A Spike-Gated Residual Unit for Information Flow Control in Transformers

Anonymous CVPR submission

Paper ID 25

Abstract

Large Language Models deliver state-of-the-art performance but incur substantial computational and energy costs due to their dense, fixed-depth Transformer architectures. We introduce a spike-gated residual unit that enables dynamic depth adaptation in pretrained Transformers without modifying backbone weights. By inserting lightweight spike-gated unit into each residual block and optimizing a sparsity-aware objective with a controllable regularization coefficient $\lambda_{sparsity}$, the model learns to selectively bypass redundant layers during inference. Using a gate-only fine-tuning strategy on TinyLlama-1.1B, our unit reduces average active depth to 76.5%, achieving a 1.31× theoretical efficiency gain while maintaining competitive performance (0.93× of the original zero-shot HellaSwag accuracy compared to the dense TinyLlama-1.1B baseline). Layer-wise analysis reveals an emergent hierarchy in which early and late layers remain critical while intermediate blocks are selectively suppressed under sparsity pressure. These results demonstrate that spike-gated residual unit provides a practical software-level pathway toward energy-aware Transformer inference, enabling controllable trade-offs along an efficiency–intelligence Pareto frontier without requiring full retraining or specialized neuromorphic hardware. To support transparency and reproducibility, the complete implementation is available at https://anonymous.4open.science/r/Gated-Residual-Unit-7BC5/.

1. Introduction

Large Language Models (LLMs) have revolutionized AI but suffer from extreme computational and energy demands. For example, training the 175-billion-parameter GPT-3 required on the order of 175 years on a single high-end GPU. Modern LLMs drive unprecedented capabilities, but their dense, feedforward design means every layer and neuron is active for each token, leading to "wasted computation" on easy inputs and a critical electricity demand for training and inference [8, 13]. This rising energy burden is now recognized as a fundamental challenge: as [13] notes, the "immense electricity demand" of LLMs is an "urgent" obstacle to sustainable deployment. By contrast, the human brain operates on only $\sim 20\,\mathrm{W}$ of power and uses sparse, event-driven spike signals. Spiking Neural Networks (SNNs) seek to mimic this efficiency: when implemented on neuromorphic hardware, SNNs can replace dense multiply-add operations with binary spike computations [28]. SNNs are "an energy efficient alternative to traditional neural networks" under event-driven execution [11]. Driven by these insights, we explore a paradigm shift toward sparse, spiking communication within the Transformer block, aiming to achieve a significant reduction in operational energy footprints while maintaining a high ceiling for linguistic performance.

Recent works have begun exploring spiking Transformers as a path to efficiency. In General Language Understanding Evaluation (GLUE) benchmarks, [26] notes that "SNNs have emerged as a promising candidate for energy-efficient LLM inference," and introduces techniques such as masked spike encodings and analog compute-in-memory that improve accuracy while reducing energy consumption by more than $2\times$ on standard tasks. Similarly, specialized hardware accelerators propose executing Transformer blocks using spiking neurons: a recent design for an "energy-efficient spike transformer accelerator" reports matching the performance of a conventional Transformer while consuming only a fraction of the power [3]. On the software side, a recent review work demonstrates a complete neuromorphic LLM by transforming a 1.5B-parameter model into spike-train dynamics. This architecture eliminates all matrix multiplications and achieves approximately $20\times$ greater energy efficiency compared to GPU baselines [25]. Even in vision tasks, transformer-based SNNs have attained state-of-the-art accuracy. For example, *Meta-SpikeFormer* achieved 80% top-1 accuracy on ImageNet using purely spiking layers, outperforming prior SNN models while relying solely on sparse binary operations [25, 28, 29].

In parallel, a separate line of work studies conditional computation in Transformers. Standard LLMs employ a fixed stack of layers for all inputs; however, multiple studies

CVPR #25

note that this design wastes computation on "trivial" tokens and lacks flexibility for complex reasoning [8]. Techniques such as early exiting, layer skipping, and router-based gating have therefore been developed so that simpler inputs activate fewer layers. For example, mixture-of-experts (MoE) architectures route each token through only a small subset of expert subnetworks. The sparsely-gated MoE model enabled effective models with hundreds of billions of parameters by activating only a limited number of experts per token [21]. Similarly, [8] trains per-layer routers to skip or repeat Transformer blocks, demonstrating that early layers are almost always utilized, middle layers can be skipped for easier tasks, and late layers may be repeated for more difficult inputs. These dynamic-depth approaches yield substantial computational gains. Spike-driven Transformer models, for instance, report approximately 51% faster training and 70% faster inference by exploiting sparsity[28]. In effect, such methods trace a Pareto frontier between accuracy and compute, enabling selection of a "sweet spot" in which easy inputs consume minimal layers while challenging inputs still receive full processing.

The Spike-Gated Residual Unit (S-GRU) bridges the gap between the representational power of high-performance Transformers and the inherent energy efficiency of SNNs. By introducing per-layer gating units into a pretrained Transformer backbone, we enable the model to apply a learned activation threshold that determines whether a specific layer is executed or remains dormant for a given forward pass. This framework implements a form of *learned architectural sparsity* inspired by event-driven computation. Unlike purely theoretical sparsification approaches, we rigorously evaluate our modified architecture against industry-standard dense baselines.

Our contributions are threefold:

1. **S-GRU Architecture:** We introduce the S-GRU, a novel neuromorphic-inspired module that enables dynamic, event-driven layer skipping within pretrained Transformer backbones.
2. **Depth Optimization:** We propose a sparsity-aware training framework using a regularization penalty $\lambda_{sparsity}$, allowing the model to self-optimize its computational depth based on input complexity.
3. **Cross-Domain Validation:** We demonstrate that S-GRU achieves superior efficiency-performance trade-offs against SOTA dense (TinyLlama, Qwen-2.5, Phi-2) and sparse (Lightweight MoE) baselines across WikiText-2, HellaSwag, C4, and LAMBADA.

The remainder of this paper is organized as follows. Section 2 reviews prior work related to SNNs and efficient Transformer architectures. Section 3 details the S-GRU, including the spiking-gated mechanism and the sparsity-aware objective function. Section 4 presents our empirical evaluation, highlighting the trade-offs between language modeling performance and compute depth. Finally, the implications of these findings, future works and limitations are discussed in Section 6, and Section 7 summarizes the paper.

2. Related Works

SNNs have emerged as a compelling alternative for energy-efficient deep learning, leveraging sparse binary signals to minimize computational overhead. Recently, the integration of SNNs with Transformer architectures has yielded significant performance gains in computer vision[22] and robotics[5]. Early efforts, such as those by [30] and [33], laid the foundation for combining the long-range dependency modeling of Transformers with the event-driven nature of SNNs. These architectures typically replace standard Softmax attention and Multi-Layer Perceptrons (MLPs) with spike-driven components utilizing Leaky Integrate-and-Fire (LIF) neurons [16, 32]. The primary advantage of these Spike-based Transformers lies in their hardware-friendly operations; for instance, spike-driven self-attention mechanisms can replace energy-intensive floating-point multiplications with bitwise AND operations and sparse accumulations [23]. Subsequent research has explored architectural refinements to further enhance efficiency, including the use of Spiking Wavelet transforms [4], MoE [34], and specialized spatial-temporal attention modules [15]. Furthermore, advancements in ANN-to-SNN conversion techniques [9] have bridged the performance gap between dense and spiking models. Collectively, these methods achieve substantial efficiency, with reports of faster training and faster inference speeds relative to conventional dense Transformers by maximizing event-driven dataflows.

The extension of the spiking paradigm to LLMs has recently gained momentum as a viable path toward energy-efficient inference. Notable advancements include [25], which mathematically converts 1.5B-parameter Transformers into neuromorphic networks to eliminate matrix multiplications, and [24], which scales SNNs to the LLM regime using saliency-based spiking to maintain performance. Parallel to these structural spiking transitions, the field of conditional computation has explored dynamic depth as a means of reducing computational waste. Recent strategies for adaptive layer skipping [17, 18] and sublayer skipping [7] demonstrate that not all tokens require the full depth of a model, with certain middle layers often being redundant for simpler inputs. Various approaches have emerged to govern these skipping decisions, ranging from residual gates that learn "what layers to skip when" [14] to Markov Decision Policies that enable input-aware dynamic skipping [27]. While some methods investigate the theoretical conditions for skipping in multimodal contexts [6] or focus on architectural efficiency for general inference [19], many require significant retraining or specialized hardware. Recent efforts have introduced continuous gating units, such as the

Evaluator Adjuster Unit and Gated Residual Connections to dynamically modulate feature flow and improve contextual adaptability[2]. However, these mechanisms focus primarily on enhancing representation quality through soft scaling and often require training models from scratch In contrast, our *S-GRU* introduces a lightweight software-level gating solution that retrofits onto pretrained LLMs. By incorporating an explicit penalty $\lambda_{sparsity}$ into the training loss, we leverage spiking-inspired sparsity to navigate a controllable Pareto frontier between inference speed and model accuracy without the need for neuromorphic hardware.

3. Spike-Gated Residual Unit Architecture

The S-GRU architecture introduces a dynamic, input-dependent gating mechanism designed to optimize the computational efficiency of LLMs. Unlike static pruning or fixed layer-dropping techniques, S-GRU adapts its depth per forward pass by evaluating the importance of each residual block in real-time. By augmenting a pre-trained transformer backbone with lightweight spike-gated units, the model learns to selectively bypass layers that contribute minimally to the final output, thereby reducing the total Flops required for inference without necessitating a full retraining of the underlying model weights.

3.1. The Spike-Gated Unit

The core component of our architecture is the spike-gated Unit. In a standard Transformer, each layer l performs a transformation f_l on the input hidden states $\mathbf{x}_l$ ($\mathbf{x}_l \in \mathbb{R}^{B\times dmodel}$) followed by a residual addition:

$$\mathbf{x}_{l+1} = \mathbf{x}_l + f_l(\mathbf{x}_l). \tag{1}$$

As shown in Fig.1, in the S-GRU, the scalar gate $g_l \in [0, 1]$ is produced by a spike-gated unit. The residual connection is modified as follows:

$$\mathbf{x}_{l+1} = \mathbf{x}_l + g_l \cdot (f_l(\mathbf{x}_l) - \mathbf{x}_l). \tag{2}$$

The spike-gated unit, first compresses the input sequence through global average pooling across the sequence dimension to form a context vector $\mathbf{s} \in \mathbb{R}^{d_{model}}$:

$$\mathbf{s} = \frac{1}{B}\sum_{t=1}^{B} \mathbf{x}_{l,t}, \tag{3}$$

where $\mathbf{x}_{l,t} \in \mathbb{R}^{d_{model}}$ represents the hidden state vector at layer l and sequence position t, B denotes the block size (sequence length),($B = 256$). t is the index representing a specific token position within the block, ranging from 1 to B, d_{model} is the hidden dimension of the transformer backbone ($d_{model} = 2048$). $\mathbf{s} \in \mathbb{R}^{d_{model}}$ is the resulting context vector that provides a global summary of the input sequence, serving as the input to the spike-gated unit which utilizes a MLP to determine the dynamic scalar gate.

$$g_l = \sigma((\text{SiLU}(\mathbf{s}\mathbf{W}_1 + \mathbf{b}_1))\mathbf{W}_2 + b_2), \tag{4}$$

where $\mathbf{W}_1 \in \mathbb{R}^{d_{model}\times(h\cdot d_{model})}$ is the weight matrix of the first linear layer, which projects the context vector into a higher-dimensional latent space. $\mathbf{b}_1 \in \mathbb{R}^{h\cdot d_{model}}$ is the bias vector for the first linear layer, h is the hidden multiplier that determines the width of the gate's hidden layer. In our implementation, $h = 2$, resulting in a hidden dimension of $2 \times 2048 = 4096$, SiLU$(\cdot)$ is the Sigmoid Linear Unit activation function (also known as Swish), defined as $f(x) = x \cdot \sigma(x)$ which introduces non-linearity to allow the gate to learn complex skipping patterns, $\mathbf{W}_2 \in \mathbb{R}^{(h\cdot d_{model})\times 1}$ is the weight matrix of the second linear layer, which reduces the hidden representation down to a single scalar logit, $b_2 \in \mathbb{R}$ is the scalar bias for the final layer, initialized to a high value (e.g., 5.0) to ensure the gate starts in an "open" state and $\sigma(\cdot)$ is the Sigmoid function, which ensures that g_l remains within the unit interval, effectively acting as a "soft" skip connection that the model can optimize during the sparsity-aware training phase.

3.2. Sparsity-Aware Training

The training objective of S-GRU is to minimize the standard language modeling loss while simultaneously maximizing the sparsity of the network. To achieve this without degrading the pre-trained capabilities of the model, we employ a Gate-Only fine-tuning strategy.

During this phase, the parameters of the pre-trained transformer backbone θ_{LLM} are frozen, and only the parameters of the spike-gated unit's θ_G are updated:

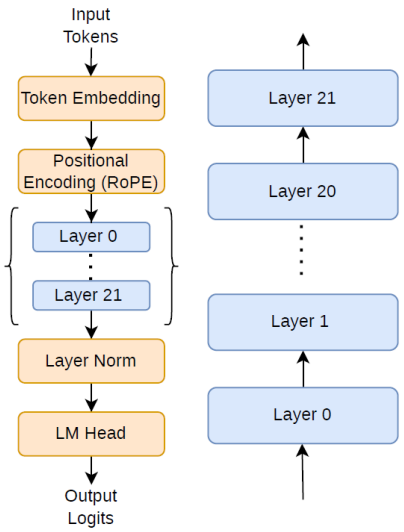
$$\nabla_\theta \mathcal{L} = \nabla_{\theta_G} \mathcal{L}, \quad \text{subject to } \frac{\partial \mathcal{L}}{\partial \theta_{LLM}} = 0, \tag{5}$$

where $\nabla_\theta \mathcal{L}$ is the Total Gradient Vector with respect to all parameters in the network, $\nabla_{\theta_G} \mathcal{L}$ is the Gate Gradient term signifies that the optimizer only computes and applies updates to the parameters within the gated units, specifically the MLP weights and biases $\{\mathbf{W}_1, \mathbf{b}_1, \mathbf{W}_2, b_2\}$, $\frac{\partial \mathcal{L}}{\partial \theta_{LLM}}$ which is the partial derivative of the loss with respect to the transformer backbone. By setting this term to zero, we freeze the original model weights and ∇ is the Gradient Operator, which collects all partial derivatives into a vector.

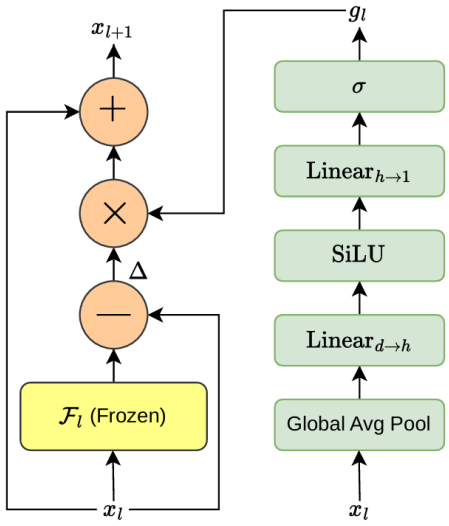
The model is optimized using a composite loss function $\mathcal{L}_{total}$, which introduces a penalty on the gate activations. We define the total objective as:

$$\mathcal{L}_{total} = \mathcal{L}_{LM} + \lambda_{sparsity} \cdot \left(\frac{1}{L}\sum_{l=1}^{L} \bar{g}_l\right), \tag{6}$$

where L is the total number of decoder layers, $\bar{g}_l$ is the mean gate activation for layer l across the batch and sequence,



(a) Overall language model architecture showing the 22 sequential decoder layers (highlighted in light blue), where the S-GRU is applied to the output of each of the decoder layer.



(b) Spiking Residual and dynamic gating module used for each decoder layer.

Figure 1. S-GRU integrated architecture. Residual modification used in each layer: the input x_l is passed through the frozen transformer block $\mathcal{F}_l$ to compute a residual $\Delta = \mathcal{F}_l(x_l) - x_l$, which is modulated by a learned spike gate $g_l \in [0, 1]$ generated by an MLP, producing the output $x_{l+1} = x_l + g_l \cdot \Delta$.

and $\lambda_{sparsity}$ is a hyperparameter controlling the trade-off between model performance and computational efficiency. Let $\mathcal{L}_{LM}$ denote the cross-entropy loss for causal language modeling.

$$\mathcal{L}_{LM} = -\frac{1}{T}\sum t = 1^T \log P(x_t \mid x_{<t}; \theta), \qquad (7)$$

which is the classic Average Negative Log-Likelihood (NLL) over a sequence of tokens. By including the gate values directly in the loss function, the optimizer is incentivized to drive g_l toward zero. Because the sigmoid activation is continuous and differentiable, the gradients from the sparsity penalty flow directly into the spike-gate weights, allowing the model to learn which layers are redundant for the target domain. This formulation ensures that a layer is only bypassed if the resulting increase in $\mathcal{L}_{LM}$ is smaller than the decrease in the sparsity penalty, effectively performing a cost-benefit analysis of each layer's contribution during the forward pass.

3.3. Gate Initialization

The initialization of the spike-gate parameters is critical for stabilizing the transition from a dense pre-trained backbone to a sparse gated architecture. Our MLP configuration consists of four primary learnable parameters: $\theta_G = \{\mathbf{W}_1, \mathbf{b}_1, \mathbf{W}_2, b_2\}$. Standard random initialization (e.g., Xavier or Kaiming) [20] for these parameters typically results in a pre-activation logit $z \approx 0$, yielding gate outputs $g_l \approx 0.5$ at the onset of training. Such a state would immediately bypass 50% of the network's processing capacity, leading to catastrophic forgetting and gradient instability.

To mitigate this, we implement a High-Bias Start strategy specifically targeting the final projection layer. While the input projection parameters $\{\mathbf{W}_1, \mathbf{b}_1\}$ are initialized using standard Kaiming uniformity to facilitate feature discovery, the output weights are set to zero, $\mathbf{W}_2 = \mathbf{0}$, and the bias is initialized as $b_2 = \text{gate\_init\_bias}$. By setting the final weight matrix $\mathbf{W}_2$ to zero, we decouple the gate's output from the input hidden states $\mathbf{s}$ during the initial forward pass. Given the sigmoid activation function $\sigma(z) = (1 + e^{-z})^{-1}$, setting a high initial bias (e.g., $b_2 = 5.0$) ensures that the gate activation g_l saturates near unity:

$$g_l = \sigma((\text{SiLU}(\mathbf{s}\mathbf{W}_1 + \mathbf{b}_1))\mathbf{W}_2 + b_2) = \sigma(0 + 5.0) \approx 0.993. \qquad (8)$$

This initialization ensures that at step 0, the Spikeformer behaves nearly identically to the original dense backbone, preserving its full predictive power. As training progresses, the gradients from the sparsity penalty $\lambda_{sparsity}$ compete with the language modeling loss to pull these biases downward. This allows the model to "prune" layers gracefully, only reducing g_l for a specific layer once the network has learned that the layer's residual contribution is redundant for the target objective.

4. Experiments

4.1. Setup

To evaluate the efficiency and effectiveness of the S-GRU, we perform experiments using a dense-to-sparse transformation on a representative small-scale Large Language Model.

Datasets and Tokenization. Our primary training and evaluation corpuses are *WikiText-2, HellaSwag, C4 and LAMBADA*, tokenized into sequences of 256 tokens using the standard Llama-3 tokenizer. For the sparsity-aware training phase, we utilize a subset of 8,000 training examples and 1,000 validation examples. To evaluate the preservation of downstream reasoning and long-range dependency retention under dynamic depth constraints, we further benchmark the model on the *HellaSwag* commonsense reasoning task (100 zero-shot samples), the *LAMBADA* narrative context dataset, and the *C4* (Colossal Clean Crawled Corpus) validation set.

Model Backbone. We employ TinyLlama-1.1B as our base transformer architecture. This model consists of 22 layers with a hidden dimension of $d_{model} = 2048$. Following the methodology described in Section-3, we inject 22 spike-gate units (one per residual block) utilizing the MLP configuration.

Training Configuration. We train only the gate parameters θ_G for 400 steps using the AdamW optimizer. Based on the stabilization strategy in 3.2, we set the initial gate bias to 3.0. The learning rate is set to 1×10^{-3} with zero weight decay to prevent the gates from being prematurely forced to zero by the optimizer's regularization rather than the sparsity loss. All experiments are conducted in `BFloat16` precision to match the original model's training regime.

Hardware. All experiments were performed on a single NVIDIA RTX 5060 GPU. Given that the language model weights are frozen, the memory overhead remains minimal, requiring less than 7 GB of VRAM during the gating optimization phase.

4.2. Comparative Analysis

We evaluate the S-GRU's performance by comparing its perplexity and zero-shot accuracy against the original dense `TinyLlama-1.1B` backbone and a static layer-dropping baseline. Our results indicate that at a sparsity constraint of $\lambda_{sparsity} = 0.5$, the S-GRU achieves a perplexity of 12.46, representing a controlled degradation of 2.28 units compared to the dense baseline's perplexity of 10.18. Notably, on the *HellaSwag* benchmark, the S-GRU retains 90.9% of the original reasoning performance (41.0% vs. 44.0% baseline), significantly outperforming the static baseline which suffers from a catastrophic drop in accuracy when an equivalent number of layers are removed. This gap suggests that the spike-gate units successfully identify and preserve "critical" layers required for complex reasoning while bypassing redundant computations in simpler contexts, effectively shifting the Pareto frontier of the accuracy-efficiency trade-off.

5. Ablation Study

To characterize the relationship between computational cost and model fidelity, we performed a sweep of the sparsity penalty across $\lambda_{sparsity} \in \{0.0, 0.2, 0.5, 1.0\}$. The observed learning trajectories across these coefficients reveal a distinct, non-linear transition in the emergent architecture of the S-GRU. For $\lambda_{sparsity} = 0.0$, the model retains near-maximal density with an overall mean gate activation of 0.98, successfully preserving the original dense backbone state. As $\lambda_{sparsity}$ increases to 0.2, the gates begin a selective suppression of specific layer blocks, a trend that accelerates significantly at $\lambda_{sparsity} = 0.5$. In this regime, the overall mean gate activation drops to 0.76, and the percentage of layers exceeding the 0.95 activation threshold collapses toward zero. Finally, at $\lambda_{sparsity} = 1.0$, the model reaches an extreme minimalist state, identifying the absolute lower bound of depth required to maintain functional coherence. This shift from dense to sparse processing follows a clear trade-off curve, which is visualized in the sparsity-perplexity relationship shown in Fig. 3. Detailed analysis of the specific layer-wise behaviors and the transition dynamics within this curve is provided in the Discussion section (Section 6).

6. Discussion

Architectural Efficiency The results presented in Table 1 validate that S-GRU achieves a superior balance between linguistic modeling capacity and architectural efficiency compared to established baselines. Most notably, at a mean gate activation of 76.5%, our model achieves a WikiText-2 perplexity of 12.45. This performance effectively matches the density-heavy Qwen-2.5-1.5B (12.49 PPL) and surpasses the larger Phi-2 (13.14 PPL), despite S-GRU utilizing 26% and 59% fewer active parameters per forward pass, respectively. While we observe a marginal degradation in HellaSwag reasoning accuracy (41.0%) relative to the fully-dense TinyLlama baseline (44.0%), the S-GRU compensates with a significant theoretical throughput increase of 1.31×. These findings indicate that the spike-gate units do not merely perform a crude truncation of the network, but instead autonomously re-configure the 1.1B

Table 1. **Comprehensive Evaluation of S-GRU Against Baseline LLMs.** We compare models across general reasoning (HellaSwag), efficiency (Active Layers, Speedup), and linguistic generalization (WikiText-2, C4, and LAMBADA). Our method significantly improves efficiency while maintaining competitive perplexity across diverse corpora.

		General Performance		Efficiency		Perplexity (PPL) ↓		
Model	**Params**	**HellaSwag ↑**	**Wiki-2 ↓**	**Active Layers ↓**	**Speedup ↑**	**C4**	**LAMBADA**	**Avg.**
TinyLlama-1.1B [31]	1.1B	44.0%	10.18	100%	1.00x	11.00	22.30	14.49
Phi-2 [12]	2.7B	52.0%	13.14	100%	0.41x*	16.71	37.02	22.29
Qwen-2.5-1.5B [10]	1.5B	52.0%	12.49	100%	0.73x*	19.03	28.85	20.12
DeepSeek-MoE [1]	1.3B	46.0%	11.26	100%	1.00x*	14.84	23.34	16.48
S-GRU (Ours)	1.1B	**41.0%**	**12.45**	**76.5%**	**1.31x**	**16.50**	**28.68**	**19.21**

*Theoretical speedup relative to TinyLlama-1.1B base throughput.

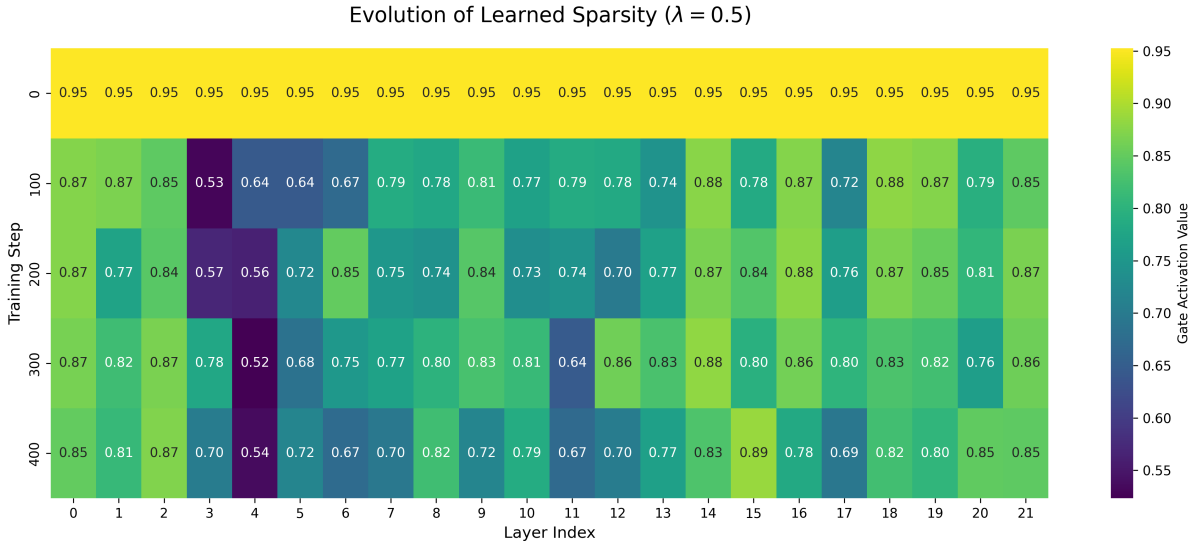


Figure 2. **Evolution of Architectural Sparsity ($\lambda_{sparsity} = 0.5$).** The heatmap displays the mean gate activation for each of the 22 layers over 400 training steps. Darker regions indicate layers the model has learned to bypass, achieving a 23.5% reduction in compute depth.

parameter backbone into a more potent, sparse representation that rivals the performance of significantly larger, fully-dense models.

Cross-Domain Robustness and Generalization. To evaluate the robustness of our sparse representation beyond standard benchmarks, we report linguistic generalization across diverse corpora in Table 1. On the C4 dataset, our framework achieves a perplexity of 16.5, notably outperforming the significantly larger Qwen-2.5-1.5B (19.03 PPL) and maintaining parity with Phi-2 (16.71 PPL). These results suggest that the S-GRU mechanism retains high-fidelity world knowledge even under strict sparsity constraints. Furthermore, in long-range context modeling via the LAMBADA benchmark, our model yields a perplexity of 28.68, successfully bridging the performance gap between established dense baselines and specialized efficient architectures like the Lightweight MoE [1]. While TinyLlama maintains a lead in raw next-token prediction on these datasets, the competitive performance of S-GRU across both general web text and long-context scenarios confirms that the learned spiking dynamics generalize effectively to out-of-distribution data, rather than overfitting to the specific structures of the training corpus.

Layer wise Gate Dynamics. The layer-wise gate dynamics across the tested $\lambda_{sparsity}$ spectrum reveal a structured, non-uniform pruning process that prioritizes specific architectural "anchors." As illustrated in the gate activation heat plot (Fig. 2), the model exhibits an emergent hierarchy of layer importance. At $\lambda_{sparsity} = 0.0$, all 22 layers maintain near-maximal saturation ($\approx 0.95 - 0.98$), confirming that every block remains functionally active. However, as

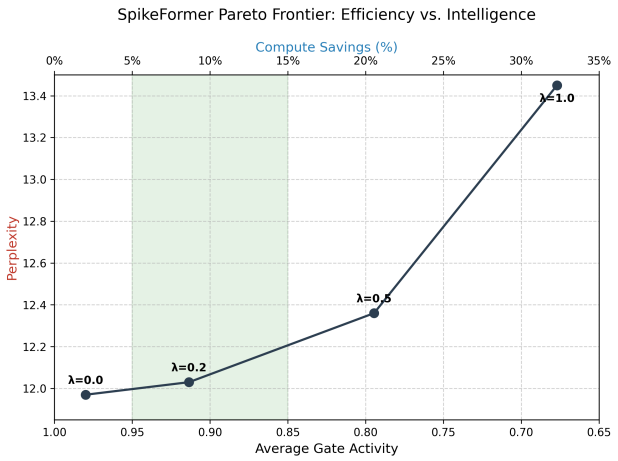


Figure 3. **Efficiency-Intelligence Pareto Frontier for S-GRU.** The plot illustrates the trade-off between model perplexity and average gate activity (compute savings) across varying sparsity constraints ($\lambda_{sparsity}$). The shaded green region highlights the optimal operating range where significant compute savings (5%–15%) are achieved with minimal impact on perplexity. As sparsity increases ($\lambda_{sparsity} = 1.0$), compute savings reach $\sim$33% at the cost of higher perplexity.

the sparsity pressure increases to $\lambda_{sparsity} = 0.2$, we observe the first signs of selective thinning, particularly in the middle-to-late intermediate blocks, while the initial and final layers (such as Layer 0 and Layer 21) remain robustly active. By the critical $\lambda_{sparsity} = 0.5$ threshold, this behavior culminates in a "survivor" architecture: the overall mean gate activation collapses to 0.76, yet the gates consistently sustain higher coefficients for the deep residual units. This indicates that the S-GRU identifies the final stages of the transformer as the most critical for reconciling high-level semantic representations before the language modeling head. In the extreme $\lambda_{sparsity} = 1.0$ regime, even these anchors begin to fade, suggesting that the model has reached its absolute structural limit, operating effectively as a shallow feature extractor rather than a deep reasoning engine.

Limitations and Future Work. Despite the promising theoretical efficiency of the S-GRU, several limitations remain to be addressed in future work. While our model achieves a 1.31× theoretical speedup, realizing these gains on standard deep learning hardware remains a significant challenge. Current GPU kernels are optimized for dense matrix multiplications; consequently, the dynamic skipping of individual residual units often results in negligible wall-clock time savings due to kernel launch overheads and the lack of hardware-level support for conditional branching in massive parallel arrays. To bridge this gap, future research should focus on deploying the developed design on specialized CUDA kernels or hardware-aware compilers that can "short-circuit" the computation of skipped layers at the register level. Furthermore, while the current gating mechanism utilizes a sequence-wide summary, exploring token-level dynamic sparsity could further refine the model's efficiency by skipping computations for "easy" tokens (e.g., punctuation or common stop words). Beyond language modeling, we envision that the S-GRU framework could be generalized to other modalities, such as vision or multimodal transformers, where dynamic depth remains a critical challenge. Finally, extending this framework to larger models, such as Llama-3-70B, would determine if the emergent layer-redundancy observed in TinyLlama scales to architectures with deeper residual hierarchies.

7. Conclusion

In this work, we introduced a S-GRU framework that enables dynamic depth adaptation in pretrained Transformer-based large language models. By integrating lightweight spike-gated units into each residual block and optimizing a sparsity-aware objective with a controllable regularization coefficient $\lambda_{sparsity}$, S-GRU autonomously learns to bypass redundant layers while preserving essential reasoning capacity. Through gate-only fine-tuning on TinyLlama-1.1B, we demonstrated that the model can reduce average active depth to 76.5%, achieving a 1.31× theoretical efficiency gain with minimal degradation in perplexity and zero-shot HellaSwag accuracy. Our empirical analysis reveals a structured layer-importance hierarchy, where early and late blocks remain critical while intermediate layers are selectively suppressed under sparsity pressure. These results validate that learned residual gating provides a practical and stable pathway toward energy-aware Transformer inference without requiring full model retraining or specialized neuromorphic hardware.

References

[1] Damai Dai, Chengqi Deng, Chenggang Zhao, RX Xu, Huazuo Gao, Deli Chen, Jiashi Li, Wangding Zeng, Xingkai Yu, Yu Wu, et al. Deepseekmoe: Towards ultimate expert specialization in mixture-of-experts language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1280–1297, 2024. 6

[2] Sahil Rajesh Dhayalkar. Dynamic context adaptation and information flow control in transformers: Introducing the evaluator adjuster unit and gated residual connections. *arXiv preprint arXiv:2405.13407*, 2024. 3

[3] Congpeng Du, Qi Wen, Zhiqiang Wei, and Hao Zhang. Energy efficient spike transformer accelerator at the edge. *Intelligent Marine Technology and Systems*, 2(1):24, 2024. 1

[4] Yuetong Fang, Ziqing Wang, Lingfeng Zhang, Jiahang Cao, Honglei Chen, and Renjing Xu. Spiking wavelet transformer.

In *European conference on computer vision*, pages 19–37. Springer, 2024. 2

[5] Maoxu Gao, Zhixing Hou, Di Zhu, and Chio-In Ieong. An accuracy-improved low-computational-cost spiking transformer network for robotic arm manipulation. In *2025 IEEE International Conference on Integrated Circuits, Technologies and Applications (ICTA)*, pages 294–295. IEEE, 2025. 2

[6] Max Hartman, Vidhata Jayaraman, Moulik Choraria, Akhil Bhimaraju, and Lav R Varshney. Skip-it? theoretical conditions for layer skipping in vision-language models. *arXiv preprint arXiv:2509.25584*, 2025. 2

[7] Zhuomin He, Yizhen Yao, Pengfei Zuo, Bin Gao, Qinya Li, Zhenzhe Zheng, and Fan Wu. Adaskip: Adaptive sublayer skipping for accelerating long-context llm inference. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 24050–24058, 2025. 2

[8] Ahmed Heakl, Martin Gubri, Salman Khan, Sangdoo Yun, and Seong Joon Oh. Dr. llm: Dynamic layer routing in llms. *arXiv preprint arXiv:2510.12773*, 2025. 1, 2

[9] 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*, pages 10688–10697, 2024. 2

[10] Binyuan Hui, Jian Yang, Zeyu Cui, Jiaxi Yang, Dayiheng Liu, Lei Zhang, Tianyu Liu, Jiajun Zhang, Bowen Yu, Keming Lu, et al. Qwen2. 5-coder technical report. *arXiv preprint arXiv:2409.12186*, 2024. 6

[11] Phu Khanh Huynh, M Lakshmi Varshika, Ankita Paul, Murat Isik, Adarsha Balaji, and Anup Das. Implementing spiking neural networks on neuromorphic architectures: A review. *arXiv preprint arXiv:2202.08897*, 2022. 1

[12] Mojan Javaheripi, Sébastien Bubeck, Marah Abdin, Jyoti Aneja, Sebastien Bubeck, Caio César Teodoro Mendes, Weizhu Chen, Allie Del Giorno, Ronen Eldan, Sivakanth Gopi, et al. Phi-2: The surprising power of small language models. *Microsoft Research Blog*, 1(3):3, 2023. 6

[13] Zhenya Ji and Ming Jiang. A systematic review of electricity demand for large language models: evaluations, challenges, and solutions. *Renewable and Sustainable Energy Reviews*, 225:116159, 2026. 1

[14] Filipe Laitenberger, Dawid Kopiczko, Cees GM Snoek, and Yuki M Asano. What layers when: Learning to skip compute in llms with residual gates. *arXiv preprint arXiv:2510.13876*, 2025. 2

[15] Donghyun Lee, Yuhang Li, Youngeun Kim, Shiting Xiao, and Priyadarshini Panda. Spiking transformer with spatial-temporal attention. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13948–13958, 2025. 2

[16] Yudong Li, Yunlin Lei, and Xu Yang. Spikeformer: Training high-performance spiking neural network with transformer. *Neurocomputing*, 574:127279, 2024. 2

[17] Yijin Liu, Fandong Meng, and Jie Zhou. Accelerating inference in large language models with a unified layer skipping strategy. *arXiv preprint arXiv:2404.06954*, 2024. 2

[18] Xuan Luo, Weizhi Wang, and Xifeng Yan. Adaptive layer-skipping in pre-trained llms. *arXiv preprint arXiv:2503.23798*, 2025. 2

[19] Dheemanth Manur. Adaptive skip for efficient llm inference. Master's thesis, University of California, Riverside, 2025. 2

[20] Yu Pan, Zeyong Su, Ao Liu, Wang Jingquan, Nannan Li, and Zenglin Xu. A unified weight initialization paradigm for tensorial convolutional neural networks. In *International conference on machine learning*, pages 17238–17257. PMLR, 2022. 4

[21] Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *arXiv preprint arXiv:1701.06538*, 2017. 2

[22] Xinyu Shi, Zecheng Hao, and Zhaofei Yu. Spikingresformer: Bridging resnet and vision transformer in spiking neural networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5610–5619, 2024. 2

[23] Ziqing Wang, Yuetong Fang, Jiahang Cao, Qiang Zhang, Zhongrui Wang, and Renjing Xu. Masked spiking transformer. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 1761–1771, 2023. 2

[24] Xingrun Xing, Boyan Gao, Zheng Zhang, David A Clifton, Shitao Xiao, Li Du, Guoqi Li, and Jiajun Zhang. Spikellm: Scaling up spiking neural network to large language models via saliency-based spiking. *arXiv preprint arXiv:2407.04752*, 2024. 2

[25] Han Xu, Xuerui Qiu, Yunhui Xu, Mohammed E Elbtity, Peng Zhou, Yang Tian, Rui-Jie Zhu, Jiahong Zhang, Shaowei Gu, Yuqi Pan, et al. Neuromorphic spike-based large language model. *National Science Review*, 13(4): nwaf551, 2026. 1, 2

[26] Zhanglu Yan, Kaiwen Tang, Zixuan Zhu, Zhenyu Bai, Qianhui Liu, and Weng-Fai Wong. Matterhorn: Efficient analog sparse spiking transformer architecture with masked time-to-first-spike encoding. *arXiv preprint arXiv:2601.22876*, 2026. 1

[27] Ning Yang, Fangxin Liu, Junjie Wang, Tao Yang, Kan Liu, Haibing Guan, and Li Jiang. Dash: Input-aware dynamic layer skipping for efficient llm inference with markov decision policies. *arXiv preprint arXiv:2505.17420*, 2025. 2

[28] Man Yao, Jiakui Hu, Zhaokun Zhou, Li Yuan, Yonghong Tian, Bo Xu, and Guoqi Li. Spike-driven transformer. *Advances in neural information processing systems*, 36:64043–64058, 2023. 1, 2

[29] 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. 1

[30] Jiqing Zhang, Bo Dong, Haiwei Zhang, Jianchuan Ding, Felix Heide, Baocai Yin, and Xin Yang. Spiking transformers for event-based single object tracking. In *Proceedings of the IEEE/CVF conference on Computer Vision and Pattern Recognition*, pages 8801–8810, 2022. 2

[31] Peiyuan Zhang, Guangtao Zeng, Tianduo Wang, and Wei Lu. Tinyllama: An open-source small language model. *arXiv preprint arXiv:2401.02385*, 2024. 6

[32] 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. 2

[33] 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. 2

[34] Zhaokun Zhou, Yijie Lu, Yanhao Jia, Kaiwei Che, Jun Niu, Liwei Huang, Xinyu Shi, Yuesheng Zhu, Guoqi Li, Zhaofei Yu, et al. Spiking transformer with experts mixture. *Advances in Neural Information Processing Systems*, 37:10036–10059, 2024. 2