE neurons specialized for exponential computation. Next, the summed exponential terms $S = \sum_{j=1}^{K} e^{x_j}$ are computed through iterative spike-based summation of individual outputs. The reciprocal operation $\frac{1}{S}$ is then implemented by decomposing it into IEEE 754 floating-point format components, leveraging MBE neurons for both mantissa and exponent 2^{-E} calculations. The final multiplication $y_i = e^{x_i} \cdot \frac{1}{S}$ is performed via spike-based matrix multiplication between the exponential and reciprocal MBE outputs. This complete pipeline culminates in spike-driven Softmax output $\hat{y}_i = \text{Softmax}(x_i)$, achieving analog computation of the probability normalization function through discrete

TABLE VIII
STEPS OF SPIKE-BASED IMPLEMENTATION FOR SOFTMAX

Step	Operation	Formulation	Spike Implementation	Description
0	Input	x_i $(i \in [1, K])$	Raw input tokens	Input sequence to be normalized
1	Exponential	$e^{x_i} = 2^{x_i \log_2 e}$	$e^{x_i} \approx MBE_{\exp}(x_i, T, B)$	Change-of-base decomposition
2	Summation	$S = \sum_{j=1}^{N} e^{x_j}$	$S = \sum_{j=1}^{N} MBE_{\exp}(x_j, T, B)$	Accumulate all exponential outputs
3	Reciprocal	$\frac{1}{S} = \frac{1}{M} \cdot 2^{-E}$	$\frac{1}{S} \approx MBE_{\text{inv}}(M, T, B) \cdot 2^{-E}$	Extract IEEE 754 format
4	Multiplication	$y_i = e^{x_i} \cdot \frac{1}{S}$	$y_i = \text{SpikeMul}(MBE_{\exp}(x_i), MBE_{\text{inv}}(S))$	Spike-based matrix multiplication
5	Output	$\hat{y}_i = \text{Softmax}(x_i)$	Spike-driven Softmax output	Final normalized distribution

spike events while maintaining mathematical equivalence to the continuous Softmax formulation.

F.3 Overall Process Pseudocode See Algorithm 1.

Algorithm 1 Conversion Procedure of Proposed Framework

Input: Pre-trained Transformer ANN model $f_{\text{ANN}}(W)$; Dataset D; Timesteps T and basis B of MBE neurons.
Output: SNN model $f_{\text{SNN}}(W, T, B)$
Step 1: Collect intervals R of MBE neurons
Sample minibatch data from D.
Run the data on f_{ANN} and record input ranges of nonlinear operations in: FP multiplication, nonlinear activation functions, Softmax, LayerNorm.
Step 2: Construct Spiking Modules
Train MBE neurons with timestep T and basis B using collected intervals R.
Build spiking implementations of:
    FP multiplication
    nonlinear activation functions
    Softmax & LayerNorm
Step 3: Model Conversion
for each module m in f_{ANN} modules **do**
    if m is Matrix Multiplication **then**
        Replace m with spiking FP multiplication
    else if m is Activation Function **then**
        Replace m with spiking nonlinear activation
    else if m is Softmax **then**
        Replace m with spiking Softmax
    else if m is LayerNorm **then**
        Replace m with spiking LayerNorm
    end if
end for
return $f_{\text{SNN}}(W, T, B)$

G. Supplementary Experimental Materials

G.1 Implementation Details The number of basis N and timesteps T are determined according to the complexity of the target functions being approximated. For GELU and Tanh approximation, we employ MBE neurons configured with $N = 4$ and $T = 16$. To mitigate approximation errors in identity mapping, all learnable parameters in the MBE neuron are fixed as [20]. Using this configuration with $N = 8$, $T = 16$, we implement function approximations for three distinct mathematical operations: the exponential function 2^x over the interval [0, 1], the reciprocal function $1/x$ within [0.5, 1], and the inverse square root function $1/\sqrt{x}$ across [0.5, 2]. A unified training protocol is adopted, with a fixed learning rate of 0.01 for 200 epochs. To further enhance training stability, an exponential optimizer with a decay rate of 0.99 is used to ensure stable convergence. **G.2 Evaluation of CNN architectures** To further validate the broad applicability of the proposed framework, we evaluate it on two representative CNN architectures: VGG16 and ResNet34. As shown in Tab. IX, the framework incurs only a 0.75% accuracy drop at T=10, demonstrating its effectiveness and competitiveness compared to state-of-the-art (SOTA) methods.

TABLE IX
PERFORMANCE COMPARISON OF DIFFERENT METHODS ON VGG16 AND RESNET34 ARCHITECTURES

Method	Arch.	ANN	T	Acc.[%](Δ)
QCFS [16]	VGG16	74.29	32	68.47 (-5.82)
SRP [14]	VGG16	74.29	16	69.13 (-5.16)
SNM [25]	VGG16	73.18	64	71.50 (-1.68)
Ours	VGG16	**73.37**	**10**	**72.61 (-0.76)**
QCFS [16]	Resnet34	74.23	64	72.35 (-1.88)
ECL [39]	Resnet34	74.36	16	72.37 (-1.99)
AdaFire [38]	Resnet34	75.66	8	72.96 (-2.70)
Ours	Resnet34	**76.31**	**10**	**75.57 (-0.74)**

G.3 Ablation Study

TABLE X
EFFECT OF N AND DECAY IN MBE NEURON. FUNC REPRESENTS "FUNCTION" AND * MEANS MBE NEURONS WITHOUT DECAY.

Func.	N=1	N=2	N=4	N=6	N=8	N=8*
GELU	7.1e-03	4.1e-03	2.3e-04	1.7e-04	1.0e-04	2.8e-01
invsqrt	1.4e-03	1.2e-03	3.1e-04	1.0e-04	4.9e-05	2.8e-03
inv	8.6e-03	2.7e-03	1.1e-03	2.2e-03	4.4e-04	9.8e-03
2^x	8.9e-04	4.5e-04	4.0e-04	2.4e-04	5.3e-05	4.5e-03

The proposed multi-basis encoding architecture employs a hierarchical set of specialized basis kernels, each designed to capture distinct value ranges, thereby enabling high-fidelity approximation through adaptive multi-scale representation. Concurrently, the exponential decay mechanism facilitates progressive refinement via iterative parameter attenuation, effectively mitigating premature saturation while preserving dynamic representation capacity.

To quantitatively evaluate the individual contributions of these components to approximation accuracy enhancement, we conducted comprehensive ablation studies. As demonstrated in Tab. X, systematic expansion of the basis count (N=1→8)

yields significant MSE reduction across all evaluated nonlinear operators (GELU, invsqrt, inv, and 2^x), with invsqrt achieving a remarkable 96.46% improvement. Crucially, ablation of the decay mechanism (N=8 (w/o decay)) results in substantial performance deterioration, unequivocally establishing its fundamental role in ensuring approximation stability and precision.

G.4 Overall Energy Estimation

TABLE XI
FIRING RATES OF MBE NEURON WHEN FITTING VARIOUS FUNCTIONS AT T=16.

Operations	FiringRate(%)
Attention_score	8.31
2^x	46.94
$1/x$	3.74
$1/\sqrt{x}$	25.27
LayerNorm_input_identity	27.61
LayerNorm_1/x_identity	38.46
GELU	38.22

The energy consumption of a single MBE in T time steps is calculated as $E_{MBE} = T \times \eta \times N \times C \times N_h \times EAC$, where T is the timestep, η is the average firing rate, N is the number of tokens, C is the number of token channels, and N_h is the number of heads (1 if it is a non-Attention operation). The energy consumption calculation formula for floating-point multiplication is $E_{FP} = T^2 \times \eta_1 \times \eta_2 \times MO \times EAC$, where η_i is the average firing rate of the MBE neuron i of the spike source, and MO is the number of matrix multiplication operations. Following the energy estimation model, the normalized energy ratio between an SNN module and its ANN equivalent is defined as:

$$\frac{E_{\text{SNN}}}{E_{\text{ANN}}} = \frac{\text{SOPs} \cdot E_{\text{AC}}}{\text{FLOPs} \cdot E_{\text{MAC}}}, \quad E_{\text{MAC}} = 4.6\,\text{pJ}, E_{\text{AC}} = 0.9\,\text{pJ}. \tag{28}$$

For the ViT-M/16 model, we analyze the spike firing rates of MBE neurons across various components during inference at $T = 16$ via random sampling. Subsequently, as summarized in Tab. XI, we calculate the operation counts for each component within a single block. Based on the average energy consumption per block, the energy ratio of the converted SNN compared to the original ANN is 17.29 % according to Eq.(28). The result illustrates that our framework encodes information efficiently with sparse spikes while retaining most of the original ANN accuracy.