Efficient ANN-to-SNN Conversion for Billion-Scale Language Models via Optimized Nonlinear Operator Approximation

Shuyang Chen*, Jiayu Zhang*, Chenhao Ye†, Rui Yan‡, Huajin Tang*
* College of Computer Science and Technology, State Key Laboratory of Brain-Machine Intelligence
Zhejiang University, Hangzhou, China
Email: {sychen.cs, jiayu.zhang, htang}@zju.edu.cn
† School of Mathematical Sciences
Zhejiang University, Hangzhou, China
Email: yechenhao@zju.edu.cn
‡ Zhejiang University of Technology
Email: ryan@zju.edu.cn

***Abstract*—Large Language Models (LLMs) have achieved remarkable performance across a wide range of tasks, but their high computational and energy costs remain a critical challenge. At the same time, Spiking Neural Networks (SNNs) offer exceptional energy efficiency through event-driven asynchronous computation, yet scaling them towards LLMs remains an open problem. We present an optimized ANN-to-SNN conversion framework that successfully scales to 7B-parameter LLMs by addressing the bottleneck of nonlinear operator conversion and quantization error in Transformer architectures. Our method introduces a piecewise optimization method for unary nonlinear operators, replacing gradient-based optimization with interpretable analytical method that achieve superior stability and 65% lower computational overhead compared to baseline methods. Integrated with modern LLM quantization techniques, our converted SNN maintains performance comparable to quantized ANNs while reducing theoretical matrix multiplication energy consumption to 22% of conventional ANNs. This work demonstrates a successful application of ANN-SNN conversion at billion-scale LLM parameters, providing a pathway toward energy-efficient neuromorphic language models.**

***Index Terms*—Spiking Neural Networks, ANN-SNN Conversion, Large Language Models**

I. Introduction

Recent breakthroughs in Large Language Models (LLMs) have demonstrated remarkable capabilities across numerous applications, yet the massive parameter scale and computation cost of LLMs present critical challenges, which makes the computation and energy efficiency of LLMs important research directions. Meanwhile, as a promising neuromorphic computing paradigm [1], Spiking Neural Networks (SNNs) offer exceptional energy efficiency through their event-driven asynchronous computation mechanism. However, scaling SNN over billions of parameters still remains challenging due to difficulty of convergence on larger models.

To mitigate the convergence problem of direct training methods, ANN-SNN conversion methods are proposed to obtain large-scale SNNs efficiently [2]. These conversion methods are built upon equivalence between nonlinear operators in ANNs and SNNs, e.g. equivalence between ReLU activation and the firing rate of Integrate-and-Fire (IF) neurons. In Transformer-based architectures, more nonlinear operators like LayerNorm and attention mechanism should be considered. Some recent work have already discussed the mapping of these operators and obtained vision Transformers by conversion methods [3]–[5]. However, some of the operator mappings remain suboptimal, which are uncapable of larger models.

To address these challenges, we propose an optimized ANN-to-SNN conversion method based on existing research. We find that existing approach for mapping unary operators is suboptimal, and replace it with a more precise approximation, achieving superior stability and a 65% reduction in computational overhead compared to baseline methods. By integrating this approach with state-of-the-art LLM quantization techniques [6]–[8], we successfully scale our conversion framework to a 7B-parameter LLM. Experimental results demonstrate that our converted SNN maintains performance comparable to its quantized ANN counterpart while reducing the theoretical energy consumption of matrix multiplication operations to merely 22% of conventional ANNs. This work represents successful application of ANN-SNN conversion at billion-scale LLM parameters, providing a practical and energy-efficient pathway toward neuromorphic language models.

II. Related Work

A. Spiking Transformers

The rapid advancement of Transformer architectures has made the integration of self-attention mechanisms into Spiking Neural Networks (SNNs) a promising direction for enhancing both the performance and scalability of SNNs. Recently, many variants of spiking-based attention mechanism have been proposed [9], [10], enabling attention mechanism with both the spike-driven property of SNNs and the global interaction of

tokens. Related experiments on vision tasks, including classification, object detection and segmentation, have shown that attention-based SNNs may have superior performance over traditional convolution-based SNNs. Some researches have also explored languange modeling on attention-based SNNs [11]–[13]. However, SNNs have not yet achieved competitive performance on language tasks due to the difficulty of training.

B. ANN-SNN Conversion Methods

ANN-SNN conversion methods obtain SNNs by converting pretrained ANNs to equivalent spiking models, thereby bypassing the convergence difficulties in direct training of large-scale SNNs [2]. Convolution-based SNNs utilize the equivalence of ReLU activation and firing rate of IF neurons for conversion, which mathematically use the spiking neuron as a quantizer [14]. Recently, some methods for Transformer-based ANN-SNN conversion have been proposed [3]–[5], which typically concentrate on conversion of components specific to Transformer architecture, like LayerNorm, SiLU/GELU activations, softmax operators and attention mechanism. Some researches have also proposed specially designed spiking neuron models to reduce the conversion error [15].

C. LLM Quantization Techniques

Since ANN-SNN conversion methods are built on the basis of quantization methods, a network should be effectively quantized before applying conversion methods. However, quantization of LLMs themselves is non-trivial due to the prevalence of significant outliers in their weight and activation distributions [7], which complicates straightforward quantization procedures. This key challenge also potentially hinders the effective scaling of conversion-based approaches to larger Transformer models. Recently, several low-bit quantization methods have been proposed [6], [8], which usually utilize Hadamard transformation and sink tokens to depress the quantization error while adding small amount of computation. In this paper, we establish a quantization framework compatible with SNNs based on PrefixQuant [8], and synergistically integrate it with our ANN-SNN conversion methods, enabling scaling of ANN-SNN conversion to 7B-parameter Llama-2 model [16].

III. Methodology

A. Nonlinear Operator in SNNs

Before delving into the specifics of our conversion methodology, we'd like to discuss a fundamental question: why special handling of nonlinear operators, such as softmax in Transformers, is necessary for SNNs? Even if they are illegal in conversion-based methods, why can't we preserve these operations in directly trained SNNs? Almost all spiking-based Transformer architecture choose to drop the softmax and LayerNorm in the original Transformer for the sake of event-driven property, without further discussion on the algorithmic effect.

Obviously, preserving nonlinear operators in conversion-based SNNs will cause mismatch between activation of ANN and SNN, thus disabling the conversion method. Consider a nonlinear operator F and an input spike sequence $\{s_1, s_2, ..., s_T\}$, $s_i \in \{0, 1\}$. In conversion methods, rate coding is typically used, giving the encoding $\frac{1}{T}\sum_{i=1}^{T} s_i$ in the ANN counterpart. Passing it through the operator will yield $o = F\left(\frac{1}{T}\sum_{i=1}^{T} s_i\right)$. On the other hand, passing $s_i, i = 1, 2, ..., T$ separately in the SNN will yield $\bar{o} = \frac{1}{T}\sum_{i=1}^{T} F(s_i)$ as the rate encoding at the output end. Since F is nonlinear, we have (by Jensen's Inequality):

$$F\left(\frac{1}{T}\sum_{i=1}^{T} s_i\right) \neq \frac{1}{T}\sum_{i=1}^{T} F(s_i). \tag{1}$$

Similar arguments can also be found in [4].

Generalizing to directly trained SNN with the assumption of rate coding, we partition the input spikes into two categories: active, i.e. $\Phi_1 = \{i | s_i = 1\}$, and inactive, i.e. $\Phi_0 = \{i | s_i = 0\}$. We can rewrite the output encoding of SNN as

$$\begin{aligned} \bar{o} &= \frac{1}{T}\sum_{i\in\Phi_0} F(s_i) + \frac{1}{T}\sum_{i\in\Phi_1} F(s_i) \\ &= \frac{1}{T}\sum_{i\in\Phi_0} F(0) + \frac{1}{T}\sum_{i\in\Phi_1} F(1) \\ &= F(1)\cdot\lambda + F(0)\cdot(1-\lambda), \end{aligned} \tag{2}$$

where $\lambda \in 0, 1/T, 2/T, ..., 1$ is the firing rate. In other words, the output encoding is only a linear transformation of the input encoding even if the nonlinear operator is applied. As a result, inserting a nonlinear operator directly into a rate-coded SNN is invalid, since it does not provide any nonlinear ability.

To generalize the conclusion beyond rate coding, we prove that

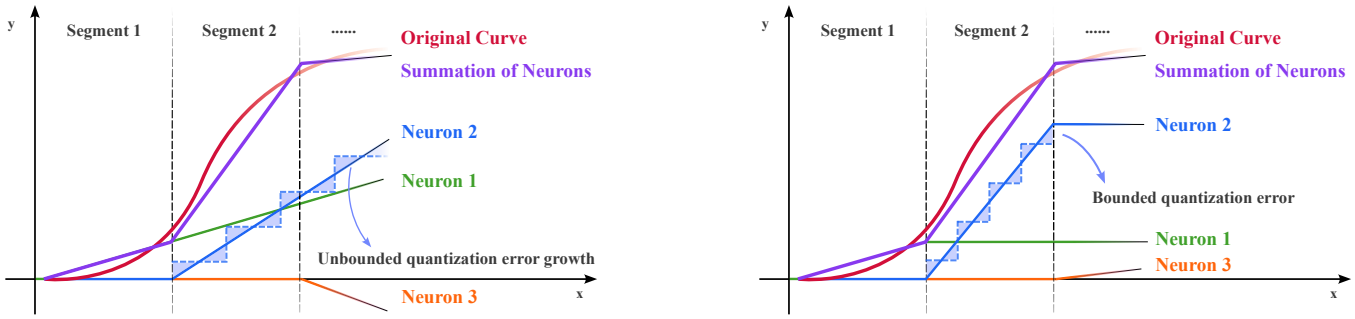
Theorem 1: Let $e(s_1, s_2, ..., s_T) : \mathbb{R}^T \mapsto \mathbb{R}$ be the spike encoding function. If e is a linear function, then $e(F(s_1), F(s_2), ..., F(s_T))$ is always a linear transformation of $e(s_1, s_2, ..., s_T)$ (i.e. $\exists c, d \in \mathbb{R}$ which do not rely on $s_i, i = 1, 2, ..., T$, so that $e(F(s_1), F(s_2), ..., F(s_T)) = ce(s_1, s_2, ..., s_T) + d$), for any nonlinear operator F.

The proof of Theorem 1 can be found in Appendix A. Note that a linear encoder function e includes a large set of encoding functions, including timing-dependent encoding methods like binary encoding. In conclusion, preserving a nonlinear operator in SNNs is usually invalid, since it actually performs a linear transformation on the spike representation.

For Transformers, the nonlinear operators can be classified into two categories: unary operators, e.g. SiLU, softmax; binary operators, e.g. attention mechanism. In the following sections, we discuss the conversion of these two classes of operators in turn.

B. Optimized Piecewise Approximation for Unary Operators

In the Transformer architecture, unary operators are always scalar functions or composition of scalar functions. To convert the nonlinear operator $f(x) : \mathbb{R} \mapsto \mathbb{R}$ into its spiking



(a) ReLU-based fitting.

(b) Clip-based fitting.

Fig. 1. Comparison between behavior of ReLU-based neurons and Clip-based neurons when fitting a curve. The quantization curve for one neuron is indicated using dashed lines. The unbounded ReLU curve keeps growing beyond effective range, leading to growth of quantization error.

counterpart, STA [3] proposed to approximate it with a two-layer MLP (which we refer to as the surrogate network in the following), i.e.

$$f(x) \approx \boldsymbol{W}_2(ReLU(x\boldsymbol{W}_1 + \boldsymbol{b}_1)) + b_2, \tag{3}$$

where $\boldsymbol{W}_1 \in \mathbb{R}^{1\times d}, \boldsymbol{W}_2 \in \mathbb{R}^{d\times 1}, \boldsymbol{b_1} \in \mathbb{R}^d, b_2 \in \mathbb{R}$. The MLP is trained to approximate $f(x)$ and then converted to SNN, so that the equivalence is ensured. On the other hand, with this approach, we must account for both the approximation error and the additional error introduced during the conversion to SNN. In STA, the network is directly optimized with gradient descent and randomly sampled data. However, we find that this implementation leads to a suboptimal SNN. In larger-scale Transformer models with higher precision requirements, this approach lead to significant performance degradation, or requires to expand the surrogate network with larger d, adding to computation cost.

Note that approximating a scalar function with ReLU-based network is essentially piecewise linear approximation, and both fitting and conversion loss should be jointly considered, we propose a new approximation framework as follows:

1) Choice of Nonlinear Function: The conversion error, mainly quantization error, is determined by the output range of the neuron. Under fixed time steps, a larger output range will lead to larger quantization error. For ReLU-based surrogate networks, the output of a hidden neuron is an unbounded and scaled ReLU curve (Fig. 1a), which will cause large output range of neuron, especially when the fitting a wide input range. Actually, a ReLU-based surrogate network is mathematically equivalent to a Clip-based surrogate network, with $\text{Clip}(x) = \max(\min(x, 1), 0)$, which is strictly bounded (Fig. 1b). Quantitative comparison can be found in Fig. 3.

2) Choice of Optimization Method: While gradient optimization with respect to fitting loss is direct and simple, it remains less competitive in performance. The two-layer surrogate network is simple enough to have an analytic solution, which allows for finer control over the optimization process. With d hidden neurons, the network approximates the nonlinear function $f(x)$ with the superposition of d scaled Clip curve $g_i(x) = (\boldsymbol{W}_2)_i\text{Clip}((\boldsymbol{W}_1)_i x + (\boldsymbol{b}_1)_i) + b_2$. We denote that effective range of $g_i(x)$ as $[-\frac{(\boldsymbol{b}_1)_i}{(\boldsymbol{W}_1)_i}, \frac{1-(\boldsymbol{b}_1)_i}{(\boldsymbol{W}_1)_i}]$, where $0 \le (\boldsymbol{W}_1)_i x + (\boldsymbol{b}_1)_i \le 1$. Assuming that the effective range of the d neurons do not overlap[1], we can reformulate this optimization problem as a piecewise linear approximation with d segments. Under this formulation, we change the optimization variables to the $d+1$ endpoints of the segments $\{(x_0, y_0), (x_1, y_1), ..., (x_d, y_d)\}$. After that, we use gradient descent to solve the variables. We decompose the total loss into fitting loss and quantization loss:

$$\begin{aligned} L &= L_{fit} + L_{quant} \\ &= \sum_{i=1}^{d} \int_{x_{i-1}}^{x_i} (g_i(x) - f(x))^2 dx \\ &\quad + \sum_{i=1}^{d} \int_{x_{i-1}}^{x_i} (Q_i(x) - g_i(x))^2 dx, \end{aligned} \tag{4}$$

where $g_i(x)$ is the i th segment, $Q_i(x)$ is the T-step quantized version of $g_i(x)$, formulated as

$$\begin{aligned} Q_i(x) &= \left(k + \frac{1}{2}\right) \cdot \frac{y_i - y_{i-1}}{T}, \\ \text{when } x &\in \left[k \cdot \frac{x_i - x_{i-1}}{T}, (k+1) \cdot \frac{x_i - x_{i-1}}{T}\right], \\ k &= 0, 1, ..., T-1. \end{aligned} \tag{5}$$

The L_{quant} is solved as

$$L_{quant} = \sum_{i=1}^{d} \frac{(x_i - x_{i-1})(y_i - y_{i-1})^2}{12T^2}. \tag{6}$$

For L_{fit}, as $f(x)$ is arbitrary, it's hard to find the analytic solution. Therefore we use numerical integration as a substitution. In this paper, we use the Simpson's rule as an approximation. Let $h_i(x) = (g_i(x) - f(x))^2$:

[1]The assumption is for theoretical analysis. Since all unary operators except SiLU in this paper are monotonic and convex, but the superposition of two scaled Clip functions which have overlapping effective range is either non-monotonic or non-convex (which is provable by enumeration), the assumption is usually effective. For non-convex functions like SiLU, the assumption leads to a constrained solution which might not be mathematically optimal.

$$L_{fit} \approx \frac{1}{6}\sum_{i=1}^{d}(x_i - x_{i-1})(h_i(x_{i-1}) + 4h_i(x_{i-\frac{1}{2}}) + h_i(x_i)), \tag{7}$$

where $x_{i-\frac{1}{2}} = (x_i + x_{i-1})/2$. By integrating Eq. 6 and 7 into Eq. 4, we use gradient descent to solve the variables. Notably, the optimization process no longer needs sampling data, but directly solved with respect to optimization variables.

After optimization, we convert the variables into weights of the surrogate network as

$$\begin{aligned}&(\boldsymbol{W}_1)_i = \frac{1}{x_i - x_{i-1}}, (\boldsymbol{b}_1)_i = \frac{x_{i-1}}{x_i - x_{i-1}},\\ &(\boldsymbol{W}_2)_i = y_i - y_{i-1}, b_2 = y_0.\end{aligned} \tag{8}$$

Compared to directly applying gradient optimization methods on the surrogate network, the proposed optimization is more intuitive and controllable, with significantly improved fitting ability. However, activation distributions in LLMs are typically highly dispersed [7], implying that nonlinear operators must accommodate an exceptionally wide input range. Even with the proposed optimization strategy, directly fitting such a broad range with a compact surrogate network remains infeasible. In the following section, we'll show how can we compress the activation in LLMs for efficient approximation.

C. Range Compression of Activation in LLMs

A compact surrogate network can only achieve sufficient approximation accuracy within a limited input range. To meet this requirement, we adopt different strategies for different nonlinear operators in LLMs:

1) Nonlinear Operator in MLPs: In the MLP of Transformers, nonlinear function like SiLU, GELU are used. All these functions converge to 0 when $x \to -\infty$, so we can safely truncate the left endpoint of the fitting interval and set all values below this threshold to 0. In the case of SiLU activation in Llama-2 7B, we reset the value in $[-\infty, -5]$ to 0, which is equivalent to fix (x_0, y_0) as $(-5, 0)$.

Moreover, the wide input range of these functions is actually attributed to the presence of outliers. In practice, the vast majority of activations are concentrated near $x = 0$, to incorporate this observation into the optimization process, we weight the integration intervals in Eq. (4)–(7) according to the empirical activation distribution sampled from the LLM:

$$\begin{aligned}L &= L_{fit} + L_{quant}\\ &= \sum_{i=1}^{d}\beta_i\int_{x_{i-1}}^{x_i}(g_i(x) - f(x))^2 dx\\ &\quad + \sum_{i=1}^{d}\beta_i\int_{x_{i-1}}^{x_i}(Q_i(x) - g_i(x))^2 dx.\end{aligned} \tag{9}$$

2) Softmax Operator: We use the numerically stabilized version of softmax, i.e.

$$\text{softmax}(\boldsymbol{x}) = \frac{\exp(\boldsymbol{x} - \max \boldsymbol{x})}{\sum_{i=1}^{d}\exp(x_i - \max \boldsymbol{x})} \triangleq \frac{\exp \boldsymbol{x}'}{\sum_{i=1}^{d}\exp \boldsymbol{x}'_i}. \tag{10}$$

As the implementation in STA [3], we regard the softmax operator as combination of exponential (exp) and inverse (inv) operator. To compress the input range of exp operation, we utilize the sparse nature of attention map. With $\boldsymbol{x}'_i < -5$, the impact of $\boldsymbol{x}'_i$ on the attention map is less than 0.67% ($\exp(-5) \approx 0.0067$), thus can be discarded and reset to 0. In practice, we truncate the range of $\boldsymbol{x}'$ to $[-5, 1]$.

For the inv operator, we again scale the softmax operator to control the input range:

$$\text{softmax}(\boldsymbol{x}) = \frac{\exp \boldsymbol{x}'}{\sum_{i=1}^{d}\exp \boldsymbol{x}'_i} = \frac{\lambda \exp \boldsymbol{x}'}{\sum_{i=1}^{d}\lambda \exp \boldsymbol{x}'_i}. \tag{11}$$

In practice, λ is set to 0.1 with fitting range $[0.1, 40]$.

3) RMSNorm Operator: The operator to be replaced in RMSNorm is the sqrt-inv operator $f(x) = 1/\sqrt{x}$. Since there is a significant shift on input to RMSNorm in different layers of LLM, we utilize the scale invariance of RMSNorm operation to normalize the input distribution across layers, i.e. $\text{RMSNorm}(\boldsymbol{x}) = \text{RMSNorm}(\lambda \boldsymbol{x})$. We use different scaling factor γ in different layers to ensure stable input range after scaling. The original activation distributions are sampled from the LLM using a small number of calibration samples.

D. Conversion of Binary Operators

Binary nonlinear operators in Transformers include dynamic matrix interaction in attention mechanism and normalization in softmax and RMSNorm/LayerNorm operations. The conversion of these operators has already been implemented in STA [3] and ECMT [4], where a spike-based approximation algorithm for matrix multiplication is used. In this work, we directly adopt these existing algorithms. We implement the algorithm in a parallelized way for efficiency on GPUs.

E. Integration with LLM Quantization Framework

After addressing the conversion of all nonlinear operators in the LLM, we integrate our conversion methods with state-of-the-art LLM quantization framework to enable efficient ANN-SNN conversion under short time steps. We build the quantization framework upon PrefixQuant [8], which includes:

1) Hadamard Transform: Some pairs of Hadamard transform are inserted into the architecture to reduce the quantization difficulty of the model while maintaining equivalence with the original model. Some of the transforms can be merged into the weights, which we omit in Fig. 2. We consider the method as compatible for the paradigm of SNNs since

- The randomized Hadamard transform used in the framework is a linear operation whose weights consist only of 1 and -1, involving only addition operations, and is a extensively validated hardware-friendly algorithm.
- The Hadamard transform is a lightweight operation and admits fast algorithms with logarithmic time complexity, i.e. can be computed in $\mathcal{O}(\log d)$ time for $x \in \mathbb{R}^d$.

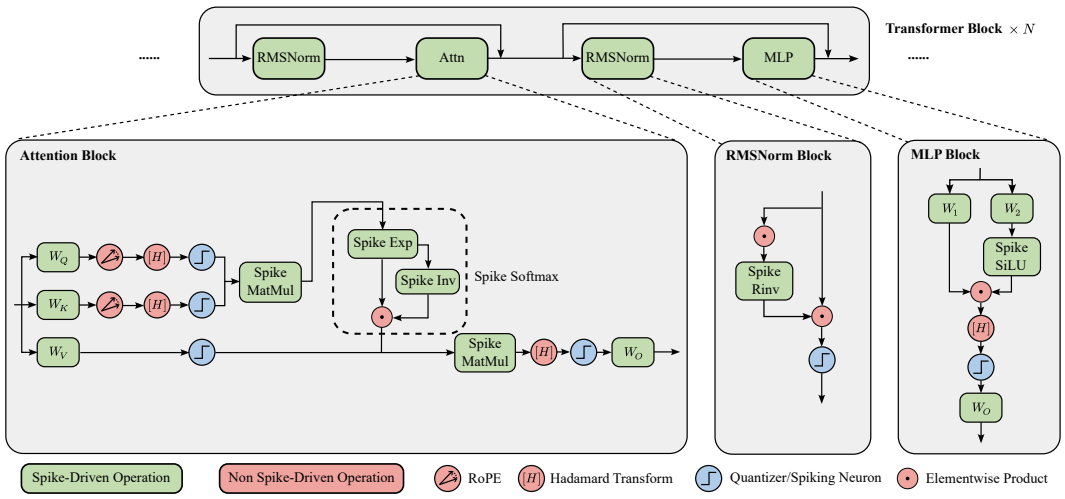


Fig. 2. Architecture of the model obtained through ANN-SNN conversion. Based on the original Llama-2 architecture [16], the Hadamard transforms and quantizers are introduced by quantization framework, and the spiking nonlinear operators (Spike Softmax, Spike Rinv, Spike SiLU and Spike Matmul) are introduced by methods in Sec. III-B- III-D

2) Prefix Tokens: To alleviate the impact of sink tokens, a set of prefix tokens is added before the input prompt as a prefix KV cache in each layer. These prefix can be seen as fixed weights in conversion as they will not change in inference.

On the basis of PrefixQuant [8], we quantize the Q matrix in attention to be compatible with SNN. A grid search over the quantization parameters of Q and K is performed with respect to the quantization loss to determine quantization scales.

IV. Experiments and Results

A. Results on General LLM Benchmarks

We evaluate our proposed SNN conversion approach on the Llama-2 7B architecture with W4A4KV4 static quantization, converted to SNN with $T = 8$. Detailed experimental setup and hyperparameters are provided in the Appendix B.

Table I presents the language modeling performance and comprehensive evaluation across standard LLM benchmark tasks. While spiking model exhibits some performance degradation compared to the quantized baseline (approximately 20% increase in perplexity), the gap is less pronounced on practical benchmarks, where concrete performance on downstream tasks, instead of perplexity, offers a more reliable metric.

B. Ablation of Unary Operator Approximation

To separately evaluation the effectiveness of the proposed piecewise approximation algorithm, we compared different methods by fitting $f(x) = 1/x, x \in [0.1, 20]$ with 8 hidden neurons and $T = 8$. For direct stochastic gradient based optimization (referred to as SGD method), we replicated the experimental setup from STA [3] for fair comparison.

The results are shown in Fig. 3. The fitting curve exhibits sawtooth oscillations due to the unbounded nature of ReLU, leading to suboptimal approximation when quantized. While the curve based on Clip is much smoother, naive gradient-based optimization method tends to converge to a suboptimal solution, compared to our proposed approximation method. As quantitative measure, the MSE loss for ReLU-based SGD method, Clip-based SGD method and our approximation method are 0.0864, 0.0256 and 0.0006, respectively.

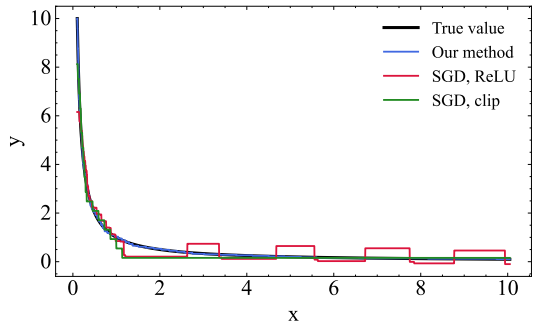


Fig. 3. Comparison of different algorithms when fitting $f(x) = 1/x$.

C. Computational Overhead and Energy Consumption Analysis

We divide the neurons into two categories: neurons obtained from nonlinear operator approximation (i.e. fitting neurons) and through ANN-SNN conversion (i.e. quantization neurons). Fig. 4 shows that layerwise average firing rate of different neuron types. Quantization neurons maintain a low average firing rate (0.11), contributing to energy efficiency. In fitting neurons, the average firing rates for rinv, exp, inv, silu operators are 0.374, 0.020, 0.315 and 0.138, respectively. The rinv and inv operators show higher firing rates, while exp and silu operators benefit from truncation mechanisms that promote sparsity.

For fitting neurons, our approach reduces the neuron count to 25%-50% and time steps to 25% of the baseline while maintaining comparable accuracy, achieving a 65% reduction in computational cost. Notably, we point out that the energy es-

TABLE I
LANGUAGE MODELING PERFORMANCE AND LLM BENCHMARK EVALUATION

Model	WikiText2↓	C4↓	PQ↑	WG↑	HS↑	Arc-e↑	Arc-c↑	LA(ppl)↓	LA(acc)↑
Original	6.29	7.53	77.64	70.32	56.55	76.05	43.86	4.4255	67.03
Quantized	6.88	8.30	77.09	68.82	54.63	74.62	41.04	5.5365	62.95
Spiking (Ours)	8.27	9.60	74.65	63.54	51.41	72.05	39.51	8.4903	54.94

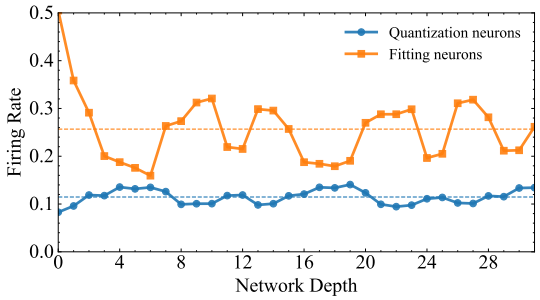


Fig. 4. Average firing rate of neurons in each layer.

timation in baseline is not entirely appropriate, as the surrogate networks cannot be fully fused into previous layers, which introduces non-spiking operations. As a conservative estimation, we compute the energy cost as $E = TNE_{MAC} + TN\eta E_{AC}$, where T is the time step, N is number of hidden neurons, η is the estimated firing rate and E_{MAC} and E_{AC} are energy cost per MAC and AC operation, respectively. The computational cost of nonlinear operations in our spiking model amounts to 95% of that in the ANN counterpart. Notably, this does not imply that the conversion is unnecessary; retaining the original nonlinear operators [5] would result in computational costs T times higher than the ANN baseline. For matrix multiplication operations dominated by quantization neurons, we adopt the estimation method in [3]. Our spiking implementation achieves 22% of the energy consumption of equivalent ANN operations at a firing rate of 0.11, demonstrating the potential for significant energy savings in large-scale deployment.

V. CONCLUSION

In this paper, we presented an optimized ANN-to-SNN conversion framework that successfully scales to billion-parameter large language models. Our contribution includes the piecewise optimization for unary nonlinear operators and integration with modern LLM quantization techniques. The proposed optimization method use interpretable analytical solutions to achieve superior stability and reducing computational overhead by 65% compared to baseline methods. By integrating with modern LLM quantization techniques, we demonstrated a successful application of ANN-SNN conversion at 7B-parameter scale. The converted SNN maintains performance comparable to quantized ANNs while reducing theoretical matrix multiplication energy consumption to merely 22% of conventional ANNs, paving a pathway toward energy-efficient neuromorphic language models.

ACKNOWLEDGMENT

This work was supported by the National Natural Science Foundation of China under grant 32441113, 62276235.

REFERENCES

[1] W. Maass, "Networks of spiking neurons: The third generation of neural network models," *Electron. Colloquium Comput. Complex.*, vol. TR96-031, 1996. [Online]. Available: https://eccc.weizmann.ac.il/eccc-reports/1996/TR96-031/index.html

[2] Y. Li, S. Deng, X. Dong, R. Gong, and S. Gu, "A free lunch from ANN: towards efficient, accurate spiking neural networks calibration," in *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, ser. Proceedings of Machine Learning Research, M. Meila and T. Zhang, Eds., vol. 139. PMLR, 2021, pp. 6316–6325. [Online]. Available: http://proceedings.mlr.press/v139/li21d.html

[3] Y. Jiang, K. Hu, T. Zhang, H. Gao, Y. Liu, Y. Fang, and F. Chen, "Spatio-temporal approximation: A training-free SNN conversion for transformers," in *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024. [Online]. Available: https://openreview.net/forum?id=XrunSYwoLr

[4] Z. Huang, X. Shi, Z. Hao, T. Bu, J. Ding, Z. Yu, and T. Huang, "Towards high-performance spiking transformers from ANN to SNN conversion," in *Proceedings of the 32nd ACM International Conference on Multimedia, MM 2024, Melbourne, VIC, Australia, 28 October 2024 - 1 November 2024*, J. Cai, M. S. Kankanhalli, B. Prabhakaran, S. Boll, R. Subramanian, L. Zheng, V. K. Singh, P. César, L. Xie, and D. Xu, Eds. ACM, 2024, pp. 10 688–10 697.

[5] K. You, Z. Xu, C. Nie, Z. Deng, Q. Guo, X. Wang, and Z. He, "Spikezip-tf: Conversion is all you need for transformer-based SNN," in *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net, 2024. [Online]. Available: https://openreview.net/forum?id=NeotatlYOL

[6] S. Ashkboos, A. Mohtashami, M. L. Croci, B. Li, P. Cameron, M. Jaggi, D. Alistarh, T. Hoefler, and J. Hensman, "Quarot: Outlier-free 4-bit inference in rotated llms," in *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, A. Globersons, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. M. Tomczak, and C. Zhang, Eds., 2024. [Online]. Available: http://papers.nips.cc/paper_files/paper/2024/hash/b5b939436789f76f08b9d0da5e81af7c-Abstract-Conference.html

[7] G. Xiao, J. Lin, M. Seznec, H. Wu, J. Demouth, and S. Han, "Smoothquant: Accurate and efficient post-training quantization for large language models," in *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, ser. Proceedings of Machine Learning Research, A. Krause, E. Brunskill, K. Cho, B. Engelhardt, S. Sabato, and J. Scarlett, Eds., vol. 202. PMLR, 2023, pp. 38 087–38 099. [Online]. Available: https://proceedings.mlr.press/v202/xiao23c.html

[8] M. Chen, Y. Liu, J. Wang, Y. Bin, W. Shao, and P. Luo, "Prefixquant: Eliminating outliers by prefixed tokens for large language models quantization," Oct. 2024, arXiv:2410.05265 [cs.CL].

[9] Z. Zhou, Y. Zhu, C. He, Y. Wang, S. Yan, Y. Tian, and L. Yuan, "Spikformer: When spiking neural network meets transformer," in *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023. [Online]. Available: https://openreview.net/forum?id=frE4fUwz_h

[10] M. Yao, J. Hu, T. Hu, Y. Xu, Z. Zhou, Y. Tian, B. Xu, and G. Li, "Spike-driven transformer V2: meta spiking neural network architecture inspiring the design of next-generation neuromorphic chips," in *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024. [Online]. Available: https://openreview.net/forum?id=1SIBN5Xyw7

[11] R. Zhu, Q. Zhao, G. Li, and J. Eshraghian, "Spikegpt: Generative pre-trained language model with spiking neural networks," *Trans. Mach. Learn. Res.*, vol. 2024, 2024. [Online]. Available: https://openreview.net/forum?id=gcf1anBL9e

[12] J. Zhang, J. Shen, Z. Wang, Q. Guo, R. Yan, G. Pan, and H. Tang, "Spikingminilm: energy-efficient spiking transformer for natural language understanding," *Sci. China Inf. Sci.*, vol. 67, no. 10, 2024.

[13] M. Bal and A. Sengupta, "Spikingbert: Distilling BERT to train spiking language models using implicit differentiation," in *Thirty-Eighth AAAI Conference on Artificial Intelligence, AAAI 2024*, M. J. Wooldridge, J. G. Dy, and S. Natarajan, Eds. AAAI Press, 2024, pp. 10 998–11 006.

[14] Y. Hu, Q. Zheng, X. Jiang, and G. Pan, "Fast-snn: Fast spiking neural network by converting quantized ANN," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 45, no. 12, pp. 14 546–14 562, 2023.

[15] X. Xing, B. Gao, Z. Liu, D. A. Clifton, S. Xiao, W. Zhang, L. Du, Z. Zhang, G. Li, and J. Zhang, "Spikellm: Scaling up spiking neural network to large language models via saliency-based spiking," in *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025. [Online]. Available: https://openreview.net/forum?id=ZadnlOHsHv

[16] H. Touvron, L. Martin, K. Stone, P. Albert, A. Almahairi, Y. Babaei *et al.*, "Llama 2: Open foundation and fine-tuned chat models," Jul. 2023, arXiv:2307.09288 [cs.CL].

[17] Y. Wang, M. Zhang, Y. Chen, and H. Qu, "Signed neuron with memory: Towards simple, accurate and high-efficient ANN-SNN conversion," in *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI 2022, Vienna, Austria, 23-29 July 2022*, L. D. Raedt, Ed. ijcai.org, 2022, pp. 2501–2508.

APPENDIX A
PROOF OF THEOREMS

Here we provide the proof of Theorem 1 in Section III-A. For convenience, we restate Theorem 1 here:

Let $e(s_1, s_2, ..., s_T) : \mathbb{R}^T \mapsto \mathbb{R}$ be the spike encoding function. If e is a linear function, then $e(F(s_1), F(s_2), ..., F(s_T))$ is always a linear transformation of $e(s_1, s_2, ..., s_T)$ (i.e. $\exists c, d \in \mathbb{R}$ which do not rely on $s_i, i = 1, 2, ..., T$, so that $e(F(s_1), F(s_2), ..., F(s_T)) = ce(s_1, s_2, ..., s_T) + d$), for any nonlinear operator F.

Proof: We still classify the input spikes into two categories: active and inactive, i.e., $\Phi_0 = \{i | s_i = 0\}$ and $\Phi_1 = \{j | s_j = 1\}$. Denote the linear encoding function as $e(s_1, s_2, ..., s_T) = \sum_{i=1}^{T} w_i s_i + b$. Note that this form of encoding function already allows for coding related to spike timing, since the weights w_i can differ at different times. Binary encoding is also included in this form ($w_i = 2^{T-i}$).

We have

$$\begin{aligned} e(s_1, s_2, ..., s_T) &= \sum_{i \in \Phi_0} w_i \cdot 0 + \sum_{i \in \Phi_1} w_i \cdot 1 + b \\ &= \sum_{i \in \Phi_1} w_i + b. \end{aligned} \tag{12}$$

For the output of the nonlinear operator, denote $\bar{w} = \sum_i w_i$, we have

$$\begin{aligned} & e(F(s_1), F(s_2), ..., F(s_T)) \\ &= \sum_{i \in \Phi_0} w_i F(0) + \sum_{i \in \Phi_1} w_i F(1) + b \\ &= F(0)(\bar{w} - \sum_{i \in \Phi_1} w_i) + F(1) \sum_{i \in \Phi_1} w_i + b \\ &= (F(1) - F(0))(\sum_{i \in \Phi_1} w_i + b) \\ &\quad + (F(0)\bar{w} + (1 + F(0) - F(1))b) \\ &= (F(1) - F(0))e(s_1, s_2, ..., s_T) \\ &\quad + (F(0)\bar{w} + (1 + F(0) - F(1))b). \end{aligned} \tag{13}$$

Therefore, the output encoding $e(F(s_1), F(s_2), ..., F(s_T))$ is still a linear mapping of the input encoding $e(s_1, s_2, ..., s_T)$. ■

APPENDIX B
DETAILS OF EXPERIMENT SETUP

The conversion is evaluated on Llama-2 7B. The quantization level is selected to be static W4A4KV4 (4 bit weights, 4 bit activations and 4 bit KV cache) to allow short time steps and compatibility with SNNs. The signed IF neuron [17] is used to reduce conversion error. The time step T is 8, which is decided by quantization level and neuron model. To reduce error during asynchronous spike emission, we synchronize the spikes every RMSNorm, Attn and MLP block (Fig. 2). This approach inevitably introduces additional latency, but is indispensable for preserving model accuracy.

For surrogate networks, we use 16 hidden neurons for SiLU operator and 8 hidden neurons for other operators. To ensure that the optimization variables $(x_0, x_1, ..., x_d)$ remain strictly within the prescribed fitting interval, we reparameterize the variables as $(\theta_0, \theta_1, ..., \theta_d)$, and let

$$x_i = x_s + \text{Sigmoid}(\theta_i)(x_e - x_s), i = 0, 1, ..., d. \tag{14}$$

where x_s and x_e are the lower bound and upper bound of the fitting interval. We use the Adam optimizer with learning rate of $1e-2$, and run the optimization process for 8000 iterations.

In Fig. 3, we replicate the configuration in STA [3] for SGD based optimization. We use the Huber loss and SGD optimizer with learning rate of $1e-2$. The MultiStepLR scheduler with milestones=[500, 800] and γ=0.1 is applied. We do random sampling with a ratio of $U(0.1, 20) : U(0.1, 1) = 1 : 7$ to get points and train the network until convergence. In ReLU-based networks, we add a penalty term for quantization loss with ratio of 0.01, and no penalty in Clip-based networks. The configurations above is the optimal settings obtained through iterative refinement. It is worth noting that the SGD-based optimization approach is very sensitive to both the weighting of quantization penalty terms and the distribution of input data during training. This sensitivity results in an imbalance between fitting error and quantization error.