HardF-SNN: Hardware-Friendly Quantization for Spiking Neural Networks with Efficient Integer-Arithmetic-Only Inference

Hanwen Liu[1], Kexin Shi[1], Jieyuan Zhang[1], Yimeng Shan[1], Jibin Wu[2], Wenyu Chen[1], Malu Zhang[1]*

[1]University of Electronic Science and Technology of China
[2]The Hong Kong Polytechnic University, Hong Kong SAR, China

Abstract

Spiking Neural Networks (SNNs) are emerging as a promising energy-efficient alternative to Artificial Neural Networks (ANNs) due to their event-driven computation paradigm. However, recent advances toward large-scale high-performance SNNs inevitably lead to substantial memory and computational overhead. While quantization offers a potential way, many quantization approaches fail to deliver verifiable efficiency gains on resource-constrained hardware platforms. In this paper, we propose a lightweight and hardware-friendly SNN, termed HardF-SNN. Specifically, we first build a baseline model using shared-scale quantization and BN folding to simulate integer-only inference, as this has not been thoroughly discussed in prior SNN works. Then, through empirical and theoretical analysis, we identify that the baseline suffers from accuracy degradation and may cause training failure. To mitigate these issues, we propose proportional shared-scale quantization for enhanced dynamic range and integer-only BN using bit-shifting to stabilize training. Extensive experiments show that HardF-SNN achieves an optimal balance between performance and efficiency with excellent hardware compatibility. To demonstrate its effectiveness on resource-limited platforms, HardF-SNN is deployed on a dedicated FPGA-based hardware accelerator. Evaluation results indicate that our implementation achieves significant performance improvements over several existing hardware accelerators.

Introduction

Spiking Neural Networks (SNNs) have emerged as a promising approach for realizing energy-efficient computational intelligence due to their biological plausibility (Maass 1997; Yan et al. 2025; Liu et al. 2024), spike-driven communication (Zhang et al. 2025b,c), and low power consumption (Sun et al. 2025; Liu et al. 2023). Unlike Artificial Neural Networks (ANNs), spiking neurons transmit information through sparse and binary spikes, thereby replacing computationally intensive multiply and accumulate (MAC) operations with low-cost accumulate (AC) operations. Benefiting from their energy-efficient nature, SNNs have been widely adopted in a variety of tasks, including image classification (Xue et al. 2024; Wang et al. 2024; Shi et al. 2024; Chen et al. 2023), object detection (Wang et al. 2025c,b; Zhang et al. 2025a), and audio recognition (Shi et al. 2025; Wang et al. 2025a). Despite their commendable performance, these studies come at the cost of massive model parameters and high computational complexity, which hinder their efficient deployment on resource-constrained edge devices. A promising way to tackle this challenge is via model compression.

Model quantization (Jacob et al. 2018; Krishnamoorthi 2018; Deng et al. 2021; Wei et al. 2025a; Cao et al. 2025), as one of the most effective model compression techniques, enables the conversion of model parameters from high-precision floating-point to low-bit-width integer representations. It not only reduces memory footprint by storing parameters in low precision but also enhances computational efficiency with low-precision arithmetic. Despite these advantages, existing SNN quantization methods continue to encounter two critical challenges: (1) achieving an efficient trade-off between efficiency and accuracy, and (2) delivering verifiable efficiency improvements on resource-limited hardware devices.

Firstly, most existing SNN quantization methods focus primarily on low-precision weight quantization and largely overlook the substantial memory overhead associated with membrane potentials (Pei et al. 2023; Wei et al. 2025b; Deng et al. 2020b). Although prior lightweight studies have attempted to quantize both weights and membrane potentials simultaneously (Yin et al. 2024; Putra and Shafique 2021; Hasssan et al. 2024), these methods often suffer from significant performance degradation or lack validation on large-scale datasets. Therefore, there remains room for further performance and efficiency optimizations.

Secondly, most quantization schemes for SNNs primarily adopt simulated quantization, where some or even all computations still depend on floating-point arithmetic. This floating-point dependency not only undermines the benefits of quantization but also imposes additional hardware implementation costs for SNN deployment. In addition, these SNN quantization methods are incompatible with neuromorphic hardware, such as Loihi (Davies et al. 2018), TrueNorth (Akopyan et al. 2015), and Tianjic (Deng et al. 2020a), which are designed for integer-only processing. Similarly, these models are not suitable for deployment on FPGA-based SNN accelerators (Li et al. 2023; Aung et al.

*Corresponding author: maluzhang@uestc.edu.cn

2023), as the inherent fabric of FPGAs only supports integer computation. Although previous studies (Jacob et al. 2018; Tan and Wu 2023; Guo, Wang, and Cui 2021; Yin et al. 2024) have proposed integer-only quantization pipelines that fully exploit efficient integer arithmetic units, these methods commonly suffer from performance degradation and exhibit limited applicability to SNN models. More seriously, they have yet to achieve ultra-low precision ($<$ 4-bit) integer quantization. Thus, achieving efficient and high-performance deployment of SNNs on resource-limited hardware platforms remains challenging.

In this paper, we propose HardF-SNN for the efficient deployment of high-performance SNNs in resource-constrained scenarios. We first build a baseline model that integrates shared-scale quantization and batch normalization (BN) folding to enable SNNs to perform inference without floating-point arithmetic. While the baseline facilitates efficient deployment on hardware accelerators, it suffers from performance degradation and even training failure. To address these, we analyze the underlying causes and propose effective solutions to enhance model performance while ensuring hardware-friendly implementation. To demonstrate its effectiveness on a resource-constrained platform, we deploy HardF-SNN on a dedicated FPGA-based hardware accelerator to validate its hardware performance. The main contributions of this paper are outlined as follows:

- We first develop a hardware-friendly and lightweight HardF-SNN baseline by integrating shared-scale quantization and BN folding to simulate integer-only inference during training. This baseline effectively minimizes storage and computational needs, making it well-suited for hardware deployment, but it suffers from performance limitations and training failure at ultra-low precision.
- We reveal that shared-scale quantization significantly compresses the dynamic range of membrane potentials, causing many distinct values to collapse into the same quantization levels and thereby constraining the effective dynamic range. To address this issue, we propose proportional shared-scale quantization, which better preserves the distribution of membrane potentials.
- We reveal that under ultra-low quantization, BN folding causes most BN-folded weights to collapse into the same quantization bin, leading to ineffective gradient updates and severely hindering model convergence. To enable integer-only inference, we propose integer-only BN to ensure stable training of the model while supporting efficient hardware deployment.
- We conduct comprehensive experiments on various benchmark datasets to validate the effectiveness of our proposed method. Extensive experimental results demonstrate that HardF-SNN achieves maximized efficiency while obtaining highly competitive performance, and offers high compatibility with hardware accelerators.
- We design an FPGA-based SNN hardware accelerator tailored for HardF-SNN. Hardware evaluation results indicate that our implementation surpasses other state-of-the-art accelerators, highlighting the potential of HardF-SNN for deployment on resource-limited devices.

Hardware-Friendly SNN Baseline

In this section, we build a HardF-SNN baseline by integrating shared-scale quantization and BN folding techniques, since it has not been thoroughly discussed in previous SNN work. In the baseline, the forward pass simulates the quantization process as implemented in actual inference engines. Backpropagation proceeds in the standard manner with all parameters maintained in floating-point format.

Neuron Model

SNNs are inspired by biological neural networks, achieving energy efficiency by utilizing binary spikes for computation. It is known that the Leaky Integrate-and-Fire (LIF) model is the most commonly used neuron model in SNN applications (Wu et al. 2018a). The membrane potential of the LIF model is expressed as:

$$\tilde{\mathbf{U}}^l[t] = \mathbf{U}^l[t-1] + \mathbf{W}^l\mathbf{S}^{l-1}[t], \tag{1}$$

where $\tilde{\mathbf{U}}^l[t]$ is the membrane potential of neurons in layer l at timestep t, $\mathbf{W}^l$ is the weight matrix of layer l, and $\mathbf{S}^{l-1}$ is the input spike from layer $l-1$. When $\tilde{\mathbf{U}}$ exceeds the threshold θ, the LIF neuron will fire a spike; otherwise, it remains silent. The spike generation function is defined as:

$$\mathbf{S}^l[t] = \begin{cases} 1, & \text{if } \tilde{\mathbf{U}}^l[t] \geq \theta, \\ 0, & \text{otherwise.} \end{cases} \tag{2}$$

After spike emission, we use the hard reset mechanism to update the membrane potential. The reset mechanism can be described as:

$$\mathbf{U}^l[t] = \tau\tilde{\mathbf{U}}^l[t] \cdot (1 - \mathbf{S}^l[t]), \tag{3}$$

where τ is the time constant. This mechanism resets the membrane potential to zero whenever a spike occurs.

Shared-Scale Quantization

Quantization methods are generally categorized into non-uniform and uniform schemes. Non-uniform approaches may better capture the distribution of parameters and achieve superior performance, but they are typically challenging to deploy on hardware. In contrast, uniform quantization is a linear mapping method with constant spacing between quantization levels, thereby simplifying hardware implementation. In this study, we adopt uniform quantization (Zhu et al. 2017) for both weights and membrane potentials to maximize efficiency. The uniform quantization for parameters in layer l (i.e., weights or membrane potential, $\mathbf{X}^l$) can be formulated as follows:

$$\mathbf{X}^l_{int} = \left\lfloor \text{clip}(\frac{\mathbf{X}^l}{\alpha},\ -2^{b-1},\ 2^{b-1}-1) \right\rceil,\ \hat{\mathbf{X}}^l = \alpha\mathbf{X}^l_{int}. \tag{4}$$

Here, $\mathbf{X}^l_{int}$ denotes the quantized integer values of the weights or membrane potentials in layer l, b specifies the quantization bit precision, and α is a scaling factor used to reduce quantization error and restore numerical precision, i.e., by mapping $\mathbf{X}^l_{int}$ to $\hat{\mathbf{X}}^l$. Omitting the dequantization can lead to substantial accuracy degradation at inference (Krishnamoorthi 2018). Moreover, $\text{clip}(\cdot)$ is a clipping operator, and $\lfloor\cdot\rceil$ denotes the rounding operator. Due to

the non-differentiability of these two operators, the straight-through estimator (STE) (Bengio, Léonard, and Courville 2013) is adopted to approximate gradient propagation. Eq. 4 is applied to all weights and membrane potentials in the baseline model, i.e., $\hat{\mathbf{U}}^l = \alpha_{\mathbf{u}} \mathbf{U}^l_{int}$, $\hat{\mathbf{W}}^l = \alpha_{\mathbf{w}} \mathbf{W}^l_{int}$.

To avoid floating-point operations during the dequantization process, we employ a shared scaling factor for both weights and membrane potentials (Yin et al. 2024), i.e., $\alpha_{\mathbf{q}} = \alpha_{\mathbf{u}} = \alpha_{\mathbf{w}}$. Thus, the membrane potential update and spike generation mechanisms in the HardF-SNN baseline can be formulated as follows:

$$\mathbf{H}^l[t] = \mathbf{U}^l_{int}[t-1] + \mathbf{W}^l_{int}\mathbf{S}^{l-1}[t], \tag{5}$$

$$\tilde{\mathbf{U}}^l[t] = \alpha_{\mathbf{q}} \mathbf{H}^l[t], \tag{6}$$

$$\mathbf{S}^l[t] = \begin{cases} 1, & \text{if } \mathbf{H}^l[t] \geq \theta/\alpha_{\mathbf{q}} \\ 0, & \text{otherwise} \end{cases}. \tag{7}$$

Here, $\mathbf{H}^l[t]$ denotes the integer membrane potential. While shared-scale uniform quantization enables integer-only inference, it significantly compresses the dynamic range of membrane potentials due to the distinct distributions between weights and membrane potentials. This restricts the model's representational capacity and degrades performance. In the next section, we address this issue by effectively scaling the dynamic range of the membrane potential.

Batch Normalization Folding

BN is currently the most widely used normalization technique in SNNs and has been extensively applied in spiking VGGNet (Wu et al. 2018b), spiking ResNet (Zheng et al. 2021; Fang et al. 2022), and spiking Transformer architectures (Yao et al. 2023; Zhou et al. 2022). To accurately simulate quantization effects in the forward pass of training, SNN models with BN should first perform BN folding (Ioffe and Szegedy 2015), and then apply quantization to the folded weights (Jacob et al. 2018). The BN-folded weight $\mathbf{W}_{fold}$ can be reformulated as follows:

$$\mathbf{W}_{fold} = \frac{\gamma \mathbf{W}}{\sqrt{EMA(\sigma^2) + \epsilon}}. \tag{8}$$

Here, γ are floating-point linear affine factors, $EMA(\sigma^2)$ denotes the exponential moving average estimates of the variance of the convolution outputs across batches, and ϵ is a small constant for numerical stability.

Subsequently, Eq. 4 is applied to the BN-folded weights $\mathbf{W}_{fold}$, yielding the dequantized BN-folded weights $\hat{\mathbf{W}}_{fold}$. Although simulating BN folding ensures that BN is implemented as in actual inference engines, it simultaneously diminishes its normalization effect during training. This leads to performance degradation and may even render the model untrainable. In the next section, we propose an integer-only BN to enable stable model training while supporting hardware-friendly implementation.

Methodology

In this section, we analyze and resolve the issues in shared-scale quantization and BN folding that lead to performance degradation and training failure. To improve model performance, we propose a proportional shared-scale quantization to better preserve the distinct dynamic ranges of weights and membrane potential for improved representation. To enable integer-only inference, we introduce integer-only BN to ensure stable model training and hardware efficiency.

Proportional Shared-Scale Quantization

Problem analysis Shared-scale quantization maximizes both computational and storage efficiency. However, using the same scaling factor for both membrane potentials and weights significantly compresses their dynamic range, limiting representational capacity. To intuitively demonstrate the impact of the shared scaling factor, we analyze the data distributions of membrane potentials and weights using representative examples. Figure 1(a) shows the full-precision distributions of weights and membrane potentials from the 3rd layer of ResNet19 on CIFAR100 and the 5th layer of VGG16 on TinyImageNet. It is observed that the weights exhibit a relatively narrow and centralized distribution, whereas the membrane potentials exhibit a significantly broader distribution with a pronounced long tail, indicating a much larger dynamic range. Figure 1(b) shows the distributions of weights and membrane potentials after 4-bit shared-scale quantization. Due to the shared scaling factor, the membrane potentials are compressed into the same dynamic range as the weights. This causes substantial clipping of their long-tailed distribution, thereby significantly reducing the dynamic range of membrane potentials.

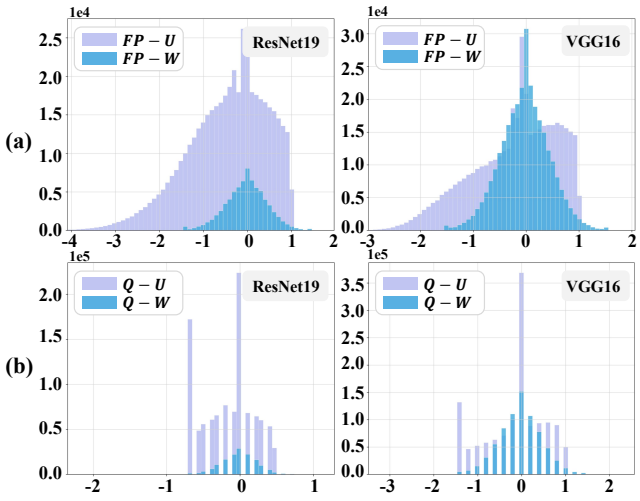

Figure 1: (a) Distinct distributions of full-precision weights and membrane potentials. (b) Distributions of weights and membrane potentials after 4-bit shared-scale quantization.

Method To address the limited dynamic range, we propose an effective proportional shared-scale quantization for HardF-SNN, which enables more efficient quantization for both weights and membrane potentials. Specifically, we introduce a scale coefficient η to ensure that the scaling factors used for quantizing weights and membrane potentials

are proportionally related rather than simply set to be equal, i.e., $\alpha_{\mathbf{u}} = \eta\alpha_{\mathbf{w}}$. We provide two options for η: (1) rounding the maximum absolute value: $\lfloor \mathrm{Max}(|\mathbf{U}^l|) \rceil$; (2) the learnable parameter in the form of a power of two Pow2. The impact of these two options on performance will be explored in the experimental section. Therefore, we formulate the membrane potential update and spike generation process as:

$$\mathbf{H}^l[t] = \eta\mathbf{U}^l_{int}[t-1] + \mathbf{W}^l_{int}\mathbf{S}^{l-1}[t], \tag{9}$$

$$\tilde{\mathbf{U}}^l[t] = \alpha_{\mathbf{w}}\mathbf{H}^l[t], \tag{10}$$

$$\mathbf{S}^l[t] = \begin{cases} 1, & \text{if } \mathbf{H}^l[t] \geq \theta/\alpha_{\mathbf{w}} \\ 0, & \text{otherwise} \end{cases}. \tag{11}$$

η is represented as either an integer or a power of two, both of which are favorable for hardware implementation. Benefit from the proportional shared scaling factor, the weights and membrane potential can better preserve the distinct dynamics ranges, thereby enhancing the performance of the HardF-SNN baseline.

Integer-only Batch Normalization

Problem analysis Simulating BN folding allows BN to be implemented consistently with practical inference engines, but it introduces performance degradation and even renders models untrainable at extremely low bit-widths. Figure 2 presents the accuracy under 8-bit, 4-bit, and 2-bit quantization with and without BN folding on CIFAR100 with ResNet19 and on TinyImageNet with VGG16. Compared to models without BN folding, those with BN folding exhibit lower accuracy at 8-bit and 4-bit precision, and become untrainable at 2-bit precision due to convergence failure.

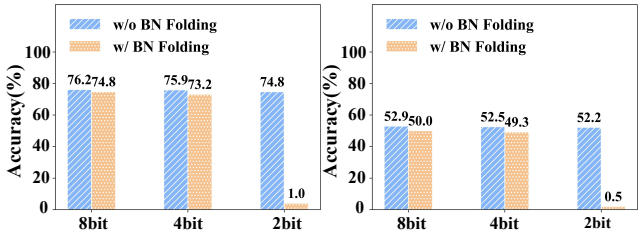

Figure 2: Accuracy on CIFAR100 with ResNet19 and on TinyImageNet with VGG16 under quantization.

To analyze this phenomenon, we compare the distributions of the weights $\mathbf{W}$ and the BN-folded weights $\mathbf{W}_{fold}$, as illustrated in Figure 3(a). It can be observed that the weights $\mathbf{W}$ exhibit a smoother and broader distribution, while the BN-folded weights $\mathbf{W}_{fold}$ are highly concentrated near zero, resulting in a sharper and more peaked distribution. Figure 3(b) shows the weight distributions after 2-bit quantization. The quantized weights $\hat{\mathbf{W}}$ retain a reasonable distribution across multiple quantization bins, enabling more effective utilization of the limited quantization levels. Moreover, BN remains capable of dynamically regulating the distributions of convolutional outputs, providing the model with greater flexibility to ensure effective convergence and stable training. In contrast, the distribution of quantized BN-folded weights $\mathbf{W}_{fold}$ becomes highly concentrated, with the majority of weights collapsing into a single quantization bin. This results in many gradient updates becoming ineffective, thereby severely hampering model convergence (Courbariaux et al. 2016). In this case, BN loses its ability to regulate the output distribution.

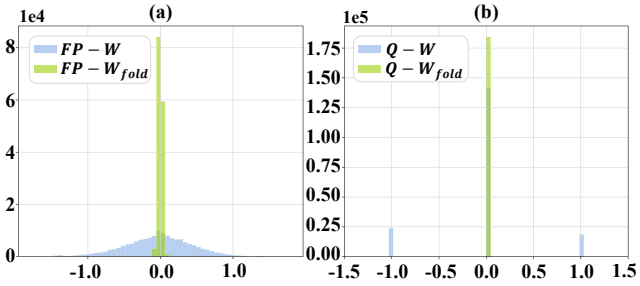

Figure 3: (a) Distribution of raw weights and BN-folded weights before quantization. (b) Distribution of raw weights and BN-folded weights after 2-bit quantization.

Method Based on the above analysis, quantized SNN models without BN folding facilitate effective training. However, this inevitably introduces hardware-unfriendly floating-point arithmetic, thereby hindering deployment on integer-only hardware platforms. To facilitate stable model training while ensuring hardware-friendly implementation, we propose an integer-only BN scheme. The detailed algorithm is provided in Algorithm 1. For a given mini-batch of convolutional outputs $x_{b,c}$ (where b is the number of samples in the mini-batch and c is the channel index), the channel-wise mini-batch mean μ_c and variance σ_c^2 are first calculated. Subsequently, the BN scaling factor is first divided by the normalization term $\sqrt{\sigma_c^2 + \epsilon}$, and then approximated to the nearest power-of-two value using the Pow2STE operation. This allows the multiplication to be replaced with a low-cost bit-shifting operation. Meanwhile, the BN bias term is adjusted by subtracting $\gamma_c \frac{\mu_c}{\sqrt{\sigma_c^2+\epsilon}}$, and then converted to an integer using the RoundSTE operation, thereby ensuring hardware-friendly implementation. In the experimental section, we will demonstrate in the experimental section that the proposed integer-only BN ensures stable convergence of HardF-SNN. Furthermore, it can be efficiently implemented on resource-constrained hardware platforms.

Experiments

In this section, we first describe the experimental setup. Then, we perform extensive ablation studies to validate the effectiveness of proportional shared-scale quantization and integer-only BN. Finally, we evaluate HardF-SNN by comparing it against the state-of-the-art methods.

Experimental Setup

We evaluate the proposed method on image classification tasks, including static datasets such as CIFAR (Krizhevsky, Hinton et al. 2009), TinyImageNet, and ImageNet-1k (Deng et al. 2009), as well as the neuromorphic dataset DVS-CIFAR10 (Li et al. 2017). For network architectures, we

Algorithm 1: Integer-only batch normalization.

Input: Mini-batch convolutional outputs of $x_{b,c}$, trainable parameters of BN: γ_c, β_c. RoundSTE$(x) = \text{round}(x)$, Pow2STE$(x) = \text{sign}(x)2^{\text{round}(log2|x|)}$.
Output: $y_{b,c} = I\_BN(x_{b,c})$.

$\mu_c = \frac{1}{b}\sum_{i=1}^{b} x_{i,c}$ ▷ mini-batch mean
$\sigma_c^2 = \frac{1}{b}\sum_{i=1}^{b}(x_{i,c} - \mu_c)^2$ ▷ mini-batch variance
$\hat{\gamma}_c = \text{Pow2STE}(\frac{\gamma_c}{\sqrt{\sigma_c^2+\epsilon}})$ ▷ power-of-two γ
$\hat{\beta}_c = \text{RoundSTE}(\beta_c - \gamma_c\frac{\mu_c}{\sqrt{\sigma_c^2+\epsilon}})$ ▷ integer β
$y_{b,c} = \hat{\gamma}_c \cdot x_{b,c} + \hat{\beta}_c$ ▷ integer-only BN

adopt the classical VGGNet and Spiking ResNet (Zheng et al. 2021). We conduct comprehensive experiments under three different bit-width configurations: 2w-2u, 4w-4u, and 8w-8u, where "2w-2u" denotes that HardF-SNN employs 2-bit synaptic weights and 2-bit membrane potentials. All experiments employ the AdamW optimizer (Kingma and Ba 2017) with an initial learning rate of 0.01, decayed using a cosine annealing schedule (Loshchilov and Hutter 2017). We conducted training on eight 24GB 3090 GPUs.

Ablation Study

To thoroughly validate the effectiveness of HardF-SNN, we perform extensive ablation studies. For clearer experimental descriptions, we define proportional shared-scale quantization as **PSSQ**, and Integer-only BN as **I-BN**. All ablation studies are conducted on the CIFAR-100 dataset using the standard ResNet19 architecture.

Analysis of two options for η We compare the performance of HardF-SNN under different settings of η to identify the optimal option as the default experimental configuration, as depicted in Figure 4. Both approaches demonstrate comparable performance. Compared to the integer multiplication used in $\lfloor \text{Max}(|\mathbf{U}|) \rceil$, Pow2 enables lower-cost shift operations. Therefore, we adopt the power-of-two formulation as the default setting in our experiments.

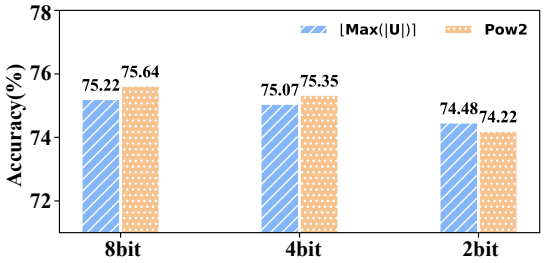

Figure 4: Analysis of two options for η.

Effectiveness of HardF-SNN As shown in Table 1, we conduct extensive ablation experiments to verify the effectiveness of PSSQ and I-BN in HardF-SNN. First, we evaluate the effectiveness of PSSQ. Compared with the HardF-SNN baseline, PSSQ achieves performance gains of 0.80% and 1.78% under the 8w-8u and 4w-4u settings, respectively. Due to the impact of BN folding, neither PSSQ nor the baseline can be trained successfully under the 2w-2u configuration. Second, we evaluate the effectiveness of I-BN. Compared to the baseline model, I-BN also yields a slight performance improvement under both 8w-8u and 4w-4u configurations. Notably, I-BN enables stable model training at the 2w-2u configuration and achieves 73.93% accuracy. By integrating both methods into the baseline, we observe performance improvements of 0.82% and 2.20% under the 8w-8u and 4w-4u configurations, respectively. Additionally, under the 2w-2u configuration, the performance reaches 74.22%, achieving a 0.29% improvement compared to the I-BN method. These findings further highlight the critical importance of these techniques in enabling effective training and maintaining high performance for HardF-SNN under various quantization settings.

Method	Bit-Width	Acc.(%)	Compared to	Incr.(%)
baseline	8w-8u	74.82	-	-
w/PSSQ		75.62	baseline	0.80 ↑
w/I-BN		75.04	baseline	0.22 ↑
w/both		75.64	baseline	0.82 ↑
baseline	4w-4u	73.15	-	-
w/PSSQ		74.93	baseline	1.78 ↑
w/I-BN		74.61	baseline	1.46 ↑
w/both		75.35	baseline	2.20 ↑
baseline	2w-2u	-	-	-
w/PSSQ		-	-	-
w/I-BN		73.93	-	-
w/both		74.22	I-BN	0.29 ↑

Table 1: Ablation study on the effectiveness of two proposed methods.'w/' denotes 'with'.

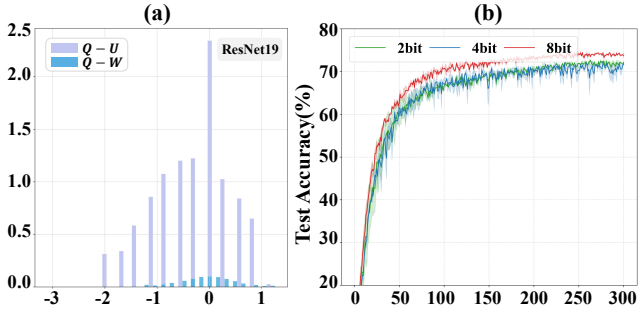

Figure 5: (a) Distributions of weights and membrane potentials after 4-bit shared-scale quantization using the PSSQ. (b) Test accuracy curves across three runs with different random seeds using I-BN.

Visualization analysis To demonstrate that the proposed PSSQ and I-BN effectively address the aforementioned issues, we visualize the membrane potential distributions before and after quantization, as well as test accuracy curves of HardF-SNN. Figure 5(a) shows that the PSSQ method results in a broader distribution of membrane potentials, indicating an improved dynamic range. Furthermore, Fig-

Datasets	Methods	Model	Bits-Width	Q-BN	INT	T	Accuracy(%)	Memory(MB)
CIFAR10	(Hasssan et al. 2024)	ResNet19	2w-2u	✗	✗	2	92.89	68.28
	(Yin et al. 2024)	ResNet19	4w-4u	✓	✓	4	90.79	116.07
	(Wei et al. 2025b)	ResNet20	8w-32u	✗	✗	2	94.56	341.84
			4w-32u	✗	✗	2	94.65	339.90
			2w-32u	✗	✗	2	94.44	338.94
	Proposed HardF-SNN	ResNet19	8w-8u	✓	✓	4	**95.47**	**209.11**
			4w-4u	✓	✓	4	**95.53**	**110.80**
			2w-2u	✓	✓	4	**94.49**	**68.28**
CIFAR100	(Yoo and Jeong 2023)	VGG16	2w-32u	✗	✗	32	66.53	139.52
	(Hasssan et al. 2024)	ResNet19	4w-2u	✗	✗	2	71.87	71.3
	(Wei et al. 2025b)	ResNet20	8w-32u	✗	✗	2	74.78	365.06
			4w-32u	✗	✗	2	74.73	363.13
			2w-32u	✗	✗	2	73.89	362.16
	Proposed HardF-SNN	ResNet19	8w-8u	✓	✓	4	**75.64**	**210.43**
			4w-4u	✓	✓	4	**75.35**	**114.82**
			2w-2u	✓	✓	4	**74.22**	**68.29**
Tiny-ImageNet	(Yin et al. 2024)	VGG16	8w-8u	✓	✓	4	50.18	166.45
			4w-4u	✓	✓	4	49.36	91.55
	(Wei et al. 2025b)	VGG16	8w-32u	✗	✗	4	51.99	300.35
			4w-32u	✗	✗	4	51.78	298.85
			2w-32u	✗	✗	4	51.67	298.09
	Proposed HardF-SNN	VGG16	8w-8u	✓	✓	4	**52.38**	**166.45**
			4w-4u	✓	✓	4	**52.07**	**91.55**
			2w-2u	✓	✓	4	**51.87**	**54.73**
ImageNet	(Yoo and Jeong 2023)	ResNet18	2w-32u	✗	✗	4	54.34	1165.03
	(Wei et al. 2025b)	ResNet18	8w-32u	✗	✗	4	**61.36**	1175.52
			4w-32u	✗	✗	4	58.06	1169.95
	Proposed HardF-SNN	ResNet18	8w-8u	✓	✓	4	59.52	**327.31**
			4w-4u	✓	✓	4	**58.97**	**182.21**
			2w-2u	✓	✓	4	**57.71**	**107.75**
DVS-CIFAR10	(Yoo and Jeong 2023)	VGG16	2w-32u	✗	✗	16	74.70	317.11
	(Hasssan et al. 2024)	VGG9	2w-2u	✗	✗	10	77.94	26.52
	(Wei et al. 2025b)	VGG9	8w-32u	✗	✗	10	75.90	**40.08**
			4w-32u	✗	✗	10	75.40	**39.97**
			2w-32u	✗	✗	10	74.90	39.92
	Proposed HardF-SNN	VGG9	8w-8u	✓	✓	10	**79.53**	80.48
			4w-4u	✓	✓	10	**78.74**	44.08
			2w-2u	✓	✓	10	**78.35**	**26.52**

Table 2: Performance comparison on static and neuromorphic datasets. 'Q-BN' refers to BN quantization, and 'INT' denotes integer-only quantization. Memory footprints are calculated using a batch size of 128.

ure 5(b) illustrates the test accuracy curves of I-BN method, demonstrating its capability to achieve stable training and model convergence.

Performance Comparison

As shown in Table 2, we compare HardF-SNN against existing approaches in terms of accuracy and memory footprint to validate its effectiveness and efficiency. Compared to existing integer-only quantization methods, HardF-SNN successfully enables inference at extremely low bit-widths (e.g., 2w-2u) while consistently achieving superior performance under identical model configurations.

Moreover, it also exhibits superior performance relative to quantization approaches that rely on floating-point arithmetic. Specifically, on CIFAR10 and CIFAR100, HardF-SNN surpasses prior state-of-the-art methods (Wei et al. 2025b; Hasssan et al. 2024), significantly reducing both memory footprint (e.g., CIFAR10: 2w-2u configuration, Acc = 94.49%, Memory footprint = 68.28MB; CIFAR100: 4w-4u configuration, Acc = 75.35%, Memory footprint = 114.82MB). On TinyImageNet, HardF-SNN demonstrates consistently superior accuracy and reduced memory footprint across all configurations when compared to (Wei et al. 2025b) under identical network architecture and timesteps. On the ImageNet dataset, HardF-SNN demonstrates superior performance over weight-only quantization methods (Yoo and Jeong 2023) and achieves competitive accuracy with (Wei et al. 2025b), while simultaneously reducing memory footprint by 10.9×. On the DVS-CIFAR10 dataset, HardF-SNN similarly achieves state-of-the-art performance while delivering substantial efficiency improvements. These extensive results show that HardF-SNN achieves a superior balance between efficiency and accuracy, while offering high compatibility with hardware accelerators, thereby paving the way for practical deployment of SNNs on resource-constrained hardware platforms.

Dataset	Work	Device	Network	Bits	Accuracy(%)	Latency(ms)	Throughput(GOPS)	Efficiency(GOPS/W)
CIFAR10	SyncNN	xczu9eg	VGG13	4bit	90.37	16.13	28.5	71.2
	FireFly	xczu3eg	VGG7	8bit	91.36	4.16	273.1	107.1
	SiBrain	xc7v2000t	VGG11	8bit	90.25	18.90	130.8	84.2
	Ours	xc7z035	VGG7	8bit	**91.91**	3.99	284.74	72.64
				4bit	**91.83**	2.22	511.75	129.23
				2bit	**90.68**	**1.23**	**923.66**	**224.19**
CIFAR100	E3NE	xcvu13p	VGG11	6bit	65.00	163.90	3.58	0.72
	FireFly	xczu3eg	VGG11	8bit	64.28	8.52	275.1	107.9
	SiBrain	xc7v2000t	VGG11	8bit	66.97	19.00	130.1	83.4
	Ours	xc7z035	VGG11	8bit	**67.26**	7.88	297.44	79.74
				4bit	**66.55**	4.23	554.10	139.92
				2bit	**65.49**	**2.36**	**993.16**	**241.06**

Table 3: Comparison with representative hardware accelerator.

Practical Hardware Deployment

In this section, we deploy HardF-SNN on a dedicated SNN hardware accelerator to validate its effectiveness on resource-constrained hardware platforms.

Hardware Architecture

The overall architecture and main components are shown in Figure 6. The processing system (PS) acts as the controller for system state control and external memory access. The programmable logic (PL) accelerates the SNN inference. The execution of the accelerator is managed by the global controller. Input feature map (IFmap), weight, and output feature map (OFmap) are transferred through AXI-DataMover between the off-chip memory and the global buffers. The row stationary (RS) systolic array consists of multiple processing elements (PEs) and is specifically designed to handle the arithmetic operations of SNNs. Synaptic input integration is achieved through the static and dynamic configuration of DSPs (Li et al. 2023). According to feature size and systolic array dimension, weight data and IFmap data are dynamically reconfigured by the configuration unit and mapped into the RS systolic array with RS dataflow (Chen et al. 2016). The multi-functional unit is responsible for integer-only BN, membrane potential update, and spike generation. Finally, the generated output spikes are transferred to the off-chip memory, where they are stored for use in subsequent computations.

Performance Evaluation

As shown in Table 3, we compare our implementation in terms of accuracy, latency, throughput, and energy efficiency under different bit-width settings with representative FPGA-based SNN accelerators, including SyncNN (Panchapakesan, Fang, and Li 2021), E3NE (Gerlinghoff et al. 2022), FireFly (Li et al. 2023), and SiBrain (Chen et al. 2024). Across both CIFAR10 and CIFAR100 datasets, our 8-bit implementation achieves the highest accuracy compared to FireFly and SiBrain, while also delivering competitive latency, throughput, and energy efficiency. In contrast, our 2-bit implementation achieves significant improvements in latency, throughput, and energy efficiency. For example, in terms of inference latency, our method achieves a 15× reduction compared to SiBrain on the CIFAR10 dataset. Regarding throughput and energy efficiency, we achieve 3.6× and 2.2× improvements, respectively, over FireFly on the CIFAR100 dataset. In summary, the 4-bit implementation offers an optimal balance between accuracy and hardware performance. These results further indicate that HardF-SNN lays a solid foundation for the efficient deployment of SNNs on resource-constrained hardware platforms.

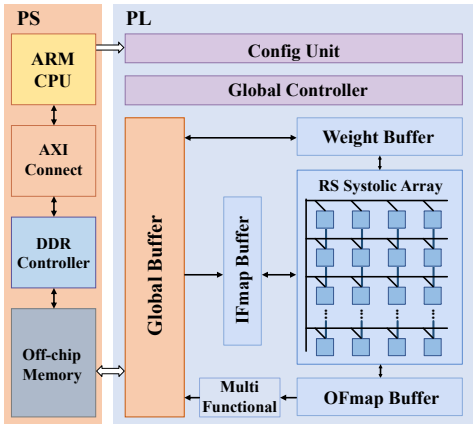

Figure 6: Overall architecture of dedicated SNN accelerator.

Conclusion

In this paper, we first propose a hardware-friendly baseline model—HardF-SNN, which maximizes efficiency while enabling integer-only inference. Through extensive experiments and analysis, we identify that the HardF-SNN baseline suffers from performance degradation and even training failure. To enhance performance, we propose a proportional shared-scale quantization. To enable integer-only inference, we introduce an Integer-only BN. By integrating the above two methods, HardF-SNN achieves both high efficiency and superior performance. Finally, HardF-SNN is deployed on a dedicated FPGA-based hardware accelerator, demonstrating the state-of-the-art performance, thereby highlighting its strong potential for energy-efficient edge applications.

Acknowledgments

This work is supported in part by the National Natural Science Foundation of China (62220106008 and 62576080), in part by the State Key Laboratory of Brain Cognition and Brain-inspired Intelligence Technology, Grant No. SKLBI-K2025010.

References

Akopyan, F.; Sawada, J.; Cassidy, A.; Alvarez-Icaza, R.; Arthur, J.; Merolla, P.; Imam, N.; Nakamura, Y.; Datta, P.; Nam, G.-J.; Taba, B.; Beakes, M.; Brezzo, B.; Kuang, J. B.; Manohar, R.; Risk, W. P.; Jackson, B.; and Modha, D. S. 2015. TrueNorth: Design and Tool Flow of a 65 mW 1 Million Neuron Programmable Neurosynaptic Chip. *Trans. Comp.-Aided Des. Integ. Cir. Sys.*, 34(10): 1537–1557.

Aung, M. T. L.; Gerlinghoff, D.; Qu, C.; Yang, L.; Huang, T.; Goh, R. S. M.; Luo, T.; and Wong, W.-F. 2023. Deepfire2: A convolutional spiking neural network accelerator on fpgas. *IEEE Transactions on Computers*, 72(10): 2847–2857.

Bengio, Y.; Léonard, N.; and Courville, A. 2013. Estimating or Propagating Gradients Through Stochastic Neurons for Conditional Computation. arXiv:1308.3432.

Cao, H.; Zhou, Z.; Wei, W.; Belatreche, A.; Liang, Y.; Zhang, D.; Zhang, M.; Yang, Y.; and Li, H. 2025. Binary Event-Driven Spiking Transformer. arXiv:2501.05904.

Chen, Y.; Liu, H.; Shi, K.; Zhang, M.; and Qu, H. 2023. Spiking neural network with working memory can integrate and rectify spatiotemporal features. *Frontiers in Neuroscience*, Volume 17 - 2023.

Chen, Y.; Ye, W.; Liu, Y.; and Zhou, H. 2024. SiBrain: A Sparse Spatio-Temporal Parallel Neuromorphic Architecture for Accelerating Spiking Convolution Neural Networks With Low Latency. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 1–13.

Chen, Y.-H.; Krishna, T.; Emer, J. S.; and Sze, V. 2016. Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks. *IEEE journal of solid-state circuits*, 52(1): 127–138.

Courbariaux, M.; Hubara, I.; Soudry, D.; El-Yaniv, R.; and Bengio, Y. 2016. Binarized Neural Networks: Training Deep Neural Networks with Weights and Activations Constrained to +1 or -1. arXiv:1602.02830.

Davies, M.; Srinivasa, N.; Lin, T.-H.; Chinya, G.; Cao, Y.; Choday, S. H.; Dimou, G.; Joshi, P.; Imam, N.; Jain, S.; Liao, Y.; Lin, C.-K.; Lines, A.; Liu, R.; Mathaikutty, D.; McCoy, S.; Paul, A.; Tse, J.; Venkataramanan, G.; Weng, Y.-H.; Wild, A.; Yang, Y.; and Wang, H. 2018. Loihi: A Neuromorphic Manycore Processor with On-Chip Learning. *IEEE Micro*, 38(1): 82–99.

Deng, J.; Dong, W.; Socher, R.; Li, L.-J.; Li, K.; and Fei-Fei, L. 2009. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, 248–255. Ieee.

Deng, L.; Wang, G.; Li, G.; Li, S.; Liang, L.; Zhu, M.; Wu, Y.; Yang, Z.; Zou, Z.; Pei, J.; Wu, Z.; Hu, X.; Ding, Y.; He, W.; Xie, Y.; and Shi, L. 2020a. Tianjic: A Unified and Scalable Chip Bridging Spike-Based and Continuous Neural Computation. *IEEE Journal of Solid-State Circuits*, 55(8): 2228–2246.

Deng, L.; Wu, Y.; Hu, Y.; Liang, L.; Li, G.; Hu, X.; Ding, Y.; Li, P.; and Xie, Y. 2020b. Comprehensive SNN Compression Using ADMM Optimization and Activity Regularization. arXiv:1911.00822.

Deng, L.; Wu, Y.; Hu, Y.; Liang, L.; Li, G.; Hu, X.; Ding, Y.; Li, P.; and Xie, Y. 2021. Comprehensive snn compression using admm optimization and activity regularization. *IEEE transactions on neural networks and learning systems*, 34(6): 2791–2805.

Fang, W.; Yu, Z.; Chen, Y.; Huang, T.; Masquelier, T.; and Tian, Y. 2022. Deep Residual Learning in Spiking Neural Networks. arXiv:2102.04159.

Gerlinghoff, D.; Wang, Z.; Gu, X.; Goh, R. S. M.; and Luo, T. 2022. E3NE: An End-to-End Framework for Accelerating Spiking Neural Networks With Emerging Neural Encoding on FPGAs. *IEEE Transactions on Parallel and Distributed Systems*, 33(11): 3207–3219.

Guo, Q.; Wang, Y.; and Cui, X. 2021. Integer-Only Neural Network Quantization Scheme Based on Shift-Batch-Normalization. arXiv:2106.00127.

Hasssan, A.; Meng, J.; Anupreetham, A.; and Seo, J.-s. 2024. IM-SNN: Memory-Efficient Spiking Neural Network with Low-Precision Membrane Potentials and Weights. In *2024 International Conference on Neuromorphic Systems (ICONS)*, 148–155.

Ioffe, S.; and Szegedy, C. 2015. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. arXiv:1502.03167.

Jacob, B.; Kligys, S.; Chen, B.; Zhu, M.; Tang, M.; Howard, A.; Adam, H.; and Kalenichenko, D. 2018. Quantization and training of neural networks for efficient integer-arithmetic-only inference. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2704–2713.

Kingma, D. P.; and Ba, J. 2017. Adam: A Method for Stochastic Optimization. arXiv:1412.6980.

Krishnamoorthi, R. 2018. Quantizing deep convolutional networks for efficient inference: A whitepaper. *arXiv preprint arXiv:1806.08342*.

Krizhevsky, A.; Hinton, G.; et al. 2009. Learning multiple layers of features from tiny images.

Li, H.; Liu, H.; Ji, X.; Li, G.; and Shi, L. 2017. Cifar10-dvs: an event-stream dataset for object classification. *Frontiers in neuroscience*, 11: 309.

Li, J.; Shen, G.; Zhao, D.; Zhang, Q.; and Zeng, Y. 2023. Firefly: A high-throughput hardware accelerator for spiking neural networks with efficient dsp and memory optimization. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 31(8): 1178–1191.

Liu, H.; Chen, Y.; Zeng, Z.; Zhang, M.; and Qu, H. 2023. A Low Power and Low Latency FPGA-Based Spiking Neural Network Accelerator. In *2023 International Joint Conference on Neural Networks (IJCNN)*, 1–8.

Liu, Y.; Chen, W.; Liu, H.; Zhang, Y.; Zhang, M.; and Qu, H. 2024. Biologically Plausible Sparse Temporal Word Representations. *IEEE Transactions on Neural Networks and Learning Systems*, 35(11): 16952–16959.

Loshchilov, I.; and Hutter, F. 2017. SGDR: Stochastic Gradient Descent with Warm Restarts. arXiv:1608.03983.

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

Panchapakesan, S.; Fang, Z.; and Li, J. 2021. SyncNN: Evaluating and Accelerating Spiking Neural Networks on FPGAs. In *2021 31st International Conference on Field-Programmable Logic and Applications (FPL)*, 286–293.

Pei, Y.; Xu, C.; Wu, Z.; Liu, Y.; and Yang, Y. 2023. ALBSNN: ultra-low latency adaptive local binary spiking neural network with accuracy loss estimator. *Frontiers in Neuroscience*, 17.

Putra, R. V. W.; and Shafique, M. 2021. Q-SpiNN: A Framework for Quantizing Spiking Neural Networks. In *2021 International Joint Conference on Neural Networks (IJCNN)*, 1–8. IEEE.

Shi, K.; Hou, M.; Luo, X.; Zhang, D.; Liu, H.; and Wang, J. 2025. Spike-VAD: Efficient and Robust Spiking Neural Network for Voice Activity Detection. *IEEE Transactions on Cognitive and Developmental Systems*, 1–13.

Shi, K.; Liu, H.; Chen, Y.; and Qu, H. 2024. HL-ESViT: High-Low Frequency Efficient Spiking Vision Transformer. In *2024 International Joint Conference on Neural Networks (IJCNN)*, 1–8.

Sun, Q.; Lu, C.; Chen, W.; Wei, W.; Wang, J.; Zhang, J.; Liu, X.; Ye, Y.; Yang, Y.; and Zhang, M. 2025. Temporal-coded Spiking Transformer. In *Proceedings of the 33rd ACM International Conference on Multimedia*, MM '25, 2616–2624. New York, NY, USA: Association for Computing Machinery. ISBN 9798400720352.

Tan, P.-Y.; and Wu, C.-W. 2023. A low-bitwidth integer-stbp algorithm for efficient training and inference of spiking neural networks. In *Proceedings of the 28th Asia and South Pacific Design Automation Conference*, 651–656.

Wang, S.; Zhang, D.; Belatreche, A.; Xiao, Y.; Qing, H.; Wei, W.; Zhang, M.; and Yang, Y. 2025a. Ternary spike-based neuromorphic signal processing system. *Neural Networks*, 187: 107333.

Wang, S.; Zhang, M.; Zhang, D.; Belatreche, A.; Xiao, Y.; Liang, Y.; Shan, Y.; Sun, Q.; Zhang, E.; and Yang, Y. 2025b. Spiking vision transformer with saccadic attention. *arXiv preprint arXiv:2502.12677*.

Wang, S.; Zheng, H.; Chen, Y.; Belatreche, A.; Wang, G.; Jin, Y.; Wu, J.; Zhang, M.; Yang, Y.; and Li, H. 2025c. SNN-FT: Temporal-Coded Spiking Neural Networks for Fourier Transform. *IEEE Transactions on Neural Networks and Learning Systems*.

Wang, Y.; Liu, H.; Zhang, M.; Luo, X.; and Qu, H. 2024. A universal ANN-to-SNN framework for achieving high accuracy and low latency deep Spiking Neural Networks. *Neural Networks*, 174: 106244.

Wei, W.; Zhang, M.; Zhang, J.; Belatreche, A.; Wang, S.; Shan, Y.; Liu, H.; Cao, H.; Wang, G.; Yang, Y.; and Li, H. 2025a. S^2NN: Sub-bit Spiking Neural Networks. arXiv:2509.24266.

Wei, W.; Zhang, M.; Zhou, Z.; Belatreche, A.; Shan, Y.; Liang, Y.; Cao, H.; Zhang, J.; and Yang, Y. 2025b. QP-SNN: Quantized and Pruned Spiking Neural Networks. arXiv:2502.05905.

Wu, Y.; Deng, L.; Li, G.; Zhu, J.; and Shi, L. 2018a. Spatio-Temporal Backpropagation for Training High-Performance Spiking Neural Networks. *Frontiers in Neuroscience*, 12.

Wu, Y.; Deng, L.; Li, G.; Zhu, J.; and Shi, L. 2018b. Spatio-temporal backpropagation for training high-performance spiking neural networks. *Frontiers in neuroscience*, 12: 331.

Xue, L.; Liu, H.; Wang, J.; and Qu, H. 2024. Reasonable Gradients for Online Training Algorithms in Spiking Neural Networks. In *ECAI 2024*, 2709–2716. IOS Press.

Yan, Y.; Yang, Q.; Wu, Y.; Liu, H.; Zhang, M.; Li, H.; Tan, K. C.; and Wu, J. 2025. Efficient and robust temporal processing with neural oscillations modulated spiking neural networks. *Nature communications*, 16(1): 8651.

Yao, M.; Hu, J.; Zhou, Z.; Yuan, L.; Tian, Y.; Xu, B.; and Li, G. 2023. Spike-driven Transformer. arXiv:2307.01694.

Yin, R.; Li, Y.; Moitra, A.; and Panda, P. 2024. MINT: Multiplier-less INTeger Quantization for Energy Efficient Spiking Neural Networks. In *2024 29th Asia and South Pacific Design Automation Conference (ASP-DAC)*, 830–835. IEEE.

Yoo, D.; and Jeong, D. S. 2023. CBP-QSNN: Spiking Neural Networks Quantized Using Constrained Backpropagation. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, 13(4): 1137–1146.

Zhang, J.; Zhou, X.; Wang, S.; Wei, W.; Liu, H.; Sun, Q.; Zhang, M.; Yang, Y.; and Li, H. 2025a. Unveiling the Spatial-temporal Effective Receptive Fields of Spiking Neural Networks. arXiv:2510.21403.

Zhang, M.; Luo, X.; Wu, J.; Belatreche, A.; Cai, S.; Yang, Y.; and Li, H. 2025b. Toward Building Human-Like Sequential Memory Using Brain-Inspired Spiking Neural Models. *IEEE Transactions on Neural Networks and Learning Systems*, 36(6): 10143–10155.

Zhang, M.; Wei, W.; Zhou, Z.; Liu, W.; Zhang, J.; Belatreche, A.; and Yang, Y. 2025c. Spike-Driven Lightweight Large Language Model With Evolutionary Computation. *IEEE Transactions on Evolutionary Computation*, 1–1.

Zheng, H.; Wu, Y.; Deng, L.; Hu, Y.; and Li, G. 2021. Going deeper with directly-trained larger spiking neural networks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, 11062–11070.

Zhou, Z.; Zhu, Y.; He, C.; Wang, Y.; Yan, S.; Tian, Y.; and Yuan, L. 2022. Spikformer: When Spiking Neural Network Meets Transformer. arXiv:2209.15425.

Zhu, C.; Han, S.; Mao, H.; and Dally, W. J. 2017. Trained Ternary Quantization. arXiv:1612.01064.