https://doi.org/10.1007/s11704-025-51519-1

RESEARCH ARTICLE

Ternary quantization of spiking neural networks

Yu-Lun WU[1,2], Rui-Rui TAN[1,2], Shu-Hao ZHANG[1,2], Shuang LIANG[1,2], Zi-Ang LIU[3], Zhao WANG[3]✉, Shao-Qun ZHANG[1,2]✉

1. State Key Laboratory of Novel Software Technology, Nanjing University, Nanjing 210023, China
2. School of Intelligent Science and Technology, Nanjing University, Suzhou 215163, China
3. China Mobile Zijin Innovation Institute, Nanjing 211899, China

Received September 22, 2025; accepted November 18, 2025

E-mail: wangzh8@js.chinamobile.com; zhangsq@lamda.nju.edu.cn

Abstract

In recent years, there has emerged an increasing interest in Spiking Neural Networks (SNNs) due to their potential to handle spatio-temporal information and adapt to resource-constrained environments. However, the high memory and computational demands of full-precision embeddings pose challenges for deployment in low-power edge devices. Existing lightweight algorithms of SNNs often pursue memory compression and inference acceleration, leading to precision dropping and uncertainty increasing. To address this challenge, we introduce a ternary quantization method for achieving both lightweight computation and uncertainty reduction of SNNs. The proposed method comprises two quantization training algorithms that apply ternary quantization to synaptic weights and a regularizer for enhancing both accuracy improvement and uncertainty reduction of ternary SNNs. Our method is compatible with various SNN architectures and surrogate-gradient-based training algorithms. Experiments conducted on various datasets demonstrate that our proposed method outperforms existing methods in terms of accuracy, inference efficiency, energy consumption, memory storage, and uncertainty.

Keywords

spiking neural networks; ternary quantization; surrogate gradients; ternarization-aware distillation; uncertainty reduction

1 Introduction

Recent years have witnessed an increasing interest in SNNs as they pose the potential to handle spatio-temporal information and enable efficient hardware implementations [1−4]. However, the substantial memory requirements and high computational demands associated with full-precision embeddings present obstacles when it comes to their deployment in low-power edge devices [5,6]. There has been great progress in developing lightweight computations of SNNs; weight quantization is the typical one, which usually encodes n-bit synaptic weights and spiking excitation. The seminal studies on quantized SNNs contribute to effective memory compression and inference acceleration, including fusing the ADMM optimization method for network pruning and quantization [7], directly transferring quantized ANNs to SNNs [8], and employing the weight-spike dual regulation method based on information entropy theory to enhance the performance of binary quantized models [9]. However, it is observed that quantized SNNs inevitably suffer from significant precision dropping and uncertainty increasing, which limits the scope of weight quantization in SNNs.

In this paper, we focus on the ternary quantization of SNNs to achieve both lightweight computation and uncertainty reduction. The proposed method comprises two training algorithms that apply ternary quantization to synaptic weights and a regularizer for enhancing both accuracy improvement and uncertainty reduction of ternary SNNs. Notice that our method can be grafted onto multiple SNN architectures and surrogate-gradient-based training algorithms. Empirical investigations on real-world datasets show that our proposed methods surpass existing methods in terms of accuracy, inference efficiency, energy consumption, memory storage, and uncertainty.

The rest of this paper is organized as follows. Section 2 reviews the related works. Section 3 proposes our methods for implementing ternary quantization and uncertainty optimization. Section 4 conducts the experiments to demonstrate the effectiveness of the proposed methods. Section 5 concludes our work with prospects.

2 Related work

In recent years, significant studies have been conducted on deep learning equipped with weight quantization, broadly including two categories: Post-Training Quantization (PTQ) and Quantization-Aware Training (QAT). PTQ involves quantizing a fully trained model, while QAT incorporates quantization parameters throughout

the training process to account for the loss induced by parameter quantization. PTQ has seen widespread research and application, such as using SQNR-optimal quantization [10] and applying it to vision transformers [11]. These studies have demonstrated the considerable improvements in computational efficiency and reduction in storage energy consumption achieved through quantization. Another widely explored approach is QAT. Han et al. [12] combined QAT with pruning and Huffman coding. Courbariaux et al. [13,14] proposed a method that introduces binary weights during training, which is later extended to simultaneously binarize both weights and activations. Since then, various works have improved weight quantization techniques and network architectures to address the trade-offs between latency and accuracy [15−17], and deployed quantized models on hardware to verify their significant advantages in terms of computation and storage [18−20]. Among all the quantization schemes, ternary quantization, where weights are constrained to $\{-1, 0, +1\}$ has also gained attention to further balance model accuracy and computational efficiency [21−23]. The additional zero state allows for sparser weight representations, reducing storage and energy consumption while maintaining performance, particularly in resource-constrained hardware deployments.

Recent advances introduced useful techniques to enhance the performance of SNNs, such as surrogate gradient [24], self-connection [25], tdBN [26], Rate Norm Layer for optimal conversion [27], and stochastic firing [4], which may improve the performance of more flexible quantization schemes and energy-efficient implementations. For SNNs with quantized weights, two main approaches are employed: converting Quantized-ANNs into the corresponding Quantized-SNNs (Q-SNNs) [28,29] and training Q-SNNs directly. In former frameworks, to improve prediction accuracy and reduce the inference latency, subsequent research introduced methods such as the "Early Exit" method [28], and weight-threshold balance conversion [29]. The alternative approach is to directly train Q-SNNs. Pfeiffer and Pfeil [30] explored the feasibility of directly training Q-SNNs and analyzed the impact of quantization on network performance during training. Further research on directly training Q-SNNs has introduced techniques such as ADMM [7], threshold annealing techniques [31], spatio-temporal batch normalization [32], Bayesian methods [33], BS4NN algorithm [34], and adaptive local binarization [35] to reduce storage and computational energy consumption while minimizing accuracy loss.

3 Methodology

Here, we formally propose the ternary quantization methods in Subsection 3.2 and the uncertainty reduction strategy of quantized SNNs with an apposite estimator in Subsection 3.3.

3.1 Training SNNs with surrogate gradients

We start our work with a brief introduction to the feedforward and backpropagation calculations of SNNs. Here, we take the classical Leaky Integrate-and-Fire spiking models as an example, which updates the potential value u_i according to

$$u_i(t) = u_i(t-1) - u_\Delta + \sum_j \boldsymbol{W}_{ij} s_j(t), \tag{1}$$

where $u_i(t)$ denotes the potential value of neuron i at timestamp t, u_Δ is the decaying potential, s_j represents the received spike from neuron j, and $\boldsymbol{W}_{ij}$ represents the connection weight between neurons i and j. The spiking neuron employs the typical threshold rule, that is, neuron i fires spikes s_i at time t if and only if $u_i(t) \geqslant u_{\text{th}}$ where u_{th} indicates the firing threshold. We formulate the spiking firing procedure as

$$s_i(u,t) = \begin{cases} 1, & u_i(t) \geqslant u_{\text{th}}, \\ 0, & u_i(t) < u_{\text{th}}, \end{cases}$$

in which u_{th} denotes the firing threshold. It is observed that spiking dynamics are non-differentiable since the derivative of the spike is

$$\frac{\partial s}{\partial u} = \delta(u - u_{\text{th}}) = \begin{cases} +\infty, & u = u_{\text{th}}, \\ 0, & \text{otherwise}, \end{cases}$$

where $\delta(u)$ is the Dirac delta function. This manner makes it hard to train a supervised SNN using a backpropagation methodology like that in ANNs. Existing solution approaches can be roughly divided into two categories, that is, surrogate-gradient-based [24] and ANN-conversed algorithms [27,36]. Throughout this paper, we focus on the surrogate-gradient-based approach, the key idea of which is to still employ the Heaviside function in the forward phase but replace the derivative of the Heaviside function with alternative differentiable functions in the backpropagation phase.

3.2 Ternary quantization of SNNs

In this work, we focus on the ternary quantization of SNNs that converts connection weights into a ternary format. Formally, we employ $\boldsymbol{T}$ to denote the ternary weights, where each element of $\boldsymbol{T}$ belongs to $\{-1, 0, 1\}$. Provided the input-output pair $(\boldsymbol{x}, \boldsymbol{y})$ and an apposite loss function $\mathcal{L}(\cdot,\cdot)$, we can build the following optimization in the supervised learning paradigm

$$\boldsymbol{T}^* = \arg\min_{\boldsymbol{T}} \mathcal{L}(\boldsymbol{y}, f(\boldsymbol{x}, \boldsymbol{T})), \tag{2}$$

where $f(\boldsymbol{x}, \boldsymbol{T})$ denotes an SNN equipped with ternary weights. In contrast to traditional SNNs that utilize full-precision or floating-point connection weights, the concerned Ternary SNN (T-SNN) requires only 2 bits to store learnable parameters and perform calculations within the network model, thereby significantly compressing the model size.

However, training T-SNNs is intractable, the difficulty of which is twofold. First, the feedforward computations of T-SNNs are stored in integer programming. Thus, a lot of information would be lost once the spiking neuron is activated. Second, the existing SNNs are almost trained based on the spike response model scheme, where the derivative of produced spikes with respect to connection weights is dominated by a composite product of those between produced spikes, potential variables, and connection weights. Thus, the backpropagation computations relative to ternary weights exacerbate discontinuities and non-differentiability. To tackle these challenges, we here extend the key idea of surrogate gradients for SNN training and propose the indirect and direct methods to solve the optimization problem in Eq. (2).

3.2.1 Direct method

An intuitive approach to solving Eq. (2) is to persist in computing surrogate gradients by adding a collection of full-precision weights $\boldsymbol{W}$ as latent variables during training. Here, we can list the weight-updating procedure as follows:

$$\text{Step 1:}\quad \boldsymbol{W}^{t+1} = \boldsymbol{W}^{t} - \eta \frac{\partial \mathcal{L}(\boldsymbol{y}, f(\boldsymbol{x}, \boldsymbol{T}^{t}))}{\partial \boldsymbol{T}^{t}} \frac{\partial \boldsymbol{T}^{t}}{\partial \boldsymbol{W}^{t}}, \tag{3}$$

$$\text{Step 2:}\quad \boldsymbol{T}^{t+1} = g(\boldsymbol{W}^{t+1} \mid \gamma), \tag{4}$$

where η is the learning rate, g denotes the function of ternary quantization, and the superscript t indicates the round of alternating optimization. Here, we employ the ternarization function [37] as follows:

$$\boldsymbol{T}_{ij} = g(\boldsymbol{W}_{ij}|\gamma) = \begin{cases} 1, & \boldsymbol{W}_{ij} \geqslant \gamma, \\ 0, & -\gamma < \boldsymbol{W}_{ij} < \gamma, \\ -1, & \boldsymbol{W}_{ij} \leqslant -\gamma, \end{cases} \tag{5}$$

where the hyper-parameter $\gamma(>0)$ splits the mapping region and $\boldsymbol{W}_{ij}$ and $\boldsymbol{T}_{ij}$ are the (i, j)-elements of matrices $\boldsymbol{W}$ and $\boldsymbol{T}$, respectively.

In Step 1, we extend surrogate gradients to approximate the derivative of ternary weights with respect to full-precision weights. Following the straight-through estimators presented by Bengio et al. [38], we here force the derivative of the ternarization function to 1, that is,

$$\frac{\partial \boldsymbol{T}^{t}}{\partial \boldsymbol{W}^{t}} = \frac{\partial g(\boldsymbol{W}^{t} \mid \gamma)}{\partial \boldsymbol{W}^{t}} = 1. \tag{6}$$

In Step 2, we bind the partition factor γ to the latent matrix $\boldsymbol{W}$ in the ternarization function without any extra optimization, replacing conventional usage in Eq. (9). It is recommended $\gamma = N^{-1}\|\boldsymbol{W}\|_1$, where N denotes the number of all elements of $\boldsymbol{W}$. Formally, we build the following comparison between vanilla gradient descent and ours:

$$\boldsymbol{T}^{*} \leftarrow g\left(\boldsymbol{W}^{*} \leftarrow \frac{\partial \mathcal{L}}{\partial \boldsymbol{W}} \middle| \gamma\right) \quad \text{versus} \quad \boldsymbol{T}^{*} \leftarrow \frac{\partial \mathcal{L}}{\partial \boldsymbol{T}},$$

where we use the symbol $\leftarrow$ to denote the induced assignment. It is observed that the direct method claims that directly optimizes ternary weights by adjusting the full-precision weights correspondingly.

3.2.2 Indirect method

An alternative idea for T-SNN training is to transfer a Full-Precision SNN (FP-SNN) with high performance into the T-SNN. The key to this approach is to establish the relationship between T-SNNs and FP-SNNs so that one can apply mature algorithms to find an FP-SNN and then obtain the T-SNN by narrowing the gap between the well-trained FP-SNN and the corresponding T-SNN. Formally, we employ $\boldsymbol{W}$ to indicate the full-precision weight of the FP-SNN. If the loss function conforms to the triangle inequality, the original optimization objective of Eq. (2) can be relaxed by inserting $\boldsymbol{W}$ as follows:

$$\underbrace{\mathcal{L}(\boldsymbol{y}, f(\boldsymbol{x}, \boldsymbol{T}))}_{①} \leqslant \underbrace{\mathcal{L}(\boldsymbol{y}, f(\boldsymbol{x}, \boldsymbol{W}))}_{②} + \underbrace{\mathcal{L}(f(\boldsymbol{x}, \boldsymbol{W}), f(\boldsymbol{x}, \boldsymbol{T}))}_{③}, \tag{7}$$

where ① and ② separately describe the training losses led by the ternary weight $\boldsymbol{T}$ and the full-precision one $\boldsymbol{W}$, and ③ is the gap between outputs determined by $\boldsymbol{W}$ and $\boldsymbol{T}$. In contrast to directly solving Eq. (2), this work attempts to minimize the sum of ② and ③. Formally, we have the following iterative optimization scheme:

$$\text{Step 1:}\quad \boldsymbol{W}^{t+1} = \arg\min_{\boldsymbol{W}} \mathcal{L}(\boldsymbol{y}, f(\boldsymbol{x}, \boldsymbol{W} \mid \boldsymbol{T}^{t})), \tag{8}$$

$$\text{Step 2:}\quad \boldsymbol{T}^{t+1} = \arg\min_{\boldsymbol{T}} \mathcal{L}\left(f(\boldsymbol{x}, \boldsymbol{W}^{t+1}),\ f(\boldsymbol{x}, \boldsymbol{T})\right). \tag{9}$$

In this scheme, Step 1 retrains the full-precision weights $\boldsymbol{W}^{t+1}$ by minimizing ②, provided $\boldsymbol{T}^{t}$. In practice, we implement this optimization by adding the ternarization-aware distillation [39] to avoid widening the precision degradation led by weight quantization

$$\boldsymbol{W}^{t+1} = \arg\min_{\boldsymbol{W}} \mathcal{L}(\boldsymbol{y}, f(\boldsymbol{x}, \boldsymbol{W} \mid \boldsymbol{T}^{t})) + \mathcal{L}_{\text{distill}}(\boldsymbol{W}, \boldsymbol{x} \mid \boldsymbol{T}^{t}). \tag{10}$$

The optimization sub-problem in Eq. (10) can be solved by some mature algorithms, such as SGD [40] and Adam [41]. Step 2 searches for ternary weights by minimizing ③, provided $\boldsymbol{W}^{t+1}$. This optimization problem in Eq. (9) can be solved well by zero-order search algorithms. Reusing the ternarization function in Eq. (5), solving Eq. (9) is equivalent to finding the best γ for minimizing the error gap between the two investigated SNNs. By repeating Step 1 and Step 2 iteratively, we can obtain a collection of ternary weights with relatively high performance. Figure 1 plots the workflow of the indirect method with ternarization-aware distillation.

It is evident that the proposed indirect method can converge to the optimal solution. Following the conventional line, it is sufficient to prove the convergence of each step for an iterative optimization method. The convergence of Step 1 is attributed to that of SNN optimizers, while the grid search enables the convergence of Step 2. Additionally, we acknowledge that the proposed indirect method may be hindered by the issue of slow convergence speed, as it necessitates training an FP-SNN at each iteration until convergence is achieved. Thus, this manner would inevitably consume a lot of computing time and energy in the training phase.

Figure 2 illustrates the workflows and differences between the proposed indirect and direct methods. It is observed that the indirect approach involves at least two optimizations, whereas the direct approach can optimize Eq. (2) directly using surrogate gradients without requiring any scaling factor. Therefore, our proposed direct method can accelerate the training process.

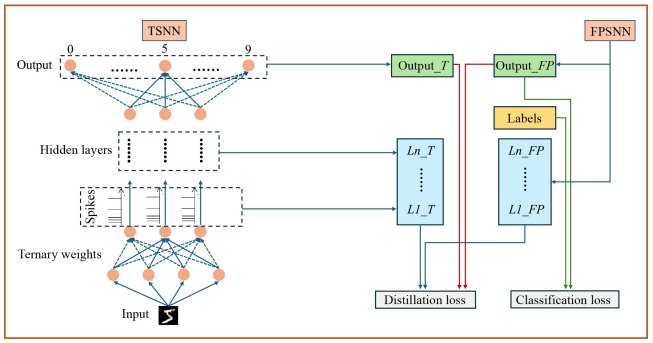

Fig. 1 Workflow of the indirect method using ternarization-aware distillation

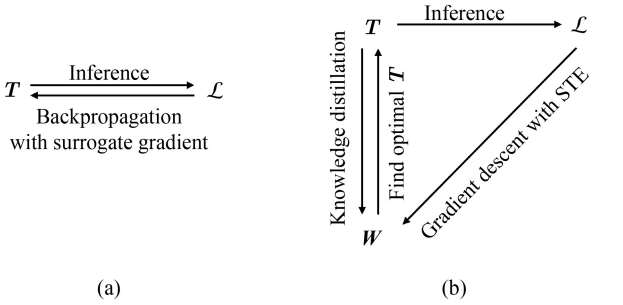

Fig. 2 Illustrations of our proposed methods. (a) Direct; (b) indirect

3.3 Uncertainty quantification and reduction

This subsection focuses on whether weight quantization increases the uncertainty of SNNs. It is obvious that the ternary weight of T-SNNs obeys a ternary distribution, that is, $t \sim \mathcal{T}(p, 1-p-q, q)$, in which $P(t=-1)=p$, $P(t=0)=1-p-q$, and $P(t=1)=q$ with $p,q \in [0,1]$. Thus, the output $f(\boldsymbol{x},\boldsymbol{T})$ of a T-SNN can be regarded as a random variable, and further the covariance $\boldsymbol{\Sigma}_{f(\boldsymbol{x},\boldsymbol{T})}$ of output can measure the uncertainty of T-SNNs [42,43].

Based on this recognition, we can implement the empirical uncertainty of T-SNNs by sampling ternary weights from $\mathcal{T}(p,1-p-q,q)$ in multiple trials and calculating the mean value of the empirical covariances of the network outputs. Equivalently, the quadratic form $\frac{1}{N}\mathbf{1}^{\mathrm{T}}\boldsymbol{\Sigma}_{f(\boldsymbol{x},\boldsymbol{T})}\mathbf{1}$ can be regarded as the uncertainty measure of T-SNNs, where N denotes the number of all elements of $\boldsymbol{\Sigma}_{f(\boldsymbol{x},\boldsymbol{T})}$. A higher value of this quadratic form reflects greater network uncertainty, whereas a lower value corresponds to reduced uncertainty.

Theorem 1. For the layer-wise propagation, the uncertainty measure $\frac{1}{N}\mathbf{1}^{\mathrm{T}}\boldsymbol{\Sigma}_{f(\boldsymbol{x},\boldsymbol{T})}\mathbf{1}$ of T-SNNs is monotonically positively correlated with $|p-q|$ and $|p+q-(p-q)^2|$. Particularly, if $q-p=0$ and $p+q-(p-q)^2=0$, then it holds for the covariance of the output $\boldsymbol{\Sigma}_{f(\boldsymbol{x},\boldsymbol{T})}=\mathbf{0}$.

Proof Sketch The pre-activation covariance is constituted by a linear combination of $\mathbb{E}(\boldsymbol{x}\boldsymbol{T}\boldsymbol{T}^{\mathrm{T}}\boldsymbol{x}^{\mathrm{T}})$ and $\mathbb{E}(\boldsymbol{x})\mathbb{E}(\boldsymbol{T})\mathbb{E}(\boldsymbol{T}^{\mathrm{T}})\mathbb{E}(\boldsymbol{x}^{\mathrm{T}})$, where $\mathbb{E}(\cdot)$ denotes the expectation. According to the law of total expectation, the concerned measure can be reformulated as a combination of $\frac{1}{N}\mathbf{1}^{\mathrm{T}}\mathbb{E}(\boldsymbol{x}\boldsymbol{x}^{\mathrm{T}})\mathbf{1}\cdot(p+q-(p-q)^2)$ and $\frac{1}{N}\mathbf{1}^{\mathrm{T}}\boldsymbol{\Sigma}_{\boldsymbol{x}}\mathbf{1}\cdot(p-q)^2$, which is monotonically positively correlated with the two terms. □

Theorem 1 reveals a close relation between the uncertainty of T-SNNs and two functions $q-p$ and $p+q-(p-q)^2$; there exists an optimal solution that adjusting these two functions can enable the uncertainty of T-SNNs reducing to zero. Inspired by this recognition, we present the optimization for uncertainty reduction as follows:

$$\gamma^* = \arg\min_{\gamma} |g_3(\gamma,\boldsymbol{W})| + \lambda |g_4(\gamma,\boldsymbol{W})|, \tag{11}$$

where γ^* is the optimal splitting threshold, $\lambda \in \mathbb{R}^+$ denotes the balance factor, $\boldsymbol{W}$ here denotes the full-precision weight, and

$$\begin{cases} g_1(\gamma,\boldsymbol{W}) = \sum_{i,j}\chi_{(-\infty,-\gamma)}(\boldsymbol{W}_{ij})/N, \\ g_2(\gamma,\boldsymbol{W}) = \sum_{i,j}\chi_{(\gamma,+\infty)}(\boldsymbol{W}_{ij})/N, \\ g_3(\gamma,\boldsymbol{W}) = g_2(\gamma,\boldsymbol{W}) - g_1(\gamma,\boldsymbol{W}), \\ g_4(\gamma,\boldsymbol{W}) = g_1(\gamma,\boldsymbol{W}) + g_2(\gamma,\boldsymbol{W}) - (g_3(\gamma,\boldsymbol{W}))^2, \end{cases}$$

in which g_1 and g_2 separately are estimators for p and q, $\boldsymbol{W}_{ij}$ denotes the (i,j)-th element of $\boldsymbol{W}$, N denotes the number of all elements of $\boldsymbol{W}$, and $\chi_A(x)$ is an indicator function that assigns a value as 1 if $x \in \mathrm{A}$ while takes 0 if $x \notin \mathrm{A}$. Notice that Eq. (11) finds the apposite ternary weights by converting the full-precision ones with the splitting threshold, where the full-precision ones can be computed by the direct and indirect methods.

4 Experiments

In this section, we conduct experiments to demonstrate the effectiveness of our proposed methods in terms of accuracy, inference efficiency, energy consumption, memory storage, and uncertainty.

4.1 Configurations

The conducted datasets comprise the MNIST [44], CIFAR-10 [45], and Fashion-MNIST datasets.

Here, we combine our proposed indirect and direct methods with various SNN models, including SNN-Slayer [46], SNN-PLIF [47], SNN-Dropconnect [48], and SNN-Dropout [48]. The SNN-Dropconnect and SNN-Dropout algorithms train SNNs by randomly dropping synaptic connection weights during SNN training and randomly dropping the states of neural units at each time instance, respectively. The SNN-PLIF algorithm introduces learnable membrane time constants to enhance the expressive capacity of SNNs. The SNN-Slayer algorithm achieves simultaneous learning of synaptic weights and axonal delays by defining specific loss functions, considering time-dependent error allocation, and employing a unique derivative approximation method for spike functions. Table 1 lists the runtime of various SNNs tested in our experiments, where Runtime (h) indicates the total training time in hours, ranging from 0.18 h (fastest) to 5 h (slowest), and is obtained by averaging over $n=5$ repeated training runs. All models were trained on NVIDIA RTX 4090 GPUs (single device, 25.2 GB memory) and implemented by SpikingJelly [49]. For each model, four independent experiments are conducted, and the corresponding confidence intervals are reported in Table 2.

Table 1 Runtime of various tested SNN models

Model	Runtime/h
SNN-Dropconnect-MNISTNet	0.31
SNN-PLIF-MNISTNet	0.18
SNN-Dropconnect-CIFARNet	0.63
SNN-PLIF-CIFARNet	1.10
SNN-Dropout-MNISTNet	0.31
SNN-Slayer-MNISTNet	5.00
SNN-Dropout-CIFARNet	0.76

Table 2 Performance of FP-SNNs and T-SNNs on MNIST, CIFAR10, and FashionMNIST datasets

Dataset	Algorithm	Configuration	Quantization part	Quantization training	Accuracy/%	Sparsity/%	Uncertainty	Storage/MB	Energy/pJ	FLOPs
MNIST	SNN-Dropconnect	FC	FC	None	98.25	0.00	0.062	3.5MB	3.10×10^{10} pJ	1.79M
			FC_q^1	Indirect method	$97.68\pm0.11_2$	53.77 ± 0.07	0.052	0.22MB	$9.20\times10^{8}\pm6.44\times10^{5}$ pJ	0.41 ± 0.0003M
				Direct method	97.43 ± 0.36	52.21 ± 0.50	0.051		$9.51\times10^{8}\pm4.76\times10^{6}$ pJ	0.43 ± 0.0022M
	SNN-Dropout	FC	FC	None	96.87	0.00	0.063	3.5MB	3.10×10^{10} pJ	1.76M
			FC_q	Indirect method	93.18 ± 0.56	53.87 ± 2.43	0.052	0.22MB	$9.18\times10^{8}\pm0.22\times10^{8}$ pJ	0.44 ± 0.01M
				Direct method	93.00 ± 0.96	53.96 ± 2.34	0.053		$9.16\times10^{8}\pm0.21\times10^{8}$ pJ	0.43 ± 0.01M
	SNN-Slayer	FC	FC	None	95.37	0.00	0.064	4.6MB	7.62×10^{10} pJ	2.38M
			FC_q	Indirect method	95.00 ± 0.74	51.78 ± 1.22	0.056	0.29MB	$2.36\times10^{9}\pm2.88\times10^{7}$ pJ	0.55 ± 0.007M
				Direct method	93.08 ± 0.83	50.35 ± 0.78	0.054		$2.43\times10^{9}\pm1.90\times10^{7}$ pJ	0.56 ± 0.004M
	SNN-PLIF	2Conv + 2FC	2Conv + 2FC	None	99.00	0.00	0.039	51MB	8.47×10^{11} pJ	85.70M
			$2Conv_q+2FC$	Indirect method	98.94 ± 0.14	0.67 ± 0.02	0.004	50MB	$2.93\times10^{11}\pm5.86\times10^{7}$ pJ	55.52 ± 0.011M
				Direct method	99.52 ± 0.09	0.65 ± 0.03	0.004		$2.93\times10^{11}\pm8.79\times10^{7}$ pJ	55.54 ± 0.017M
			$2Conv+2FC_q$	Indirect method	99.52 ± 0.21	61.00 ± 1.23	0.012	3.7MB	$2.36\times10^{11}\pm2.90\times10^{9}$ pJ	29.92 ± 0.36M
				Direct method	99.42 ± 0.09	58.99 ± 0.03	0.012		$2.48\times10^{11}\pm7.44\times10^{7}$ pJ	29.81 ± 0.009M
			$2Conv_q+2FC_q$	Indirect method	98.52 ± 0.28	62.48 ± 2.01	0.015	3.3MB	$1.99\times10^{10}\pm4.00\times10^{8}$ pJ	16.94 ± 0.34M
				Direct method	98.43 ± 0.43	60.00 ± 0.27	0.016		$2.12\times10^{10}\pm5.72\times10^{7}$ pJ	17.04 ± 0.05M
CIFAR10	SNN-Dropconnect	2Conv + 4FC	2Conv + 4FC	None	57.50	0.00	0.061	4.4MB	7.46×10^{10} pJ	9.73M
			$2Conv_q^1+4FC$	Indirect method	65.00 ± 0.56	1.33 ± 0.00	0.043	4.35MB	$2.08\times10^{10}\pm0.00\times10^{10}$ pJ	5.87 ± 0.00M
				Direct method	66.02 ± 0.25	1.25 ± 0.05	0.052		$2.08\times10^{10}\pm1.04\times10^{7}$ pJ	5.78 ± 0.003M
			$2Conv+4FC_q$	Indirect method	65.71 ± 0.21	53.50 ± 0.01	0.054	0.38MB	$2.71\times10^{10}\pm2.71\times10^{6}$ pJ	3.95 ± 0.0004M
				Direct method	65.38 ± 0.54	53.46 ± 0.01	0.056		$2.71\times10^{10}\pm2.71\times10^{6}$ pJ	3.95 ± 0.0004M
			$2Conv_q+4FC_q$	Indirect method	64.13 ± 1.07	54.81 ± 0.03	0.053	0.28MB	$2.19\times10^{9}\pm6.56\times10^{5}$ pJ	2.17 ± 0.0007M
				Direct method	61.08 ± 3.40	54.18 ± 0.07	0.051		$2.19\times10^{9}\pm1.53\times10^{6}$ pJ	2.17 ± 0.002M
	SNN-Dropout	2Conv + 2FC	2Conv + 2FC	None	65.30	0.00	0.067	2.6MB	4.26×10^{10} pJ	8.61M
			$2Conv_q+2FC$	Indirect method	66.43 ± 0.70	2.32 ± 0.02	0.064	2.45MB	$8.31\times10^{9}\pm1.66\times10^{6}$ pJ	4.82 ± 0.001M
				Direct method	66.10 ± 0.77	2.33 ± 0.00	0.065		$8.31\times10^{9}\pm0.00\times10^{9}$ pJ	4.82 ± 0.00M
			$2Conv+2FC_q$	Indirect method	66.35 ± 0.33	52.94 ± 0.30	0.036	0.26MB	$1.73\times10^{10}\pm5.20\times10^{7}$ pJ	3.73 ± 0.01M
				Direct method	65.40 ± 0.30	52.94 ± 0.28	0.037		$1.73\times10^{10}\pm4.85\times10^{7}$ pJ	3.73 ± 0.01M
			$2Conv_q+2FC_q$	Indirect method	64.83 ± 1.56	55.37 ± 0.40	0.033	0.16MB	$1.27\times10^{9}\pm5.07\times10^{6}$ pJ	1.91 ± 0.008M
				Direct method	63.59 ± 1.48	55.29 ± 2.65	0.038		$1.27\times10^{9}\pm3.36\times10^{7}$ pJ	1.91 ± 0.05M
	SNN-PLIF	6Conv + 2FC	6Conv + 2FC	None	93.16	0.00	0.062	141MB	2.35×10^{12} pJ	3.4G
			$6Conv_q+2FC$	Indirect method	90.55 ± 1.96	4.78 ± 0.01	0.003	140MB	$1.82\times10^{11}\pm1.82\times10^{7}$ pJ	1.65 ± 0.0002G
				Direct method	90.30 ± 1.47	4.72 ± 0.06	0.025		$1.82\times10^{11}\pm1.09\times10^{8}$ pJ	1.65 ± 0.001G
			$6Conv+2FC_q$	Indirect method	89.99 ± 4.95	58.22 ± 2.05	0.012	20MB	$9.65\times10^{11}\pm1.98\times10^{10}$ pJ	1.47 ± 0.03G
				Direct method	89.26 ± 3.48	55.11 ± 1.03	0.042		$1.04\times10^{12}\pm1.07\times10^{10}$ pJ	1.55 ± 0.02G
			$6Conv_q+2FC_q$	Indirect method	89.17 ± 3.12	60.28 ± 0.68	0.002	8.9MB	$5.84\times10^{10}\pm4.03\times10^{8}$ pJ	663.85 ± 4.58M
				Direct method	88.78 ± 4.35	59.52 ± 0.75	0.042		$5.95\times10^{10}\pm4.46\times10^{8}$ pJ	700.91 ± 5.26M

Table 2 (Continued)

Dataset	Algorithm	Configuration	Quantization part	Quantization training	Accuracy/%	Sparsity/%	Uncertainty	Storage/MB	Energy/pJ	FLOPs
FashionMNIST	SNN-PLIF	2Conv + 2FC	2Conv + 2FC	None	94.28	0.00	0.008	51MB	8.47×10^{11} pJ	85.70M
			$2Conv_q^1 + 2FC$	Indirect method	94.05 ± 0.37	0.72 ± 0.06	0.003	50MB	$2.93\times10^{11}\pm1.76\times10^{8}$ pJ	55.53 ± 0.03M
				Direct method	93.87 ± 0.32	0.68 ± 0.01	0.005		$2.93\times10^{11}\pm2.93\times19^{7}$ pJ	55.52 ± 0.006M
			$2Conv + 2FC_q$	Indirect method	93.27 ± 0.41	62.54 ± 1.26	0.008	3.7MB	$2.27\times10^{11}\pm2.86\times10^{9}$ pJ	28.13 ± 0.35M
				Direct method	93.23 ± 0.06	59.14 ± 0.41	0.003		$2.47\times10^{11}\pm1.01\times10^{9}$ pJ	29.99 ± 0.12M
			$2Conv_q + 2FC_q$	Indirect method	92.81 ± 0.05	63.38 ± 1.08	0.007	3.3MB	$1.94\times10^{10}\pm2.10\times10^{8}$ pJ	16.16 ± 0.18M
				Direct method	$92.21\pm0.06\%$	59.79 ± 0.41	0.004		$2.13\times10^{10}\pm8.74\times10^{7}$ pJ	17.40 ± 0.07M

[1] The subscript q indicates that this part is ternary-quantized.

[2] The values are presented as mean ± standard deviation, calculated over four independent trials.

The SNN-Dropconnect-MNISTNet, SNN-Dropout-MNISTNet, and SNN-Slayer-MNISTNet all correspond to architectures with only fully connected layers, while the SNN-PLIF-MNISTNet corresponds to a model architecture with two convolutional layers and two fully connected layers, which is similar to VGG, using 3×3 convolutional kernels and an alternating stacking design of convolutional and pooling layers. The SNN-Dropconnect-CIFARNet uses an architecture with two convolutional layers and four fully connected layers, utilizing 5×5 convolutional kernels. SNN-Dropout-CIFARNet adopts a combination of two convolutional layers and two fully connected layers, with 5×5 convolutional kernels. SNN-PLIF-CIFARNet uses six convolutional layers and two fully connected layers, with an architecture similar to VGG, employing 3×3 convolutional kernels and 2×2 max-pooling layers.

4.2 About accuracy

The first experiment is to compare the accuracy of T-SNNs and FP-SNNs across different datasets. The experimental results are shown in Table 2. It is observed that the indirect method generally achieves better accuracy than the direct method across all conducted datasets and tested SNNs. Besides, the precision degradation led by weight quantization is tolerable, where it locates [−0.52%, 1.3%] on the MNIST dataset, [−8.51%,2.61%] on the CIFAR-10 dataset, and no more than 0.23% in the Fashion-MNIST dataset. Notice that the SNN accuracy has even improved after the ternary lightweight. These observations confirm that the T-SNNs obtained by both indirect and direct methods maintain comparative performance without excessive precision degradation compared with FP-SNNs.

To further investigate the effectiveness of the proposed methods, we compare the accuracy of T-SNNs with that of other lightweight SNNs on the MNIST and CIFAR-10 datasets. The contenders on the MNIST dataset include the Single-spike Temporal Coding [50], Perceptron-Inception [51] that adopts an architecture of convolutional layers and two fully connected layers, MuST[2] representation [52], Address Event Representation [53], and Spatiotemporally Compressive Spike Feature [54]. The contenders on the CIFAR-10 dataset contain ANN2SNN-CNN [36] that are applied to an architecture with six convolutional layers and three fully connected layers, ANN2SNN-VGG [55] equipped with the VGG-9 architecture, and Direct Training [7] that adopts an architecture with seven convolutional layers and three fully connected layers. The experimental results are shown in Table 3. Our proposed direct method surpasses the contenders on both the MNIST and CIFAR-10 datasets, achieving the accuracy of 98% and 89.43%, respectively.

4.3 About inference efficiency

The second experiment is to investigate the inference efficiency of our proposed quantized methods, which comprises the computational efficiency and energy consumption. Firstly, we count the number of Floating Point Operations (FLOPs) to measure computational complexity during the real inference process. The results are shown in Table 2. It is observed that the FLOPs ratio between FP-SNNs and T-SNNs is close to 2, which means the ternary quantization methods can reduce the FLOPs number by 50%, implying the effectiveness of the ternary methods in terms of inference acceleration.

Secondly, we calculate the energy consumption of each model in the inference process and compare their energy consumption ratio before and after the ternarization. Here, we follow the experimental line of Yan et al. [56] for measuring the energy consumption and list the experimental results in Table 2. It is observed that our proposed methods achieve a significant reduction, around 93.58%, in energy consumption, and the average value of the energy consumption ratio of the FP-SNN and T-SNN is 15.57. Besides, the total energy consumption of addition operations in CPUs is approximately proportional to the bit width of the numbers. Since all operands are ternarized to 2 bits during the inference process of T-SNNs, the theoretical optimal ratio between FP-SNNs and T-SNNs is 16, which is very close to the empirical ratio value of 15.57, confirming that our ternary quantization methods can effectively reduce the energy consumption of the SNN inference.

4.4 About memory storage

The third experiment investigates the memory storage occupied by FP-SNNs and T-SNNs. The experimental results are presented in Table 2. It shows that our proposed methods achieve a significant reduction, around 93.64%, in memory storage, and the average memory storage ratio between FP-SNNs and T-SNNs is 15.57.

Table 3 Classification accuracy of our proposed method and contenders on MNIST and CIFAR-10 datasets

MNIST			CIFAR-10		
Lightweight methods	Bit width	Accuracy/%	Lightweight methods	Bit width	Accuracy/%
Single-spike temporal coding [50]	32	78.00	ANN2SNN-CNN [36]	1	88.27
Perceptron-inception [51]	32	88.00	ANN2SNN-VGG [55]	1	88.25
MuST[2] representation [52]	32	76.90	Direct training [7]	4	89.40
Address event representation [53]	32	91.29		3	89.32
Spatiotemporally compressive spike feature [54]	32	91.22		2	89.23
BS4NN [34]	1	97.00		1	89.01
ADMM[3] [7]	3	97.25	Weight-threshold balance [29]	1	82.73
BAYES-BISNN [33]	1	95.10	Adaptive local binarization [35]	1	86.43
ST-BISNN [33]	1	96.00	ADMM [7]	3	87.84
Direct method (this work)	2	98.43	Spatio-temporal batch normalization [32]	1	62.10
Indirect method (this work)	2	**98.52%**[4]	Direct method (this work)	2	88.78
			Indirect method (this work)	2	**89.17%**

[1]xConv: No mentions of the number of Conv layers.
[2]MuST: Multiscale spatio-temporal feature.
[3]ADMM: Alternating direction method of multipliers.
[4]Bold font indicates the best accuracy performance among the methods.

Ideally, the optimal ratio of storage between FP-SNNs and T-SNNs should be 16, which is very close to our experimental ratio. This observation indicates the effectiveness of our methods in reducing the memory storage of SNNs.

4.5 About uncertainty

In this experiment, we analyze the uncertainty of FP-SNNs and T-SNNs. Based on the settings mentioned in Subsection 3.3, we quantify the uncertainty of SNNs by adding noise to the model weights and leveraging the variance of SNN outputs along with noisy weights. Here, the injected weight noise is sampled from a zero-mean Gaussian distribution with a variance determined by the specified noise level, and the noise applied to different layers is independently sampled from the same Gaussian distribution. Figures 3(a)−3(c) display the uncertainty ratio curves of output variance with respect to the parameter noise variance. There are two observations. Firstly, the uncertainty ratio between FP-SNNs and T-SNNs with the direct method is significantly greater than 1, especially when the weight variance exceeds 0.2, and commonly this ratio shows an increasing trend. Secondly, as the noise increases, T-SNNs, especially those obtained by the direct method, exhibit higher robustness. In detail, the output of the T-SNNs is less affected and has higher certainty under the influence of noise. Table 2 further lists the empirical results relative to the uncertainty quantification of the conducted models, of which the uncertainty value refers to the point where the ratio of uncertainties between FP-SNN and T-SNN is the largest. It is observed that T-SNNs obtained by the direct method have the smallest uncertainty, while those obtained by the indirect method do not always have the smallest uncertainty.

Next, we conduct empirical investigations on whether and to what extent our proposed regularization method contributes to uncertainty reduction. We equip the regularization method to T-SNNs trained with the direct method and compare the uncertainty of the networks without and with regularization. The experimental results in Table 4 show the uncertainty value at the max ratio points, which refer to the point corresponding to the parameter noise variance and the associated uncertainty value at which the ratio of the uncertainty of the T-SNNs without regularization to that of the T-SNN with regularization is maximized. The results show that the uncertainties of T-SNNs with regularization decreased by 19.4% on average compared to those without regularization. This indicates the effectiveness of our regularization method in reducing the uncertainty of T-SNNs. Figures 3(d)−3(f) further display the uncertainty curves of output variance with respect to the parameter noise variance, which shows that the uncertainties of T-SNNs with the regularization method are always lower than those of T-SNNs without the regularization method. In comparison to Figs. 3(a)−3(c), the ratio curves of Figs. 3(d)−3(f) are basically above the threshold value of 1, which points out the significant reduction of output variance. These observations confirm the effectiveness of our proposed regularizer, where aligning the first and second moments of T-SNNs reduces uncertainty.

4.6 Ablation study

Here, we design ablation experiments to confirm the effect of adding ternarization-aware distillation in Eq. (10). We follow the configurations from the aforementioned experiments and employ the indirect method without ternarization-aware distillation as the

contender. Table 5 displays the numerical results, from which the indirect method equipped with ternarization-aware distillation performs markedly better than that without ternarization-aware distillation.

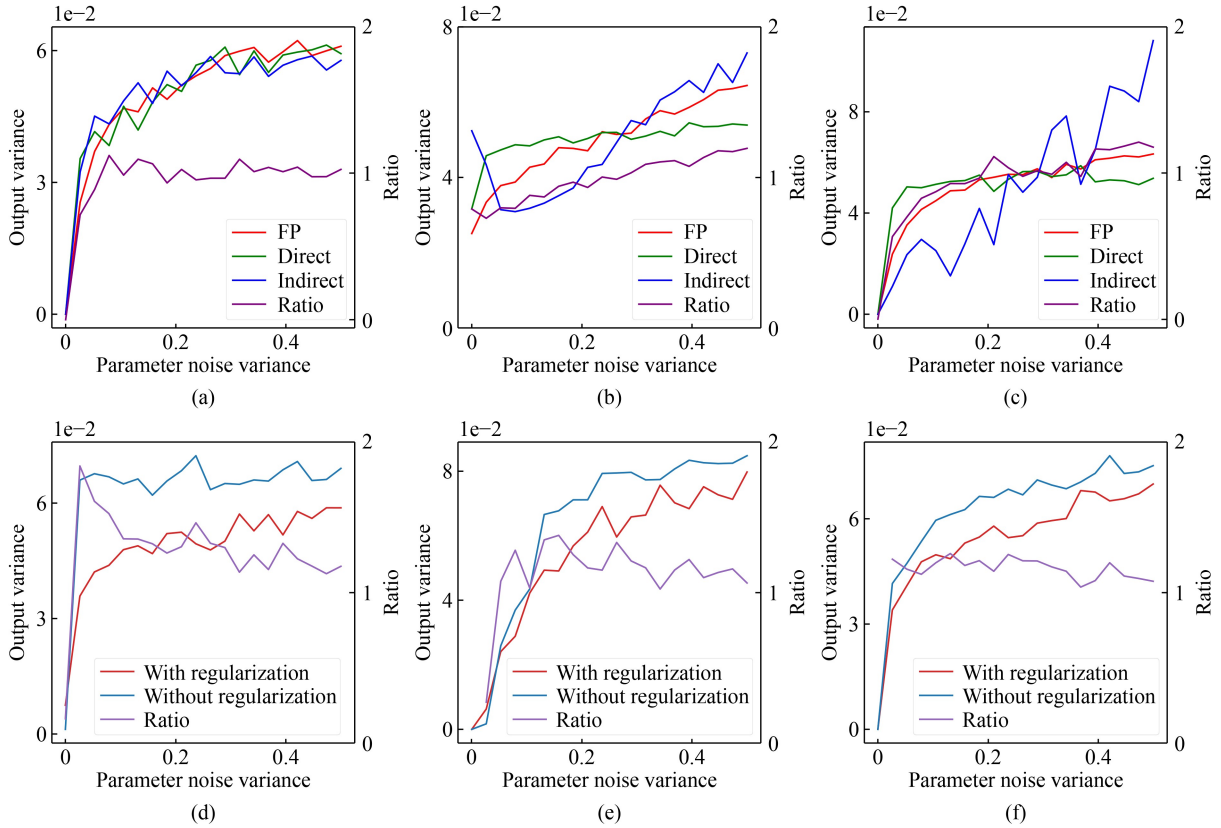

Fig. 3 Uncertainty and ratio of conducted SNNs vs. the corresponding quantized models, including (a) the FP SNN-Dropout-CIFARNet vs. Quantized SNN-Dropout-CIFARNet; (b) the FP SNN-Slayer-MNISTNet vs. Quantized SNN-Slayer-MNISTNet; (c) the FP SNN-Dropconnect-MNISTNet vs. Quantized SNN-Dropconnect-MNISTNet. Uncertainty and ratio of conducted SNNs with and without regularization, including (d) SNN-Dropout-CIFARNet; (e) SNN-Slayer-MNISTNet; (f) SNN-Dropconnect-MNISTNet

Table 4 Uncertainty comparison of T-SNNs with and without regularization

Model	Uncertainty without regularization	Uncertainty with regularization
SNN-Dropconnect-MNISTNet [48]	0.051	0.045
SNN-Dropout-CIFARNet [48]	0.038	0.033
SNN-Slayer-MNISTNet [46]	0.054	0.036

Table 5 Performance comparison of the proposed indirect methods with and without ternarization-aware distillation

Dataset	Model	Accuracy without ternarization-aware distillation/%	Accuracy with ternarization-aware distillation/%
MNIST	SNN-Dropconnect [48]	76.83	97.50
	SNN-Dropout [48]	86.93	93.73
	SNN-PLIF [47]	53.37	98.52
	SNN-Slayer [46]	69.03	95.00
CIFAR-10	SNN-Dropconnect [48]	14.48	64.27
	SNN-Dropout [48]	37.80	61.72
	SNN-PLIF [47]	38.89	89.17

5 Conclusions

This paper proposed the ternary implementation of SNNs via two quantization training algorithms that apply ternary quantization to synaptic weights and a regularizer for enhancing both accuracy improvement and uncertainty reduction of T-SNNs. Our method is compatible with various SNN architectures and conventional SNN training algorithms relative to surrogate gradients. Experimental evaluations conducted on multiple datasets showed that our method surpasses its contenders in terms of accuracy, inference efficiency, energy consumption, memory storage, and uncertainty. Our work not only highlights the potential of ternary quantization to enable the widespread deployment of lightweight SNNs on low-power edge devices, but also advances the progress of efficient intelligent computing.

Acknowledgements

This research was supported by the Jiangsu Provincial Natural Science Foundation Youth Project (BK20230782) and Nanjing University-China Mobile Communications Group Co., Ltd. Joint Institute.

Competing interests

The authors declare that they have no competing interests or financial conflicts to disclose.

References

[1] Furber S B, Galluppi F, Temple S, Plana L A. The spinnaker project. Proceedings of the IEEE, 2014, 102(5): 652−665

[2] Merolla P A, Arthur J V, Alvarez-Icaza R, Cassidy A S, Sawada J, Akopyan F, Jackson B L, Imam N, Guo C, Nakamura Y, Brezzo B, Vo I, Esser S K, Appuswamy R, Taba B, Amir A, Flickner M D, Risk W P, Manohar R, Modha D S. A million spiking-neuron integrated circuit with a scalable communication network and interface. Science, 2014, 345(6197): 668−673

[3] 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, Wang H. Loihi: a neuromorphic manycore processor with on-chip learning. IEEE Micro, 2018, 38(1): 82−99

[4] Zhang S Q, Chen J Y, Wu J H, Zhang G, Xiong H, Gu B, Zhou Z H. On the intrinsic structures of spiking neural networks. Journal of Machine Learning Research, 2024, 25(194): 1−74

[5] Samanta A, Hatai I, Mal A K. A survey on hardware accelerator design of deep learning for edge devices. Wireless Personal Communications, 2024, 137(3): 1715−1760

[6] Ding J, Zhang J, Huang T, Liu J K, Yu Z. Assisting training of deep spiking neural networks with parameter initialization. IEEE Transactions on Neural Networks and Learning Systems, 2025, 36(8): 15015−15028

[7] Deng L, Wu Y, Hu Y, Liang L, Li G, Hu X, Ding Y, Li P, Xie Y. Comprehensive SNN compression using ADMM optimization and activity regularization. IEEE Transactions on Neural Networks and Learning Systems, 2023, 34(6): 2791−2805

[8] Hu Y, Zheng Q, Jiang X, Pan G. Fast-SNN: fast spiking neural network by converting quantized ANN. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2023, 45(12): 14546−14562

[9] Wei W, Liang Y, Belatreche A, Xiao Y, Cao H, Ren Z, Wang G, Zhang M, Yang Y. Q-SNNs: quantized spiking neural networks. In: Proceedings of the 32nd ACM International Conference on Multimedia. 2024, 8441−8450

[10] Lin D D, Talathi S S, Annapureddy V S. Fixed point quantization of deep convolutional networks. In: Proceedings of the 33rd International Conference on Machine Learning. 2016, 2849−2858

[11] Liu Z, Wang Y, Han K, Zhang W, Ma S, Gao W. Post-training quantization for vision transformer. In: Proceedings of the 35th International Conference on Neural Information Processing Systems. 2021, 2152

[12] Han S, Mao H, Dally W J. Deep compression: compressing deep neural network with pruning, trained quantization and Huffman coding. In: Proceedings of the 4th International Conference on Learning Representations. 2016

[13] Courbariaux M, Bengio Y, David J P. BinaryConnect: training deep neural networks with binary weights during propagations. In: Proceedings of the 29th International Conference on Neural Information Processing Systems. 2015, 3123−3131

[14] Courbariaux M, Hubara I, Soudry D, El-Yaniv R, Bengio Y. Binarized neural networks: training deep neural networks with weights and activations constrained to +1 or -1. 2016, arXiv preprint arXiv: 1602.02830

[15] Esser S K, McKinstry J L, Bablani D, Appuswamy R, Modha D S. Learned step size quantization. 2019, arXiv preprint arXiv: 1902.08153

[16] Krishnamoorthi R. Quantizing deep convolutional networks for efficient inference: a whitepaper. 2018, arXiv: 1806.08342

[17] Rastegari M, Ordonez V, Redmon J, Farhadi A. XNOR-Net: ImageNet classification using binary convolutional neural networks. In: Proceedings of the 14th European Conference on Computer Vision. 2016, 525−542

[18] Hubara I, Courbariaux M, Soudry D, El-Yaniv R, Bengio Y. Quantized neural networks: training neural networks with low precision weights and activations. Journal of Machine Learning Research, 2018, 18(187): 1−30

[19] Jacob B, Kligys S, Chen B, Zhu M, Tang M, Howard A, Adam H, Kalenichenko D. Quantization and training of neural networks for efficient integer-arithmetic-only inference. In: Proceedings of 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2018, 2704−2713

[20] Wang K, Liu Z J, Lin Y J, Lin J, Han S. HAQ: hardware-aware automated quantization with mixed precision. In: Proceedings of 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2019, 8604−8612

[21] Zhu C, Han S, Mao H, Dally W J. Trained ternary quantization. In: Proceedings of the 5th International Conference on Learning Representations. 2017

[22] Li Y, Ding W, Liu C, Zhang B, Guo G. TRQ: ternary neural networks with residual quantization. In: Proceedings of the 35th AAAI Conference on Artificial Intelligence. 2021, 8538−8546

[23] Vindas Y, Roux E, Guépié B K, Almar M, Delachartre P. An asymmetric heuristic for trained ternary quantization based on the statistics of the weights: an application to medical signal classification.

Pattern Recognition Letters, 2025, 188: 37−45
[24] Neftci E O, Mostafa H, Zenke F. Surrogate gradient learning in spiking neural networks: bringing the power of gradient-based optimization to spiking neural networks. IEEE Signal Processing Magazine, 2019, 36(6): 51−63
[25] Zhang S Q, Zhang Z Y, Zhou Z H. Bifurcation spiking neural network. Journal of Machine Learning Research, 2021, 22(1): 253
[26] Zheng H, Wu Y, Deng L, Hu Y, Li G. Going deeper with directly-trained larger spiking neural networks. In: Proceedings of the 35th AAAI Conference on Artificial Intelligence. 2021, 11062−11070
[27] Ding J, Yu Z, Tian Y, Huang T. Optimal ANN-SNN conversion for fast and accurate inference in deep spiking neural networks. In: Proceedings of the 30th International Joint Conference on Artificial Intelligence. 2021, 2328−2336
[28] Lu S, Sengupta A. Exploring the connection between binary and spiking neural networks. Frontiers in Neuroscience, 2020, 14: 535
[29] Wang Y, Xu Y, Yan R, Tang H. Deep spiking neural networks with binary weights for object recognition. IEEE Transactions on Cognitive and Developmental Systems, 2021, 13(3): 514−523
[30] Pfeiffer M, Pfeil T. Deep learning with spiking neurons: opportunities and challenges. Frontiers in Neuroscience, 2018, 12: 774
[31] Eshraghian J K, Lu W D. The fine line between dead neurons and sparsity in binarized spiking neural networks. 2022, arXiv preprint arXiv: 2201.11915
[32] Qiao G, Ning N, Zuo Y, Hu S, Yu Q, Liu Y. Direct training of hardware-friendly weight binarized spiking neural network with surrogate gradient learning towards spatio-temporal event-based dynamic data recognition. Neurocomputing, 2021, 457: 203−213
[33] Jang H, Skatchkovsky N, Simeone O. BiSNN: training spiking neural networks with binary weights via Bayesian learning. In: Proceedings of 2021 IEEE Data Science and Learning Workshop. 2021, 1−6
[34] Kheradpisheh S R, Mirsadeghi M, Masquelier T. BS4NN: binarized spiking neural networks with temporal coding and learning. Neural Processing Letters, 2022, 54(2): 1255−1273
[35] Pei Y, Xu C, Wu Z, Liu Y, Yang Y. ALBSNN: ultra-low latency adaptive local binary spiking neural network with accuracy loss estimator. Frontiers in Neuroscience, 2023, 17: 1225871
[36] Rueckauer B, Lungu I A, Hu Y, Pfeiffer M, Liu S C. Conversion of continuous-valued deep networks to efficient event-driven networks for image classification. Frontiers in Neuroscience, 2017, 11: 682
[37] Li F, Liu B, Wang X, Zhang B, Yan J. Ternary weight networks. 2016, arXiv preprint arXiv: 1605.04711
[38] Bengio Y, Léonard N, Courville A. Estimating or propagating gradients through stochastic neurons for conditional computation. 2013, arXiv preprint arXiv: 1308.3432
[39] Zhang S H, Tang W C, Wu C, Hu P, Li N, Zhang L J, Zhang Q, Zhang S Q. TernaryCLIP: efficiently compressing vision-language models with ternary weights and distilled knowledge. 2025, arXiv preprint arXiv: 2510.21879
[40] Bottou L. Large-scale machine learning with stochastic gradient descent. In: Proceedings of the 19th International Conference on Computational Statistics. 2010, 177−186
[41] Kingma D P, Ba J. Adam: a method for stochastic optimization. 2014, arXiv preprint arXiv: 1412.6980
[42] Kendall A, Gal Y. What uncertainties do we need in Bayesian deep learning for computer vision? In: Proceedings of the 31st International Conference on Neural Information Processing Systems. 2017, 5580−5590
[43] Gal Y, Ghahramani Z. Dropout as a Bayesian approximation: representing model uncertainty in deep learning. In: Proceedings of the 33rd International Conference on International Conference on Machine Learning. 2016, 1050−1059
[44] Lecun Y, Bottou L, Bengio Y, Haffner P. Gradient-based learning applied to document recognition. Proceedings of the IEEE, 1998, 86(11): 2278−2324
[45] Krizhevsky A. Learning multiple layers of features from tiny images. Toronto: University of Toronto, 2009
[46] Shrestha S B, Orchard G. SLAYER: spike layer error reassignment in time. In: Proceedings of the 32nd International Conference on Neural Information Processing Systems. 2018, 1419−1428
[47] Fang W, Yu Z, Chen Y, Masquelier T, Huang T, Tian Y. Incorporating learnable membrane time constant to enhance learning of spiking neural networks. In: Proceedings of 2021 IEEE/CVF International Conference on Computer Vision. 2021, 2641−2651
[48] Sakai Y, Pedroni B U, Joshi S, Tanabe S, Akinin A, Cauwenberghs G. Dropout and DropConnect for reliable neuromorphic inference under communication constraints in network connectivity. IEEE Journal on Emerging and Selected Topics in Circuits and Systems, 2019, 9(4): 658−667
[49] Fang W, Chen Y, Ding J, Yu Z, Masquelier T, Chen D, Huang L, Zhou H, Li G, Tian Y. SpikingJelly: an open-source machine learning infrastructure platform for spike-based intelligence. Science Advances, 2023, 9(40): eadi1480
[50] Yu Q, Tang H, Tan K C, Li H. Rapid feedforward computation by temporal encoding and learning with spiking neurons. IEEE Transactions on Neural Networks and Learning Systems, 2013, 24(10): 1539−1552
[51] Xu Q, Qi Y, Yu H, Shen J, Tang H, Pan G. CSNN: an augmented spiking based framework with perceptron-inception. In: Proceedings of the 27th International Joint Conference on Artificial Intelligence. 2018, 1646−1652
[52] Liu Q, Pan G, Ruan H, Xing D, Xu Q, Tang H. Unsupervised AER object recognition based on multiscale spatio-temporal features and spiking neurons. IEEE Transactions on Neural Networks and Learning Systems, 2020, 31(12): 5300−5311
[53] Zhao B, Ding R, Chen S, Linares-Barranco B, Tang H. Feedforward categorization on AER motion events using cortex-like features in a spiking neural network. IEEE Transactions on Neural Networks and Learning Systems, 2015, 26(9): 1963−1978
[54] Wang T, Shi C, Zhou X, Lin Y, He J, Gan P, Li P, Wang Y, Liu L, Wu N, Luo G. CompSNN: a lightweight spiking neural network based on spatiotemporally compressive spike features. Neurocomputing, 2021, 425: 96−106
[55] Roy D, Chakraborty I, Roy K. Scaling deep spiking neural networks with binary stochastic activations. In: Proceedings of 2019 IEEE International Conference on Cognitive Computing. 2019, 50−58
[56] Yan Z, Bai Z, Wong W F. Reconsidering the energy efficiency of spiking neural networks. 2024, arXiv preprint arXiv: 2409.08290

Yu-Lun WU is currently a undergraduate student in the School of Intelligence Science and Technology at Nanjing University, China. His research interests lie primarily in spiking-based computing and model lightweighting techniques, focusing on the development of efficient and low-power next-generation artificial intelligence systems.

Rui-Rui TAN is currently a undergraduate student in the School of Intelligence Science and Technology at Nanjing University, China. Her research interests include abduction learning and spiking-based computing.

Shu-Hao ZHANG is currently a PhD candidate with the State Key Laboratory of Novel Software Technology at Nanjing University, China under the supervision of Professor Shao-Qun Zhang. He received his M.Sc. degree in computer science from The University of Hong Kong, China and his B.Eng. degree in software engineering from Nankai University, China. His research interests focus on lightweight computation and large language models.

Shuang LIANG is a graduate student with the National Key Laboratory for Novel Software Technology at Nanjing University, China under the supervision of Professor Shao-Qun Zhang. He received his B.Sc. degree in statistics from Sichuan University, China. His research interests include machine learning and uncertainty analysis.

Zi-Ang LIU is an R&D Engineer at China Mobile Zijin Innovation Institute, China. He obtained his master's degree in machine learning and computer vision from the Australian National University, Australia, and currently leads R&D projects related to industrial visual inspection. His research includes industrial artificial intelligence and computer vision.

Zhao WANG serves as Deputy Manager of the Industrial Internet R&D Department at China Mobile Zijin Innovation Institute, China. He is primarily responsible for leading technical research, architecture design, and product development in the fields of industrial internet platforms, industrial robotics, computer vision, large language models, and AI agents.

Shao-Qun ZHANG is an assistant professor in the School of Intelligence Science and Technology at Nanjing University, China. His research interests primarily include topics in machine learning and data mining, particularly intelligence-inspired computing and time series analysis.