SpikeLLM: Scaling up Spiking Neural Network to Large Language Models via Saliency-based Spiking

Anonymous Author(s)
Affiliation
Address
email

Abstract

The recent advancements in large language models (LLMs) with billions of parameters have significantly boosted their performance across various real-world applications. However, the inference processes for these models require substantial energy and computational resources, presenting considerable deployment challenges. In contrast, human brains, which contain approximately 86 billion biological neurons, exhibit significantly greater energy efficiency compared to LLMs with a similar number of parameters. Inspired by this, we redesign 7~70 billion parameter LLMs using bio-plausible spiking mechanisms, emulating the efficient behavior of the human brain. We propose the first spiking large language model termed SpikeLLM. Coupled with the proposed model, spike-driven quantization framework named Optimal Brain Spiking is introduced to reduce the energy cost and accelerate inference speed via two essential approaches: first (second)-order differentiation-based salient channel detection, and per-channel salient outlier expansion with Generalized Integrate-and-Fire neurons. The necessity of spike-driven LLM is proved by comparison with quantized LLMs. In the OmniQuant pipeline, SpikeLLM significantly reduces 25.51% WikiText2 perplexity and improves 3.08% average accuracy of common scene reasoning on a LLAMA2-7B 4A4W model. In the GPTQ pipeline, SpikeLLM realizes a sparse ternary LLM, achieving fully additive in linear layers. Compared with PB-LLM with similar operations, SpikeLLM also exceeds significantly. We will release our code on GitHub.

1 Introduction

Recent Artificial Neural Networks (ANNs) have shown scaling up Large Language Models (LLMs) [4, 48, 56, 22] can be one of the most potential techniques to access Artificial General Intelligence. However, despite these unprecedented and promising achievements, steering LLMs imposes a tremendous burden in terms of energy costs and computational requirements. For instance, running inference on the LLAMA-2 70B model requires three A100-80 GPUs, and each energy consumption is 400W. This creates a significant obstacle for generalizing LLMs to real-world applications, especially where limited battery capacity and memory size are critical, such as in mobile devices. To lower these barriers and broaden the applications of LLMs, we focus on energy-efficient artificial intelligence.

Compared with ANN-based LLMs, human brain nervous systems achieve superior intelligence with much less energy consumption [16, 21] and a comparable number of neurons, approximately 86 billion. For several decades, the brain-inspired computing (BIC) field [35, 58] focuses on mimicking the biological nature of the human brain to develop more efficient and general AI algorithms [34] and physical platforms [44, 42, 39]. Among these, spiking neural networks (SNNs) [34, 16] are particularly notable for their biological plausibility and binary event-driven efficiency [54, 44]. Despite their potential efficiency advantage, recent SNNs face two significant bottlenecks: (i) Firing

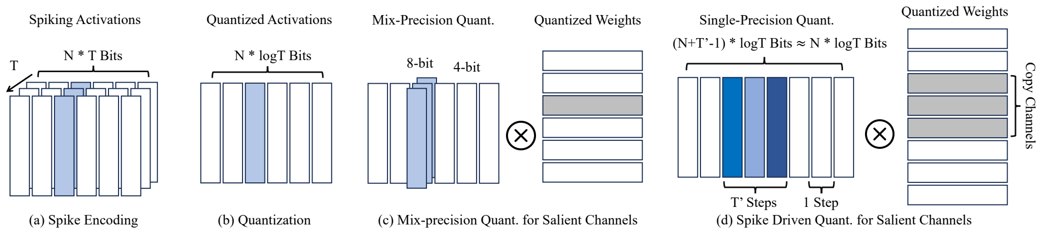

Figure 1: Different encoding methods. In (a, b), the activation has N channels; each value has T quantization levels. Given salient channels (in blue), mix-precision methods (c) are deployment unfriendly. In spike-driven methods (d), we expand salient channels by spiking dynamics to realize single precision quantization, where T' is spiking steps in salient channels.

Table 1: Comparason of encoding efficiency. L indicates quantization levels in each step, where $L \in [1, T]$. The accuracy and perplexity are evaluated in the common scene reasoning and WikiText with the LLAMA2-7B, 4A4W setting (Appendix B).

method	Steps	Bits/Step	Bits	Quant-Levels	Acc. ↑	PPL ↓
IF-SNN	T	1	T	T	–	–
GIF-SNN	T/L	log_2L	T/Llog_2L	T	–	–
Quant-ANN	1	log_2T	log_2T	T	47.58	15.25
SpikeLLM	≈ 1	$\approx log_2$T	$\approx log_2$T	T	50.65	11.36

Rate Encoding [6, 24, 9] in existing SNNs is inefficient in capturing adequate semantic information. As shown in Table 1 and Fig. 1, to express T quantization levels, a typical Integrate-and-Fire (IF) [28, 2] neuron requires T time steps to encode full information, while quantization methods only require log_2T bit digits in one step. (ii) Inefficient Optimization. Direct training [37, 49] SNNs need gradient estimation in backpropagation through time (BPTT) because of non-differentiable spiking dynamic; the ANN-SNN conversion [17, 18, 6, 24] often requires much more inference steps to simulate ANNs, both of which are impractical for scaling up to LLMs. These challenges have kept SNNs relatively small (under 1 B parameters) and hinder their ability to scale to the complexity of the human brain.

This work aims to scale up SNNs to large language models or even part of the human brain nervous system with 86B neurons. Towards this goal, we propose both micro designs to improve spike encoding efficiency and a macro design to drive spiking neurons. Currently, the primary approach to encode full-precision features to binary digits is model quantization. Given T quantization levels, quantization functions can efficiently encode into $log_2 T$ binary digits in one step but encounter significant quantization errors for outliers or salient values in low-bit conditions [50, 25]. Inspired by both quantized ANNs and SNNs, we introduce a hybrid encoding method of quantization and spike firing rate encoding, termed Generalized Integrate-and-Fire (GIF) neurons. In each spiking step, we apply the binary encoding as quantization with multiple bits; across different steps, we keep the auto-regressive spiking dynamics. As shown in Fig. 1 (d), a saliency-based spiking mechanism is proposed to divide and conquer salient channels and almost non-salient ones in LLMs, which allocates multistep spiking to encode salient channels and one-step spiking for others. As shown in Table 1, compared with IF neurons, GIF neurons efficiently compress binary code length from T bits to T/Llog_2L and saliency-based spiking further compresses to approximate log_2T bit overall channels. Compared with traditional quantization, GIF neurons maintain auto-regressive encoding advantages to accurately quantize salient channels with multisteps. At the same time, it can be implemented by expanding salient channels to multiple to avoid mix-precision matrix multiplication in Fig. 1 (c).

To drive spiking neurons, we propose a macro framework named Optimal Brain Spiking to detect salient channels, which is a weight-activation generalization of the classic Optimal Brain Surgeon (OBS) [19] for model pruning. The key concept of our method is distinguished approximations of weight and activation Taylor expansion to calculate their saliency. In detail, the first-order gradient and second-order Hessian metrics are leveraged for activations and weights respectively. For every matrix multiplication in LLMs, GIF neurons can be viewed as generalized quantizers to quantize salient and other channels based on the saliency rank generated by the Optimal Brain Spiking framework.

To evaluate the necessity of the Spiking LLMs, we integrate the proposed spiking mechanism with the most classic LLM quantization pipelines including the Omniquant [46] and GPTQ, named SpikeLLM.

For weight-activation quantization [46], we observe significant performance improvements in generation and common scene reasoning. To further achieve fully additive linear layers like previous SNNs, we quantize weights to ternary with the GPTQ [15] pipeline. Compared with PB-LLM [45] in similar cost, SpikeLLM exceeds dramatically. Our contributions are summarised as follows:

- We first scale up spiking neuronal dynamics to 7~70 billion parameters, promoting SNNs to the era of LLMs. The necessity of introducing a spiking mechanism to LLMs is proved by the comparison with classic quantization methods.
- We propose a Generalized Integrate-and-Fire (GIF) neuron as a general alternative to traditional quantizers. The firing rate encoding efficiency issue in traditional SNNs is conquered by encoding T steps into approximate log_2T steps.
- We propose an Optimal Brain Spiking framework to drive the GIF neurons. Per-channel saliency is efficiently detected with the first (second)-order differentiation-based metrics.

2 Related Works

Brain-Inspired Computing. The Brain-Inspired Computing (BIC) [35, 58] field focuses on building up bio-plausible and general fundamental theories, algorithms, software [12], and hardware platforms [44, 42, 39] inspired by the structures and functions of biological nervous systems like the human brain. Spiking neural network (SNN) [42, 34] is one of the most popular BIC algorithms which embeds biological spiking neural dynamics in each single neuron [54, 44]. Promoted by the development of both deep learning and advanced biological neuron science, recent SNNs focus on the deep residual learning [13], self-attention [53, 60], normlization [59], as well as biological learning rules [38], structures [40] and energy efficiency [44]. In optimization, recent SNNs apply ANN-SNN conversion [17, 18, 6, 24] or directly training [37, 49] techniques. Most SNNs focus on the computation vision field; language-oriented SNNs are almost less than 1 billion parameters, for example, SpikeLM [51], SpikingBERT [1], SpikeBERT [32], and SpikeGPT [61].

Model Quantization. Model quantization aims at reducing the bit-width of weights or activations to accelerate network inference. Recent quantization includes Post-Training Quantization (PTQ) [50, 15], Quantization Aware Training (QAT) [30], and calibration training methods [46]. For small models, QAT methods achieve higher performance because of training from scratch, for example, LSQ [11], U2NQ [29]. For LLM quantization, PTQ methods including GPTQ [15], GPTQ-ada [20], SpQR [10], OWQ [23], AWQ [26], and PB-LLM [45] are weight-only quantization; SmoothQuant [50] and RPTQ [55] achieve weight-activation quantization. Besides, LLM-QAT [30], QA-LORA [52], and calibration based methods including Omniquant [46], QLLM [27], and AffineQuant [33] achieve higher performance.

3 Problem Formulation

In this section, we first introduce the bio-inspired SNNs and then discuss the possibility of alternating traditional quantized LLMs with spiking LLMs. Further, we discuss the outlier and salient value quantization issue, which becomes the evidence to introduce the spiking mechanism to LLM quantization.

3.1 Spiking Neuronal Dynamics

SNNs can be considered ANNs by adding biological spiking neuronal dynamics in each neuron. Without loss of generality, the biological soma dynamics are approximately modeled by the first- or higher-order differential equations. The IF neuron is a first-order approximation of soma dynamics, combining the advantages of bio-plausibility and efficiency, which can be represented by:

$$\mathbf{v}(t) = \mathbf{v}(t-1) + \mathbf{x}^{(\ell-1)}(t) - \mathbf{s}^{(\ell)}(t-1)V_{th}, \tag{1}$$

$$\mathbf{s}^{(\ell)}(t) = \begin{cases} 0, & \text{if } \mathbf{v}(t) < V_{th} \\ 1, & \text{if } \mathbf{v}(t) \geq V_{th} \end{cases}, \tag{2}$$

$$\mathbf{x}^{(\ell-1)}(t) - \min\left(\mathbf{x}^{(\ell-1)}\right)^{\top} = \mathbf{s}^{(\ell)\top}(t)V_{th}, \tag{3}$$

where the IF neuron encodes a binary spike in each step t for a duration of T. In Eq.1, the membrane potential $\mathbf{v}(t)$ accumulates current input $\mathbf{x}^{(\ell-1)}(t)$ to the last time step $\mathbf{v}(t-1)$ to simulate the charging process in soma. A subjection reset is applied to subtract the spiking values from the membrane potential $\mathbf{v}(t)$. In Eq.2, if $\mathbf{v}(t)$ exceeds a certain firing threshold V_{th}, the neuron is fired and encodes the spike $\mathbf{s}^{(\ell)}(t)$ as 1; otherwise, encodes as 0. Thus, previous SNNs take the IF neuron as an activation quantizer, which encodes full-precision activation as 1-bit output per spiking step in Eq. 3. Different from SNNs with ReLU activations, to encode the negative values in transformers, we encode $\mathbf{x}^{(\ell-1)}(t) - \min\left(\mathbf{x}^{(\ell-1)}\right)^{\top}$ by spike firing rate, where $\min\left(\mathbf{x}^{(\ell-1)}\right)^{\top}$ can be viewed as the zero-point of the quantizer.

As shown in Table 1, compared with the asymmetric uniform quantization (Appendix A) that encodes T levels via log_2T bits, IF neurons auto-regressively encode per spike via total T bits, because firing rate encoding directly makes an average of spiking steps, which missing the numerical carry. In ANN-SNN conversion [5, 24], IF neuron equals uniform quantization, where SNNs expand T steps of their log_2T bit quantized ANN counterpart. By summation over spiking steps, the IF neuron encodes the input into log_2T bit integer values, where $\bar{\mathbf{s}}$ is the spike firing rate:

$$\mathbf{x}^{\mathrm{INT}} = \mathtt{Clip}\left(\mathtt{Round}\left\lceil T\bar{\mathbf{s}}^{(\ell)}\right\rfloor, 0, T\right). \tag{4}$$

3.2 Limitations of Traditional Quantization

Traditional quantization is an ill-posed problem between bit-width and quantization error, especially for post-training LLMs. In low-bit cases, the performance drop is often caused by quantization errors of outliers and salient values. Previous work has shown that outliers in activations have magnitudes over 100 $\times$ larger than most values, and salient values in weight matrices significantly impact the results. To more accurately quantize these values, previous methods such as AWQ [26], SpQR [10], SmoothQuant [50], and Omniquant [46] have proposed corresponding mitigation strategies. However, these methods are constrained by the limitations of traditional quantization frameworks:
(i) weight-activation quantization makes it hard to avoid quantization errors in outliers. AWQ [26] uses per-channel quantization step sizes to smooth outlier channels; however, it can only be applied to weight-only quantization, which is often insufficient for LLM compression.
(ii) Per-channel quantization is unfriendly to deployment. Since matrix multiplication is calculated per-token, per-channel quantization cannot be directly used to accelerate. As mitigation, SmoothQuant [50] and OmniQuant [46] rebalance the quantization difficulty of activations and weights; however, they do not directly eliminate the impact of outliers.
(iii) Mix-precision quantization is hardware unfriendly. SpQR [10] and PB-LLM [45] use mixed-precision quantization to avoid quantizing salient weights; however, mix-precision introduces difficulties in hardware deployment.
Based on these observations, there is an urgent requirement to explore weight-activation quantized LLMs and avoid the drawbacks of traditional quantization caused by outlier and salient values.

4 Spike-Driven Quantization

To avoid inefficient spike encoding in previous SNNs and significant quantization error in Quantized ANNs, we propose the first spiking large language model with two techniques: Generalized Integrate-and-Fire neurons for spike code length comparison and Optimal Brain Spiking framework for accurate saliency-based spike encoding. Compared with IF-SNNs, SpikeLLM compresses binary code length from T to approximate log_2 and can infer with both the low-bit matrix multiplication workload on GPUs and the per-bit computation workload on BIC chips at the same time.

4.1 Generalized Integrate-and-Fire Neuron

In Table 1, to fully express FP16 values by the firing rate encoding (IF-SNN), 2^{16} spiking steps are required, which is much more inefficient compared with 16-bit quantization. Because of the inefficiency of IF neurons, recent SNNs are almost less than 1 billion parameters and are hard to train with long-time backpropagation through time (BPTT). On the other hand, traditional quantization encodes all binary digits in one step, which makes it hard to quantize outliers. Based on both aspects, we make a balance between auto-regressive steps and code length in each step. This is achieved by

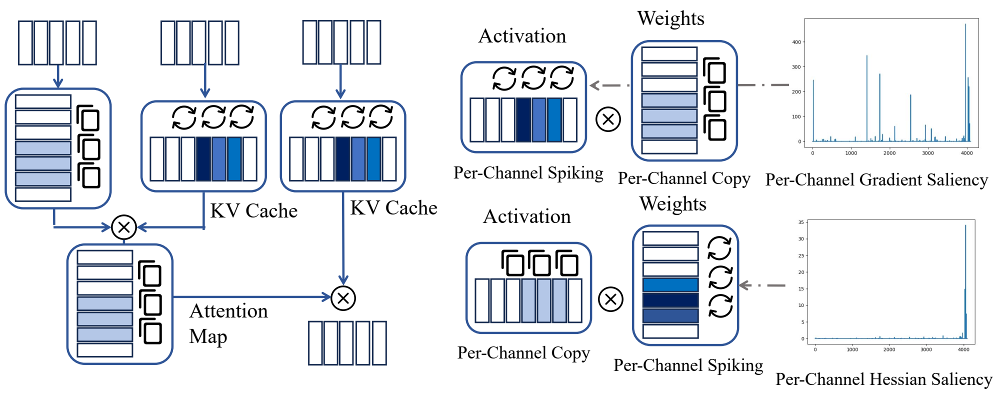

Figure 2: saliency-Aware spiking mechanisms in SpikeLLM. (Left) Spiking self-attention. Salient channels in the KV caches are encoded by multi-step spikes. (Right) Spiking activations or weights in a linear layer, where saliency is detected by gradient or Hessian metric respectively.

merging L spiking steps and encoding each step as low-bit digits with $log_2 L$ bit length. We define this L-step merged IF neuron as the Generalized Integrate-and-Fire (GIF) neuron:

$$\mathbf{s}_{GIF}(t) = \frac{1}{L}\sum_{t=1}^{L}\mathbf{s}_{IF}(t),\ \mathbf{s}_{GIF}(t) = \begin{cases} \frac{k}{L}, & \text{if } kV_{th} \le L\mathbf{v}(t) < (k+1)V_{th}, k = 0, 1, ..., L-1 \\ 1, & \text{if } \mathbf{v}(t) \ge V_{th} \end{cases}, \tag{5}$$

where there are L quantization levels in each spiking step. After merging, the auto-regressive steps T' become T/L, and each step has been encoded by $log_2 L$ bit quantization.

Remark 1. Ternary Spike. *The ternary spike $\mathbf{s}_{Ter}(t)$ proposed by SpikeLM [51] is a special case of GIF, which is formulated by merging a positive IF neuron $\mathbf{s}^+_{IF}(t)$ and a negative $\mathbf{s}^-_{IF}(t)$. Ternary spike not only increases quantization levels but also keeps additive in SNNs.*

$$\mathbf{s}_{Ter}(t) = \mathbf{s}^+_{IF}(t) + \mathbf{s}^-_{IF}(t), \quad \mathbf{s}_{Ter}(t) = \begin{cases} -1, & \text{if } \mathbf{v}(t) < -V_{th} \\ 0, & \text{if } \mathbf{v}(t) \in (-V_{th}, +V_{th}) \\ +1, & \text{if } \mathbf{v}(t) > +V_{th} \end{cases}. \tag{6}$$

4.2 Saliency-Aware Spiking Steps

Based on the fact that not all channels are equally important in LLMs, a saliency-aware spiking mechanism is proposed to address salient value quantization in Section 3.2. This is addressed by expanding salient channels in activations or weights with more spiking steps compared with unimportant channels. Given the GIF neuron to quantize activations, we first detect salient channels C' and the other channels C, and then allocate T'-step spikes to encode channels in C' and one-step for others in C, which can be represented by:

$$\begin{aligned} \mathbf{x}^{(\ell)} &= \frac{V_{th}}{T'}\sum_{t=1}^{T'}\mathbf{s}^{(\ell)}(t) + \min(\mathbf{x}^{(\ell-1)}) \\ &\simeq \frac{V_{th}}{T'}\sum_{t=1}^{T'}\mathbf{s}^{(\ell)}(t)|_{\mathbf{s}\in C'} + \mathbf{s}^{(\ell)}(1)V_{th}|_{\mathbf{s}\in C} + \min(\mathbf{x}^{(\ell-1)}), \end{aligned} \tag{7}$$

where $V_{th} = \frac{\max(\mathbf{x})}{T}$ is per-channel spiking thresholds, which confirms not clipping max values (as Eq.4). As shown in Fig. 2, this per-channel spiking mechanism can apply to KV-caches, activations, and weights. For weight-activation quantization, salient channels in KV-caches and activations have T spiking steps, while the other side of the matrix multiplication keeps one-step quantization, and copies corresponding channels for the same T steps. For weight-only quantization, the salient channels in weights have T spiking steps, while the corresponding channels in activations are copied. Following, it is essential to detect the salient channels in both weights and activations.

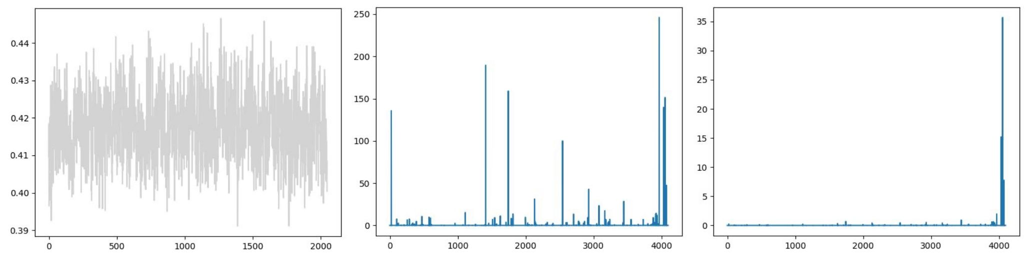
(a) Per-Token Gradient Saliency (b) Per-Channel Gradient Saliency (c) Per-Channel Hessian Saliency

Figure 3: Comparisons of different saliency metrics in the first linear layer. (a) Insignificant per-token gradient saliency in activations. (b) Significant per-channel gradient saliency in activations. (c) Significant per-channel Hessian saliency in weights.

4.3 Optimal Brain Spiking

Our Optimal Brain Spiking is a weight-activation generalization of the classic Optimal Brain Surgeon (OBS) framework [19]. Different from OBS which focuses on weight pruning with only the second-order differentiation, we focus on detecting salient channels in both activations and weights via both first and second-order differentiation. Given a post-training model well-optimized under a loss function $\mathcal{L}$, any weights or activations $\mathbf{x}$ in the model can be expressed by a second-order Taylor expansion around its optimal value $\mathbf{x}^*$:

$$\mathcal{L}(\mathbf{x}) \simeq \mathcal{L}(\mathbf{x}^*) + (\mathbf{x}-\mathbf{x}^*)^\top \nabla \mathcal{L}(\mathbf{x}^*) + \frac{1}{2}(\mathbf{x}-\mathbf{x}^*)^\top \mathbf{H}_{\mathcal{L}}(\mathbf{x}^*)(\mathbf{x}-\mathbf{x}^*), \tag{8}$$

where $\nabla \mathcal{L}(\mathbf{x}^*)$ and $\mathbf{H}_{\mathcal{L}}(\mathbf{x}^*)$ is the first-order differentiation and the second-order Hessian matrixes under the final loss $\mathcal{L}$ and we define $\delta\mathcal{L}(\delta\mathbf{x}) = \mathcal{L}(\mathbf{x}) - \mathcal{L}(\mathbf{x}^*)$. Specifically, for a linear layer with weights $\mathbf{W}$ and activations $\mathbf{X}$, we donate the quantization function as $\mathcal{Q}(.)$ and we have:

Theorem 1. Optimal Brain Spiking. *Given the layerwise objective to minimize the squared error, argmin*$||\mathbf{W}\mathbf{X} - \mathcal{Q}(\mathbf{W})\mathcal{Q}(\mathbf{X})||_2^2$, *the activation saliency is* $\mathbf{X} \circ \mathbf{W}^\top \mathbf{W}\mathbf{X}$, *and the weight saliency is* $\frac{\mathbf{W}_{ij}^2}{[\mathbf{H}_{ii}^{-1}]^2}$, *where the* $\circ$ *is Hadamard product.*

Proof. For activations, the gradient is not zero in a well-optimized model in Eq. 8, and we use the first-order Taylor expansion to approximate the effect of activation perturbations $\delta\mathbf{x}$: $\delta\mathcal{L}(\delta\mathbf{x}) \simeq \delta\mathbf{x}^\top \nabla\mathcal{L}(\mathbf{x}^*)$. Thus, the activation salient matrix is directly calculated according to $\delta\mathcal{L}(\delta\mathbf{x})$:

$$\text{Saliency}(\mathbf{X}) = \mathbf{X} \circ \mathbf{W}^\top \frac{\partial \mathcal{L}}{\partial \mathbf{W}\mathbf{X}} = \mathbf{X} \circ \mathbf{W}^\top \mathbf{W}\mathbf{X} \tag{9}$$

For weights, the gradient is zero because the optimizer directly optimizes weights to the local minimum after pretraining. Thus, the first-order term is zero in Eq.8 and $\delta\mathcal{L}(\delta\mathbf{w})$ has to approximate via the second-order term in Taylor expansion: $\delta\mathcal{L}(\delta\mathbf{w}) \simeq \frac{1}{2}\delta\mathbf{w}^\top \mathbf{H}_{\mathcal{L}}(\mathbf{w}^*)\delta\mathbf{w}$, which is proved in OBS [19]. And we apply the same weight saliency metric as OBS-based methods [45, 15, 14]:

$$\text{Saliency}(\mathbf{W}_{ij}) = \frac{\mathbf{W}_{ij}^2}{[\mathbf{H}_{ii}^{-1}]^2}, \quad \mathbf{H} = \mathbf{X}\mathbf{X}^\top. \tag{10}$$

□

Remark 2. Per-Channel Spiking Mechanism. *Given saliency matrixes Saliency*$(\mathbf{X})$ *and Saliency*$(\mathbf{W})$ *from Optimal Brain Spiking, per-channel means of first- (or second-) order differentation-based saliency are significant enough to divide salient channels in Eq.7.*

In implentation, the saliency matrix Saliency$(\mathbf{X})$ and Saliency$(\mathbf{W})$ have the same shape with activations $\mathbf{X}$ and weights $\mathbf{W}$, which are inefficient to store. As shown in Fig.3, we calculate per-channel or per-token means of the saliency matrix. We observe that both the per-channel saliency in activations and weights are robust enough to detect salient channels while per-token is insignificant. Based on *Remark 2*, we first compute per-channel saliency with calibration data and generate their rank to select the salient channels in Eq.7. Then, we store these lightweight masks for inference.

Table 2: Spiking settings to simulate quantized ANNs. We set 10% or 5% salient channels for 2 or 4 spiking steps respectively. Attention, Act. and Weight indicate where to apply GIF neurons.

Quantized-ANN	Step T	Spike-Level L	Salient Channels	Attention	Act.	Weight
4W4A	2 / 4	16	10% / 5%	✓	✓	–
2W8A	2 / 4	4	10% / 5%	–	–	✓
2W16A	2 / 4	4	10% / 5%	–	–	✓

Table 3: Weight-activation quantization results of LLaMA Models. OmniQuant† indicates we retrain the OmniQuant of the LLAMA-1 model with the unified scheme (see Appendix B).

Method	Saliency	#Bits	ACEs	PIQA	ARC-e	Arc-c	BoolQ	HellaSwag	Winogrande	**Avg.**
LLAMA-1-7B	–	FP16	1×	77.47	52.48	41.46	73.08	73.00	67.07	64.09
SmoothQuant	–	W4A4	0.0625×	49.80	30.40	25.80	49.10	27.40	48.00	38.41
LLM-QAT	–	W4A4	0.0625×	51.50	27.90	23.90	61.30	31.10	51.90	41.27
LLM-QAT+SQ	–	W4A4	0.0625×	55.90	35.50	26.40	62.40	47.80	50.60	46.43
OS+	–	W4A4	0.0625×	62.73	39.98	30.29	60.21	44.39	52.96	48.43
OmniQuant†	–	W4A4	0.0625×	63.28	46.38	27.56	62.23	40.24	52.49	48.70
SpikeLLM$_{T=2}$	0.10	W4A4	0.0687×	65.83	47.98	27.82	64.22	43.49	53.28	**50.44**
LLAMA-2-7B	–	FP16	1×	78.45	69.32	40.02	71.07	56.69	67.25	63.80
OmniQuant	–	W4A4	0.0625×	62.19	45.62	25.43	60.89	39.15	52.17	47.58
SpikeLLM$_{T=2}$	0.10	W4A4	0.0687×	64.47	48.74	27.30	63.27	43.29	56.83	**50.65**
OmniQuant	–	W2A8	0.0625×	51.90	27.82	19.97	38.26	25.85	50.36	35.69
SpikeLLM$_{T=4}$	0.05	W2A8	0.075×	54.62	30.89	20.90	53.91	30.75	53.12	**40.70**
OmniQuant	–	W2A16	0.125×	57.13	35.02	21.16	53.46	29.32	50.36	41.08
SpikeLLM$_{T=2}$	0.10	W2A16	0.138×	65.61	48.15	27.39	60.46	39.01	52.80	**48.90**
LLAMA-2-13B	–	FP16	1×	78.78	73.36	45.56	68.99	59.73	69.61	66.01
OmniQuant	–	W4A4	0.0625×	67.03	53.96	30.55	62.91	44.83	53.91	52.20
SpikeLLM$_{T=2}$	0.10	W4A4	0.0687×	66.49	55.30	30.12	64.16	47.43	51.54	**52.49**
OmniQuant	–	W2A8	0.0625×	57.51	35.27	19.11	61.31	29.95	49.41	42.09
SpikeLLM$_{T=4}$	0.05	W2A8	0.075×	64.09	48.74	24.23	62.29	40.38	52.49	**48.70**
OmniQuant	–	W2A16	0.125×	63.55	45.16	23.55	62.45	39.84	53.12	47.95
SpikeLLM$_{T=2}$	0.10	W2A16	0.138×	67.63	53.70	28.33	63.64	45.10	57.22	**52.60**
LLAMA-2-70B	–	FP16	1×	81.07	77.74	51.11	76.70	63.99	77.03	71.27
OmniQuant	–	W2A16	0.125×	62.57	43.86	22.78	56.42	39.60	52.49	46.29
SpikeLLM$_{T=2}$	0.10	W2A16	0.138×	76.44	66.92	38.31	66.88	51.86	59.19	**59.93**

5 Experiments

We evaluate the necessity to introduce SpikeLLM by comparison with quantized ANNs (because there are no spiking LLMs previously). Experiments are set from two aspects: (i) general weight-activation quantization in very low-bits; (ii) ternary quantized LLM towards fully additive linear layers.

Training Details. As shown in Table 2, SpikeLLM can simulate 4W4A (4-bit activation, 4-bit weight), 2W8A, or 2W16A quantization with different hyper-parameters, spiking step T and spike-level L, in GIF neurons. We follow main streams of post-training quantization (PTQ) [15] and calibration [46] pipelines. (i) For the low-bit quantization setting, our primary baseline is OmniQuant [46] and we report the detailed pipeline in Appendix B. We select LLAMA-1 (7B) [47] and LLAMA-2 (7B, 13B, 70B) [48] as full-precision models. (ii) For the additive SpikeLLM setting, we follow the GPTQ [15] framework and report training details in Appendix C.

Evaluation Tasks. We follow the same evaluation methods as the primary baselines, OmniQuant [46] and PB-LLM [45]. We evaluate perplexity (PPL) of language generation in WikiText2 [36] and C4 [41] benchmarks. We also evaluate zero-shot common scene reasoning tasks including PIQA [3], ARC-easy [8], ARC-challenge [8], BoolQ [7], HellaSwag [8], and Winogrande [43] datasets. For quantized ANNs and SNNs, ACE metric [57] is applied to evaluate operations (Appendix A).

5.1 Main Results

Comparison with quantized-ANNs. As shown in Table 3, we compare SpikeLLM with state-of-the-art weight-activation quantization methods including SmoothQuant, LLM-QAT, and OmniQuant,

Table 4: Comparisons between SpikeLLM and OmniQuant in the same pipeline with the Wikitext2 and C4 PPL metrics. We do not evaluate the W4A4 and W2A8 settings for LLAMA2-70B because the grouped-query attention (GQA) makes training unstable in the OmniQuant pipeline.

			LLAMA-2-7B		**LLAMA-2-13B**		**LLAMA-2-70B**	
Method	Saliency	#Bits	Wikitext2	C4	Wikitext2	C4	Wikitext2	C4
OmniQuant	–	W4A4	15.25	19.35	12.40	15.87	–	–
SpikeLLM$_{T=2}$	0.10	W4A4	**11.36**	15.87	**9.71**	**12.10**	–	–
SpikeLLM$_{T=4}$	0.05	W4A4	11.41	**14.34**	9.75	12.17	–	–
OmniQuant	–	W2A8	287.64	445.21	53.87	72.33	–	–
SpikeLLM$_{T=2}$	0.10	W2A8	**22.13**	**30.45**	13.56	18.73	–	–
SpikeLLM$_{T=4}$	0.05	W2A8	28.78	44.80	**12.80**	**17.05**	–	–
OmniQuant	–	W2A16	38.05	98.74	17.14	27.12	10.04	19.31
SpikeLLM$_{T=2}$	0.10	W2A16	**14.16**	**19.73**	9.45	13.86	6.35	9.62
SpikeLLM$_{T=4}$	0.05	W2A16	14.50	19.82	**9.17**	**12.37**	–	–

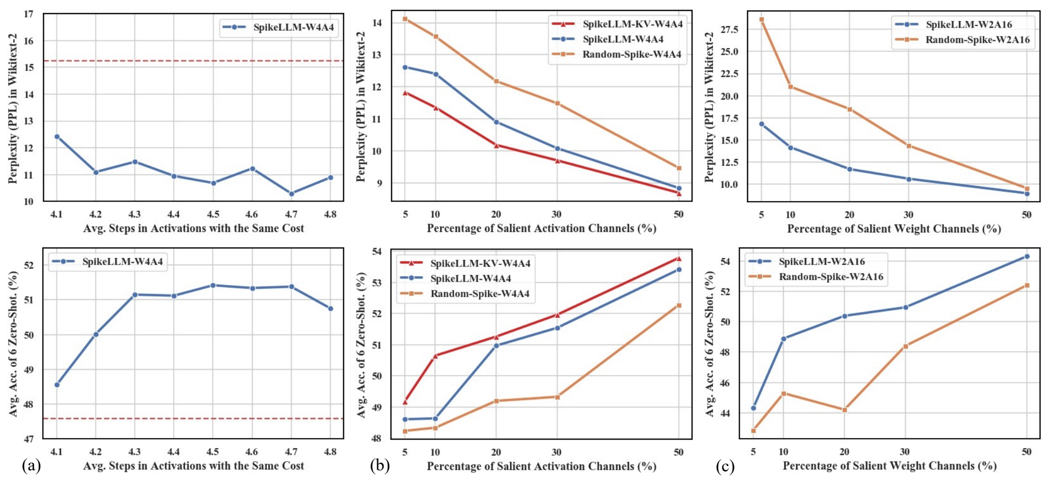

Figure 4: Ablation studies of GIF neurons and Optimal Brain Spiking in LLAMA-2-7B. (a) Comparison between SpikeLLM and Quantized-ANN with the same operations. (b) Ablations on spiking salient channels in activations and KV-Caches. (c) Ablations on spiking salient channels in weights.

which shows SpikeLLM improves significantly based on OmniQuant with a few additional spikes to enhance salient channels. For LLAMA-2-7/13B in Table 3, this performance enhancement is dramatic: for example, for the 2W8A and 2W16A quantization of 7B, improvements are 5.01% and 7.82% respectively with 20% and 10% additional spikes. In Table 4, their WikiText2 PPL of SpikeLLM$_{T=2}$ also significantly decrease 92.31% and 62.79% compared with baselines.

Efficiency of GIF neurons. We evaluate the efficiency of GIF neurons by comparison with OmniQuant-4A4W with almost the same operations. This is achieved by applying GIF neurons in both weights and activations. To confirm the same operations, we increase salient channels in activations and accordingly decrease spiking steps T and spike-level L in non-salient channels in weights. As shown in Fig.4 (a), given different percentages of salient channels in activations, SpikeLLM always exceeds the Omniquant-4W4A baseline (in red).

Ablations on Optimal Brain Spiking. To evaluate the efficiency of the Optimal Brain Spiking (OBSpiking) framework, we compare SpikeLLM with equal-operation baselines with randomly selected spiking channels, termed Random-Spike. In Fig. 4 (b), we compare OBSpiking in activations of linear layers alone, OBSpiking in both activations of linear layers and KV caches of self-attentions, and Random-Spike. With different percentages of salient channels, OBSpiking in both KV chches and activations helps to improve performance. In Fig. 4 (c), we compare Random-Spike and OBSpiking in weight quantization, which proves the effectiveness for weights.

Table 5: Comparisons between SpikeLLM and PB-LLM in LLAMA-2-7B towards additive linear layers. SpikeLLM$_{\texttt{Ter,x:y:z}}$ indicates using the ternary GIF neurons in Eq.6 as weight quantizers, where `x:y:z` is percentages of 1, 2, 4 spiking steps in weights (details in Appendix C).

Method	ACs	Equal Steps	PIQA	ARC-e	Arc-c	BoolQ	HellaSwag	Winogrande	**Avg.**
PB-LLM$_{80\%}$	80%	2.4	60.77	43.9	22.18	64.16	33.75	56.83	46.93
PB-LLM$_{90\%}$	90%	1.7	54.03	27.9	19.37	57.09	27.12	48.38	38.98
PB-LLM$_{95\%}$	95%	1.35	53.43	26.6	19.28	51.87	26.51	49.01	37.78
SpikeLLM$_{\texttt{Ter,70:25:5}}$	100%	1.4	65.83	51.89	25.17	68.47	40.48	60.77	52.10
SpikeLLM$_{\texttt{Ter,80:15:5}}$	100%	1.3	60.88	42.26	24.23	68.65	34.02	54.38	47.40
SpikeLLM$_{\texttt{Ter,85:10:5}}$	100%	1.25	55.44	31.99	20.99	61.83	30.02	52.01	42.05
SpikeLLM$_{\texttt{Ter,90:5:5}}$	100%	1.2	53.75	28.83	19.2	37.92	28.46	48.38	36.09

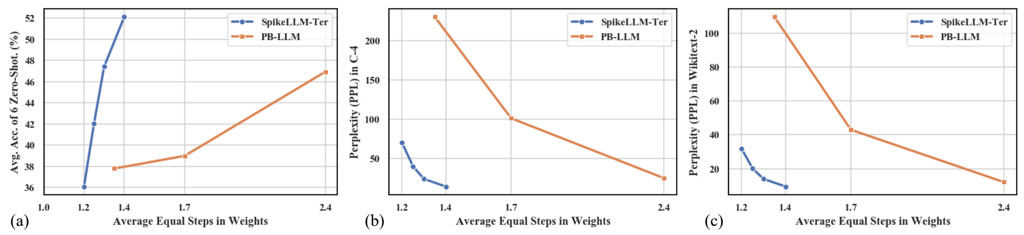

Figure 5: Effiency comparisons of SpikeLLM$_{\texttt{Ter}}$ and PB-LLM in Wikitext-2, C4 and 6 zero-shot benchmarks. We use the average of equal steps as the operation metric of SNNs and BNNs.

5.2 Additive Spiking LLMs

To maintain the additive nature and event-driven sparsity as previous SNNs, we additionally build SpikeLLM$_{\texttt{Ter}}$ based on the ternary GIF neuron in Eq.6 as the spiking weight quantizer. After weight quantization as Eq.6, linear layers in the LLM can be implemented by ACcumulation (AC). The detailed model setting, operation evaluation, and training pipeline are reported in Appendix C.

Comparison with binary LLM. We select PB-LLM for comparison since binary weight neural networks (BNNs) can be implemented by ACs. As shown in Table 5, SpikeLLM achieves full AC operations in linear layers by saliency-based spiking neuronal dynamics, instead of the deployment-unfriendly mixed-precision quantization in PB-LLM.

Efficiency of additive SpikeLLM. In Table 5, we evaluate the overall bit number of weights and AC operations in a linear layer by equal-steps. For PB-LLM, we view the average bit-width as the equal steps; for SpikeLLM, the equal steps are calculated by the average spiking steps of salient and other values. As shown in Table 5, SpikeLLM is able to exceed PB-LLM with similar equal-steps. Further, in Fig 5, we evaluate the Pareto front about equal-steps and performance, which shows SpikeLLM exceeds PB-LLM (BNN) in both effectiveness and efficiency.

6 Conclusion

This work proposes the first spiking large language models with 7~70 billion parameters, promoting SNNs to the era of LLMs. This is achieved by significant improvement of spike encoding efficiency, where the GIF neurons compress the code length from T to T/Llog_2L bits; the saliency-based spiking further compresses to log_2T bits. Different from ANN-SNN conversions that rely on quantization, this work exceeds quantization with the hybrid encoding of both SNNs and quantized-ANNs, which we think could become an interesting topic in the future.

Limitations

A primary limitation of this paper is the lack of deployment on neuromorphic chips. Recent studies [31] have shown that SNNs obtained through quantization training can maintain event-driven sparsity, making them directly usable for inference on neuromorphic chips. This theoretically ensures the feasibility of the potential deployment.

References

[1] Malyaban Bal and Abhronil Sengupta. Spikingbert: Distilling bert to train spiking language models using implicit differentiation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 10998–11006, 2024.

[2] Michele Barbi, Santi Chillemi, Angelo Di Garbo, and Lara Reale. Stochastic resonance in a sinusoidally forced lif model with noisy threshold. *Biosystems*, 71(1-2):23–28, 2003.

[3] Yonatan Bisk, Rowan Zellers, Jianfeng Gao, Yejin Choi, et al. Piqa: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 7432–7439, 2020.

[4] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[5] 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. In *International Conference on Learning Representations*, 2021.

[6] 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.

[7] Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. Boolq: Exploring the surprising difficulty of natural yes/no questions. *arXiv preprint arXiv:1905.10044*, 2019.

[8] Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*, 2018.

[9] Shikuang Deng and Shi Gu. Optimal conversion of conventional artificial neural networks to spiking neural networks. *arXiv preprint arXiv:2103.00476*, 2021.

[10] Tim Dettmers, Ruslan Svirschevski, Vage Egiazarian, Denis Kuznedelev, Elias Frantar, Saleh Ashkboos, Alexander Borzunov, Torsten Hoefler, and Dan Alistarh. Spqr: A sparse-quantized representation for near-lossless llm weight compression. *arXiv preprint arXiv:2306.03078*, 2023.

[11] Steven K Esser, Jeffrey L McKinstry, Deepika Bablani, Rathinakumar Appuswamy, and Dharmendra S Modha. Learned step size quantization. *arXiv preprint arXiv:1902.08153*, 2019.

[12] Wei Fang, Yanqi Chen, Jianhao Ding, Zhaofei Yu, Timothée Masquelier, Ding Chen, Liwei Huang, Huihui Zhou, Guoqi Li, and Yonghong Tian. Spikingjelly: An open-source machine learning infrastructure platform for spike-based intelligence. *Science Advances*, 9(40):eadi1480, 2023.

[13] 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*, 34:21056–21069, 2021.

[14] Elias Frantar and Dan Alistarh. Optimal brain compression: A framework for accurate post-training quantization and pruning. *Advances in Neural Information Processing Systems*, 35:4475–4488, 2022.

[15] Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. Gptq: Accurate post-training quantization for generative pre-trained transformers. *arXiv preprint arXiv:2210.17323*, 2022.

[16] Wulfram Gerstner, Werner M Kistler, Richard Naud, and Liam Paninski. *Neuronal dynamics: From single neurons to networks and models of cognition*. Cambridge University Press, 2014.

[17] Bing Han, Gopalakrishnan Srinivasan, and Kaushik Roy. Rmp-snn: Residual membrane potential neuron for enabling deeper high-accuracy and low-latency spiking neural network. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 13558–13567, 2020.

[18] Zecheng Hao, Jianhao Ding, Tong Bu, Tiejun Huang, and Zhaofei Yu. Bridging the gap between anns and snns by calibrating offset spikes. *arXiv preprint arXiv:2302.10685*, 2023.

[19] Babak Hassibi and David Stork. Second order derivatives for network pruning: Optimal brain surgeon. *Advances in neural information processing systems*, 5, 1992.

[20] Jung Hwan Heo, Jeonghoon Kim, Beomseok Kwon, Byeongwook Kim, Se Jung Kwon, and Dongsoo Lee. Rethinking channel dimensions to isolate outliers for low-bit weight quantization of large language models. *arXiv preprint arXiv:2309.15531*, 2023.

[21] Eugene M Izhikevich. Simple model of spiking neurons. *IEEE Transactions on neural networks*, 14(6):1569–1572, 2003.

[22] Teven Le Scao, Angela Fan, Christopher Akiki, Ellie Pavlick, Suzana Ilić, Daniel Hesslow, Roman Castagné, Alexandra Sasha Luccioni, François Yvon, Matthias Gallé, et al. Bloom: A 176b-parameter open-access multilingual language model. 2023.

[23] Changhun Lee, Jungyu Jin, Taesu Kim, Hyungjun Kim, and Eunhyeok Park. Owq: Lessons learned from activation outliers for weight quantization in large language models. *arXiv preprint arXiv:2306.02272*, 2023.

[24] 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*, pages 6316–6325. PMLR, 2021.

[25] Ji Lin, Jiaming Tang, Haotian Tang, Shang Yang, Xingyu Dang, and Song Han. Awq: Activation-aware weight quantization for llm compression and acceleration. *arXiv preprint arXiv:2306.00978*, 2023.

[26] Ji Lin, Jiaming Tang, Haotian Tang, Shang Yang, Xingyu Dang, and Song Han. Awq: Activation-aware weight quantization for llm compression and acceleration. *arXiv preprint arXiv:2306.00978*, 2023.

[27] Jing Liu, Ruihao Gong, Xiuying Wei, Zhiwei Dong, Jianfei Cai, and Bohan Zhuang. Qllm: Accurate and efficient low-bitwidth quantization for large language models. *arXiv preprint arXiv:2310.08041*, 2023.

[28] Ying-Hui Liu and Xiao-Jing Wang. Spike-frequency adaptation of a generalized leaky integrate-and-fire model neuron. *Journal of computational neuroscience*, 10:25–45, 2001.

[29] Zechun Liu, Kwang-Ting Cheng, Dong Huang, Eric P Xing, and Zhiqiang Shen. Nonuniform-to-uniform quantization: Towards accurate quantization via generalized straight-through estimation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4942–4952, 2022.

[30] Zechun Liu, Barlas Oguz, Changsheng Zhao, Ernie Chang, Pierre Stock, Yashar Mehdad, Yangyang Shi, Raghuraman Krishnamoorthi, and Vikas Chandra. Llm-qat: Data-free quantization aware training for large language models. *arXiv preprint arXiv:2305.17888*, 2023.

[31] Xinhao Luo, Man Yao, Yuhong Chou, Bo Xu, and Guoqi Li. Integer-valued training and spike-driven inference spiking neural network for high-performance and energy-efficient object detection. *arXiv preprint arXiv:2407.20708*, 2024.

[32] 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. *arXiv preprint arXiv:2308.15122*, 2023.

[33] Yuexiao Ma, Huixia Li, Xiawu Zheng, Feng Ling, Xuefeng Xiao, Rui Wang, Shilei Wen, Fei Chao, and Rongrong Ji. Affinequant: Affine transformation quantization for large language models. *arXiv preprint arXiv:2403.12544*, 2024.

[34] Wolfgang Maass. Networks of spiking neurons: the third generation of neural network models. *Neural networks*, 10(9):1659–1671, 1997.

[35] Adnan Mehonic and Anthony J Kenyon. Brain-inspired computing needs a master plan. *Nature*, 604(7905):255–260, 2022.

[36] Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. *arXiv preprint arXiv:1609.07843*, 2016.

[37] 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*, 36(6):51–63, 2019.

[38] Alexandre Payeur, Jordan Guerguiev, Friedemann Zenke, Blake A Richards, and Richard Naud. Burst-dependent synaptic plasticity can coordinate learning in hierarchical circuits. *Nature neuroscience*, 24(7):1010–1019, 2021.

[39] Jing Pei, Lei Deng, Sen Song, Mingguo Zhao, Youhui Zhang, Shuang Wu, Guanrui Wang, Zhe Zou, Zhenzhi Wu, Wei He, et al. Towards artificial general intelligence with hybrid tianjic chip architecture. *Nature*, 572(7767):106–111, 2019.

[40] Quang Pham, Chenghao Liu, and Steven Hoi. Dualnet: Continual learning, fast and slow. *Advances in Neural Information Processing Systems*, 34:16131–16144, 2021.

[41] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *The Journal of Machine Learning Research*, 21(1):5485–5551, 2020.

[42] Kaushik Roy, Akhilesh Jaiswal, and Priyadarshini Panda. Towards spike-based machine intelligence with neuromorphic computing. *Nature*, 575(7784):607–617, 2019.

[43] Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. Winogrande: An adversarial winograd schema challenge at scale. *Communications of the ACM*, 64(9):99–106, 2021.

[44] Catherine D Schuman, Shruti R Kulkarni, Maryam Parsa, J Parker Mitchell, Bill Kay, et al. Opportunities for neuromorphic computing algorithms and applications. *Nature Computational Science*, 2(1):10–19, 2022.

[45] Yuzhang Shang, Zhihang Yuan, Qiang Wu, and Zhen Dong. Pb-llm: Partially binarized large language models. *arXiv preprint arXiv:2310.00034*, 2023.

[46] Wenqi Shao, Mengzhao Chen, Zhaoyang Zhang, Peng Xu, Lirui Zhao, Zhiqian Li, Kaipeng Zhang, Peng Gao, Yu Qiao, and Ping Luo. Omniquant: Omnidirectionally calibrated quantization for large language models. *arXiv preprint arXiv:2308.13137*, 2023.

[47] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

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

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

[50] Guangxuan Xiao, Ji Lin, Mickael Seznec, Hao Wu, Julien Demouth, and Song Han. Smoothquant: Accurate and efficient post-training quantization for large language models. In *International Conference on Machine Learning*, pages 38087–38099. PMLR, 2023.

[51] Xingrun Xing, Zheng Zhang, Ziyi Ni, Shitao Xiao, Yiming Ju, Siqi Fan, Yequan Wang, Jiajun Zhang, and Guoqi Li. Spikelm: Towards general spike-driven language modeling via elastic bi-spiking mechanisms. *arXiv preprint arXiv:2406.03287*, 2024.

[52] Yuhui Xu, Lingxi Xie, Xiaotao Gu, Xin Chen, Heng Chang, Hengheng Zhang, Zhensu Chen, Xiaopeng Zhang, and Qi Tian. Qa-lora: Quantization-aware low-rank adaptation of large language models. *arXiv preprint arXiv:2309.14717*, 2023.

[53] Man Yao, JiaKui Hu, Zhaokun Zhou, Li Yuan, Yonghong Tian, XU Bo, and Guoqi Li. Spike-driven transformer. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

[54] Bojian Yin, Federico Corradi, and Sander M Bohté. Accurate and efficient time-domain classification with adaptive spiking recurrent neural networks. *Nature Machine Intelligence*, 3(10):905–913, 2021.

[55] Zhihang Yuan, Lin Niu, Jiawei Liu, Wenyu Liu, Xinggang Wang, Yuzhang Shang, Guangyu Sun, Qiang Wu, Jiaxiang Wu, and Bingzhe Wu. Rptq: Reorder-based post-training quantization for large language models. *arXiv preprint arXiv:2304.01089*, 2023.

[56] Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, et al. Opt: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068*, 2022.

[57] Yichi Zhang, Zhiru Zhang, and Lukasz Lew. Pokebnn: A binary pursuit of lightweight accuracy. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12475–12485, 2022.

[58] Youhui Zhang, Peng Qu, Yu Ji, Weihao Zhang, Guangrong Gao, Guanrui Wang, Sen Song, Guoqi Li, Wenguang Chen, Weimin Zheng, et al. A system hierarchy for brain-inspired computing. *Nature*, 586(7829):378–384, 2020.

[59] Hanle Zheng, Yujie Wu, Lei Deng, Yifan Hu, and Guoqi Li. Going deeper with directly-trained larger spiking neural networks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pages 11062–11070, 2021.

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

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

A Low-Bit Quantization

Low-bit quantization typically maps full-precision value to fewer quantization levels. We first focus on the most widely used asymmetric uniform quantization. Given the full-precision value $\mathbf{x}$, the quantizer first maps $\mathbf{x}$ to a `INT` value by linear translation and `Round` functions, and then maps the discrete `INT` value back to its original range, which the former translation can be expressed as:

$$\mathbf{x}^{\text{INT}} = \texttt{Round}\lceil \frac{\mathbf{x}^{\text{FP16}} - \min(\mathbf{x}^{\text{FP16}})}{\Delta} \rfloor, \quad \Delta = \frac{\max(\mathbf{x}) - \min(\mathbf{x})}{2^N - 1}. \tag{11}$$

In linear layers, quantized weights w^q or activations a^q can be represented as binary digits in M or N bits: $a^q = \sum_{i=0}^{M-1} \mathbf{a}_i 2^i$, $w^q = \sum_{j=0}^{N-1} \mathbf{w}_j 2^j$. Therefore, the Multiply-ACcumulate (MAC) can be implemented by bit level `AND` and `PopCount` operations:

$$a^q \cdot w^q = \sum_{i=0}^{M-1} \sum_{j=0}^{N-1} 2^{i+j} \texttt{PopCount}[\texttt{AND}(\mathbf{a}_i, \mathbf{w}_j)]. \tag{12}$$

This indicates the complexity and energy consumption of MAC operations are proportional to $M \times N$. Therefore, the number of operations in a quantized model can be measured using the arithmetic computation effort (ACE) metric [57], which is defined as $M \times N$ for a MAC operation between the M-bit weight and N-bit activation. Recent LLMs contain more than 10B ~100B parameters, making it an essential requirement to push not only weights but also activations to lower bit-width.

B Training Details of the OmniQuant Pipeline.

B.1 Training Details.

We use a unified training config as shown in all of our experiments. We train LLAMA-1 (7B), and LLAMA-2 (7B, 13B, 70B) for both OmniQuant baselines and SpikeLLMs according to this training scheme. Compared with the original OmniQuant, we do not apply loss augmentation methods except for 70B models for training stability and we set the quantization group number as 1 in all of the experiments. Therefore, this scheme can be viewed as a simplified version without bells and whistles to focus on the influence of the quantization method itself.

In SpikeLLM, we compute the saliency metric for activations layer by layer during the OmniQuant pipeline. Different from activations, we compute the saliency metric for weights directly using the features from the first embedding layer. Because we find this can make the computation of the inverse Hessian matrix more stable compared with computing layer by layer.

Table 6: Training settings on the OmniQuant scheme. LET and LWC indicate learnable equivalent transformation and learnable weight clipping.

config	4W4A	2W8A	2W16A
LET	True	True	False
LWC	True	True	True
learning rate of LET	0.001	0.001	N/A
learning rate of LWC	0.01	0.01	0.01
activation smooth	0.75	N/A	N/A
batch size	1	1	1
loss augmentation	False	False	True
epochs	20	20	40
group	1	1	1

B.2 Training Data.

Following OmniQuant [46], we randomly select 128 calibration data from WikiText2, which are 2048-token chunks. We further investigate the influence of different training samples to confirm the robustness of the proposed methods. In the OmniQuant pipeline, training samples are randomly cropped from the Wikitext2 dataset. To compare the performance of SpikeLLM and the OmniQuant baseline, we use a set of random seeds to sample different training data each time. In Fig. 6, we sample data and train 40 times, using the same seed for both OmniQuant and SpikeLLM each time. SpikeLLM achieves higher average accuracy on 6 zero-shot datasets and lower Wikitext2 or C4 perplexity at the same time, showing consistently better performance. For the same training data, SpikeLLM always performs better. In other experiments in this paper, we keep the random seed as 2.

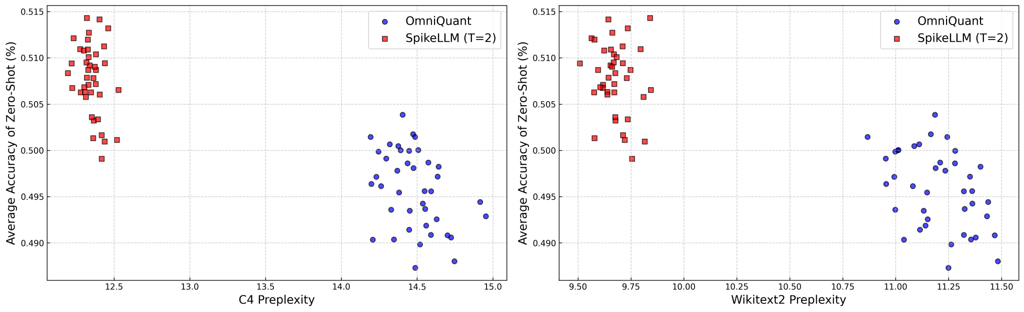

Figure 6: Comaprison of different training data for LLAMA-1-7b 4W4A models. In SpikeLLMs, 10% activation channels are set to 2 spiking steps.

C Training Details of the Ternary SpikeLLM.

C.1 Model Definition.

In additive SpikeLLM, we apply the ternary spike level {-1,0,+1} to encode weights according to Eq. 6. After weight ternarization, the matrix multiplication in the linear layer can be implemented by full ACcumulation (AC). As shown in Table 7, we have 3 spiking-step settings. For example, for the SpikeLLM$_{\texttt{Ter,70:25:5}}$ model, we allocate the most salient 5% values in weights with 3 spiking steps, the following 25% with 2 spiking steps, and the rest 70% with 1 spiking steps. The unstructured spiking steps are applied, which is different from the experiment in the Main Result Section. As the following section, this is because of more accurate quantization in ultra-low bit quantization and fair comparison with PB-LLM [45].

Table 7: Model settings in additive SpikeLLM. 1,2,3-Steps indicate the percentages of 1, 2, and 3 spiking-step encoding in weights respectively.

Model	1-Step	2-Step	3-Step	Avg. Step	Spike-Level
SpikeLLM$_{\texttt{Ter,70:25:5}}$	70%	25%	5%	1.4	{-1,0,+1}
SpikeLLM$_{\texttt{Ter,80:15:5}}$	80%	15%	5%	1.3	{-1,0,+1}
SpikeLLM$_{\texttt{Ter,85:10:5}}$	85%	10%	5%	1.25	{-1,0,+1}
SpikeLLM$_{\texttt{Ter,90:5:5}}$	90%	5%	5%	1.2	{-1,0,+1}

C.2 Structured vs. Unstructured Spiking in Weights.

For weight quantization, as shown in Table 8, unstructured spiking steps usually achieve higher performance compared with structured ones. Our per-channel spiking scheme can achieve higher performance by setting per-channel spiking steps as elementwise spiking steps. However, unstructured conditions need additional masks and are less friendly to deployment. Therefore, we keep structured settings in low-bit quantization. But for additive LLMs, the performance is more important in the extreme case, and we choose the unstructured settings. Moreover, the PB-LLM baselines are also unstructured, so that, it can also confirm the fair comparison.

C.3 Training Details.

The same as PB-LLM [45], we follow the GPT-Q [15] pipeline for the post-training weight quantization. In quantization, we keep the same block size of 128. We apply the same 128 calibration data as

Table 8: Comaprison between structured and unstructured weight quantization in the OmniQuant pipeline.

Method	Saliency	#Bits	ACEs	PIQA	ARC-e	Arc-c	BoolQ	HellaSwag	Winogrande	**Avg.**
LLAMA-2-7B	–	FP16	$1\times$	78.45	69.32	40.02	71.07	56.69	67.25	63.80
OmniQuant	–	W2A16	$0.125\times$	57.13	35.02	21.16	53.46	29.32	50.36	41.08
SpikeLLM$_{T=2}$-Structured	0.10	W2A16	$0.138\times$	65.61	48.15	27.39	60.46	39.01	52.80	48.90
SpikeLLM$_{T=2}$-Unstructured	0.10	W2A16	$0.138\times$	72.63	60.06	30.89	65.05	48.52	59.51	56.11

PB-LLMs. Each data is randomly selected from WikiText2 and tokenized to formulate a 2048-token chunk.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: the claims made in abstract and introduction are consistent and supported by the experiment results.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We have discussed and limitation of our algorithm in the experiment section while analysing.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory Assumptions and Proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [Yes]

Justification: We have the proofs along with the proposed theorem in section 4.2.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental Result Reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: We will release the code.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: We use public available datasets.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental Setting/Details**

Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

Answer: [Yes]

Justification: We explain all the details of the experiment setting at the beginning of section 5.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment Statistical Significance**

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [No]

Justification: In the Large Language Model routine, only the mean of the results are reported.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).

- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments Compute Resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We cover all the details of the experiment setting at the beginning of section 5.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code Of Ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: Our study is only about standard machine learning.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader Impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: Our study is potentially beneficial for energy saving and efficiency, which has also a potential application on medical mobile devices.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.

- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification:

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [NA]

    Justification:

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.

- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [NA]

    Justification:

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and Research with Human Subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification:

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification:

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.