Toward High-Accuracy and Low-Latency Spiking Neural Networks With Two-Stage Optimization

Ziming Wang©, Yuhao Zhang©, Shuang Lian©, Xiaoxin Cui©, Member, IEEE, Rui Yan©, Member, IEEE, and Huajin Tang©, Senior Member, IEEE

Abstract—Spiking neural networks (SNNs) operating with asynchronous discrete events show higher energy efficiency with sparse computation. A popular approach for implementing deep SNNs is artificial neural network (ANN)–SNN conversion combining both efficient training of ANNs and efficient inference of SNNs. However, the accuracy loss is usually nonnegligible, especially under few time steps, which restricts the applications of SNN on latency-sensitive edge devices greatly. In this article, we first identify that such performance degradation stems from the misrepresentation of the negative or overflow residual membrane potential in SNNs. Inspired by this, we decompose the conversion error into three parts: quantization error, clipping error, and residual membrane potential representation error. With such insights, we propose a two-stage conversion algorithm to minimize those errors, respectively. In addition, we show that each stage achieves significant performance gains in a complementary manner. By evaluating on challenging datasets including CIFAR-10, CIFAR-100, and ImageNet, the proposed method demonstrates the state-of-the-art performance in terms of accuracy, latency, and energy preservation. Furthermore, our method is evaluated using a more challenging object detection task, revealing notable gains in regression performance under ultralow latency, when compared with existing spike-based detection algorithms. Codes will be available at: https://github.com/Windere/snn-cvt-dual-phase.

Index Terms—Artificial neural network (ANN)–spiking neural network (SNN) conversion, deep SNNs, neuromorphic computing, residual membrane potential, SNN, spike-based object detection, two-stage optimization.

I. INTRODUCTION

S

PIKNG neural networks (SNNs), inspired by mimicking the dynamics of biological neurons [1], [2], [3], have gained increasing interest [4]. Different from the computation with single, continuous-valued activation in artificial neural networks (ANNs), SNNs use binary spikes to transmit and process information. Each neuron in the SNNs will remain silent without energy consumption until receiving a spike/event afferent [5]. Such an event-driven computing paradigm enables more power-efficient solutions on dedicated neuromorphic hardware by substituting dense multiplication with sparse addition [6]. As reported in [7], [8], and [9], SNNs on specified neuromorphic processors can achieve orders of magnitude lower energy consumption and latency compared with ANNs. In addition, SNNs could synergistically help denoise redundant information with inherent temporal dynamics [10]. However, it is still an open problem how to obtain highly performance SNNs efficiently.

In general, there are two mainstream methodologies for developing supervised deep SNNs up to date: 1) direct training for SNNs and 2) converting ANNs into SNNs. For direct training methods, the back-propagation technique could not be applied to SNNs directly due to the threshold-crossing firing in spiking neurons, which presents challenges for accurately calculating gradients in both spatial and temporal domains. Some researchers have suggested to circumvent this difficulty by designing a surrogate function and smoothing the nondifferentiable spike firing [11], [12], [13], [14], [15]. To efficiently utilize the temporal structure in spike trains, the time-based scheme is proposed to directly assign credits to the shift of spike time in SNNs while taking both internuron and intraneuron dependencies [16], [17], [18], [19], [20], [21] into account. Binary probabilistic models [22], [23] utilize stochasticity (e.g., the log-likelihood of a spike) to approximate the expectation value of gradients in SNNs. Dedicated to improving the performance, transfer learning in SNNs is also explored through transplanting features [24] or mitigating domain shift [25]. Nevertheless, it is still hard to train large-scale SNNs due to the limited memory capacity, the short-time dependency, and the vanishing spike rate in deep networks. Moreover, training SNNs on graphics processing units (GPUs) invariably incurs additional computation and memory overhead, which scales proportionally with the time step [26], as there is no specific optimization for storage and operations with binary events.

The alternative approach, known as ANN–SNN conversion, involves obtaining SNNs from pretrained ANNs, which has resulted in some best-performing SNNs on large-scale datasets like ImageNet [27] with significantly lower training costs compared with direct training. The core idea of conversion is...
to establish the consistent relationship between activations of analog neurons and some kind of aggregate representation of spiking neurons, such as spike count [28], spike rate [29], [30], and postsynaptic potential (PSP) [31]. With such clear criteria, ANN–SNN conversion has been applied to complex scenarios with competitive performances compared with ANNs [32], [33], [34], [35]. Nevertheless, to achieve enough representation precision, considerable simulation steps are usually required for nearly lossless conversion, known as the accuracy-delay trade-off. It restricts the practical application of SNNs greatly. A large body of recent work [26], [36], [37], [38] proposes to alleviate this problem by exploiting the quantization and clipping properties of aggregation representations. Even though, there is still a significant performance gap between ANNs and SNNs under low inference latency (≤16 time steps). Some recent works have further bridged this performance gap by introducing a signed neuron model with a customized spike counter [39] or a neuron model firing burst spikes in a single time step [40], [41]. However, the underlying cause of performance degradation when mapping ANNs into SNNs is still unclear. Therefore, our focus is on analyzing conversion errors and improving the accuracy-delay trade-off based on vanilla spiking neurons.

In this article, we explicitly identify that the conversion error under few time steps primarily arises from the misrepresentation of the residual membrane potential, which can accurately characterize information loss between the input and output of spiking neurons with asynchronous spike firing. Furthermore, we demonstrate that the error regarding residual potential representation is complementary to quantization and clipping errors. Inspired by this, we propose an ANN–SNN conversion algorithm with a two-stage optimization approach to mitigating these three types of errors. This approach achieves remarkable performance with extremely low inference delay. The main contributions of this work are summarized as follows.

1) We analyze the operator consistency between ANNs and SNNs theoretically and identify the neglected residual potential representation problem. Then, we divide conversion errors into three parts: quantization error, clipping error, and residual potential representation error.
2) We propose a two-stage scheme for the threefold errors toward lossless conversion under ultralow inference delay. In the first stage, quantization-clipping functions with trainable thresholds and quantization noise are applied to fine tune ANNs. In the second stage, we minimize the residual potential error (RPE) with layer-wise calibrations on weights and initial membrane potential. In addition, we further extend it into ANNs with leaky rectified linear unit (ReLU) as activation.
3) Experimental results on the both CIFAR and ImageNet datasets show significant improvements in accuracy–latency trade-offs compared with state-of-the-art methods across diverse architectures, including ResNet, VGG, ResNeXt, and MobileNet. For example, we achieved 70.13% top-1 accuracy (19.16% improvements) on ImageNet with VGG-16 under only 16 time steps.
4) A spike-based object detection model is implemented based on the proposed method. The results on the PASCAL VOC dataset demonstrate competitive detection performance with at least $25 \times$ inference speedup in comparison with existing spike-based object detectors.

II. RELATED WORK

Cao et al. [42] first proposed to convert pretrained ANNs into SNNs and suggested the criterion of matching ANN activation and spiking rate of SNN. After the launch of ANN–SNN conversion, the development of conversion algorithms could be divided into two routes in general.

A. From the Perspective of Constrained ANNs

Diehl et al. [29] found the importance of weight-threshold balance and design weight normalization based on the maximum of layer-wise ANN activations. Afterward, a lot of work was dedicated to developing more elaborate normalization factors, such as robust normalization [30], spike-based normalization [43], channel-wise normalization [32], threshold-balancing [44], and normalization on shortcut connections [45]. These methods can be sufficiently integrated with ReLU-based ANN to achieve lossless conversion. However, the inference delay is up to hundreds or thousands in general. Exploring the characteristics of quantization and clipping in spike rate, Yan et al. [36] first proposed the training scheme with quantization and clipping constraints for ANNs, namely, clipping and quantization (CQ)-training. Similarly, Ding et al. [46] proposed the method for weight and threshold training by stages to optimize the upper bound of the error between ReLU and CQ activations. Ho and Chang [38] suggested a trainable clipping bound in activation functions to balance the threshold. Recently, Bu et al. [37] approximated the activation of SNNs through a quantization clip-floor-shift function and explored the conversion error, when the quantization step in ANNs and the time step in SNNs are mismatched. In addition to the conversion of full-precision weights, Wang et al. [47] further explored the high-performance conversion under binary weights by rectifying normalization coefficients.

B. From the Perspective of Modified SNNs

The soft reset mechanism is widely adopted [6], [42] to avoid information loss from potential reset. Deng and Gu [31] decomposed the network conversion error into the layer-wise conversion error and proposed the extra shift of $\theta/(2T)$ in spiking neurons to reduce the expectation of quantization error. Comparably, Hu et al. [45] and Bu et al. [48] configured the initial membrane potential of spiking neurons as $\theta/2$ to reduce quantization error. Furthermore, Li et al. [26] exploited the calibration effect of a handful of samples through activation transplanting to reduce clipping error and quantization error. Yu et al. [40] first show the performance of SNNs could be enhanced greatly with augmented spike and double-threshold schemes. Wang et al. [39] introduced a signed neuron model with a specific spike counter to compensate for the inconsistency between synchronous ANNs and asynchronous SNNs. Li et al. [41] enhanced spiking neurons by allowing bursting spikes in a single time step, reducing the information loss.
In addition to rate coding, customized neuron models [49], [50] based on temporal coding were also investigated further to exploit the temporal dynamics of spiking neurons. The work introduced by Hwang et al. [51] reduced inference latency by sequentially searching for the optimal initial membrane potential. By modulating the membrane potential toward a steady firing state of spiking neurons, those methods implicitly alleviate the representation problem from residual potential. However, without explicitly identifying and formulating the error function from residual potential, it is challenging to design targeted optimization strategies.

Different from previous approaches, this work is the first to explicitly identify the source of error responsible for the stable firing state, called the RPE. Furthermore, we have introduced an additional fine-tuning stage that is specifically designed to minimize this particular source of error, thereby enabling high-performance converted SNNs with ultralow latency.

III. PRELIMINARIES

A. Analog Neuron Model

Analog neurons in feedforward neural networks, such as CNN and MLPs, communicate and learn with continuous activations. Mathematically, the forward computation of the $l$th layer in feedforward networks is formed as follows:

$$a^l_i = \sigma(z^l_i) = \sigma \left( \sum_j W^l_{ij} \cdot a^{l-1}_j + b^l_i \right)$$  \hspace{1cm} (1)

where $a^l_i$ is the output of ReLU activation function $\sigma(x) = \max(0, x)$. $W^l_{ij}$ and $b^l_i$ are the weights and bias of neuron $i$ in the $l$th layer, respectively.

B. Spiking Neuron Model

Integrate-and-fire (IF) neuron model is widely used in conversion algorithms [29], [30], [42], [43], [45] because of the low computing cost and robust representation on firing rate. At each simulating time step $t$, the IF neuron $i$ receives afferent spikes $s^{l-1}_j[t]$ and updates its state $u^l_i[t]$ by integrating the input potential $v^l_i[t]$

$$u^l_i[t] = \hat{u}^l_i[t-1] + v^l_i[t]$$  \hspace{1cm} (2)

where $u^l_i[t]$ and $\hat{u}^l_i[t]$ denote the membrane potential before and after reset, respectively. The neuron generates a spike $s^l_i[t]$ and resets the membrane potential, whenever $u^l_i[t]$ exceeds the firing threshold $\theta^l_i$

$$s^l_i[t] = \Theta(u^l_i[t] - \theta^l_i) \text{ with } \Theta(x) = \begin{cases} 1, & \text{if } x \geq 0 \\ 0, & \text{otherwise.} \end{cases}$$  \hspace{1cm} (3)

Specifically, we adopt the widely used soft-reset (reset-by-subtraction) mechanism [6], [42] rather than resetting as a constant to reduce information loss from the membrane potential over the threshold. Formally, the soft reset mechanism is presented as follows:

$$\hat{u}^l_i[t] = u^l_i[t] - s^l_i[t] \cdot \theta^l_i.$$  \hspace{1cm} (4)

C. Fusing Batch Normalization

Batch normalization is a crucial component for most CNN architecture as it helps mitigate the internal covariate shift. The absence of batch normalization can be detrimental to network convergence and generalization. However, there is no direct equivalent module for batch normalization in SNNs as standard normalization methods damage the binary nature of spikes. Therefore, as done in the most literature [26], [30], [31], [36], [41], [45], [46], we absorb the batch normalization on activations and transform it into batch normalization on weights and bias at the beginning, without detrimental effect on performance [52], [53]

$$\text{BN}_w(W^l_{ij}) = \frac{\gamma^l_i}{\sqrt{(\sigma^l_i)^2 + \epsilon}} W^l_{ij}$$

$$\text{BN}_b(b^l_i) = \frac{\gamma^l_i}{\sqrt{(\sigma^l_i)^2 + \epsilon}} (b^l_i - \mu^l_i) + \beta^l_i$$  \hspace{1cm} (5)

where $\mu^l_i$ and $\sigma^l_i$ represent the moving mean and moving variance of $a^l_i$ estimated from batch statistics during training. $\gamma^l_i$ and $\beta^l_i$ are the trainable parameters in the batch normalization layer.

D. Input and Readout

Instead of employing explicit spike coding, we opt to directly inject the image as input current repeatedly into SNNs at each time step, thereby avoiding information loss as done in [30] and [54]. In essence, the first layer in SNNs is responsible for encoding the analog pixels into discrete spikes. In addition, the average input potential $\bar{v}^l_i = (\sum_{t=1}^{T} v^l_i[t]) / T$ rather than spike rate $r^l_i = (\sum_{t} s^l_i[t]) / T$ is read out in the last layer.

IV. ERROR ANALYSIS

The goal of this article is to build a consistent mapping between spike rates $r^l_i$ in SNNs and activation values $a^l_i$ in ANNs, as shown in Fig. 1. First, we derive the exact relation between average input potential $\bar{v}^l_i$ and input spike rate $r^{l-1}_j$

$$\bar{v}^l_i = \frac{\sum_{t=1}^{T} \sum_j W^l_{ij} s^{l-1}_j[t] + b^l_i T}{T} = \sum_j W^l_{ij} r^{l-1}_j + b^l_i.$$  \hspace{1cm} (6)

Compared with (1), the functional relation between $\bar{v}^l_i$ and $r^{l-1}_j$ in IF neurons is identical to that between $z^l_i$ and $a^{l-1}_j$ in analog neurons. Then, the remaining question is what is the difference between the function $\bar{v}^l_i \to r^l_i$ and the function $z^l_i \to a^l_i$? By substituting (4) into (2) and accumulating from 0 to $T$, the relation between $r^l_i$ and $r^{l-1}_j$ is deduced as follows:

$$r^l_i = \sum_j \left( W^l_{ij} r^{l-1}_j + b^l_i \right) / \theta^l_i - \epsilon^l_i$$  \hspace{1cm} (7)
where \( \epsilon_i^r = (\hat{u}_i^r[T] - \hat{u}_i^r[0])/(T \theta_i^r) \) is the component about residual membrane potential \( \hat{u}_i^r[T] \). By substituting (6) into (7), we obtain the exact expression between \( \overline{v}_i^r \) and \( r_i^r \)

\[
\begin{align*}
  r_i^r &= f_{0,T}(\overline{v}_i^r, \epsilon_i^r) = \overline{v}_i^r/\theta_i^r - \epsilon_i^r. 
\end{align*}
\]

Remarkably, \( \epsilon_i^r \) is highly nonlinear, nondifferentiable, and nonconvex as \( \hat{u}_i^r[T] \) is calculated with the compound of \( T \) Heaviside functions \( \Theta(x) \) explicated in (2)–(4). Therefore, the only divergence of operators appears between the equivalent activation function \( f_{0,T}(x, \epsilon) \) in SNNs and \( \sigma(x) \) in ANNs. To bypass the nondeterminism and complexity of \( \epsilon \), we start from the limited residual membrane potential assumption, as illustrated in Fig. 2(c)

\[
0 \leq \hat{u}_i^r[T] - \hat{u}_i^r[0] < \theta_i^r. \tag{9}
\]

Under such assumption, \( \epsilon_i^r \) is bounded in \([0, (1/T)]\), whereas \( r_i^r \) is \( \{0/T), (1/T), \ldots, (T/T)\). The nonlinearity of \( f_{0,T}(x, \epsilon) \) can be approximated using the floor function with a quantization step of \( \theta/T \) (indicated by the green line in Fig. 3)

\[
g_{\theta,T}(x) = \frac{T}{\theta} \left \lfloor \frac{x}{\theta} \right \rfloor = \frac{T}{\theta} \left \lfloor x/\theta_T \right \rfloor. \tag{10}
\]

Furthermore, the spike rate clipping technique is adopted to compensate for the error between \( f_{0,T}(x) \) and \( g_{\theta,T}(x) \) when the condition of (9) is not satisfied

\[
g_{\theta,T}(x) = \text{clip} \left( \frac{1}{\theta} \left \lfloor x/\theta_T \right \rfloor, 0, 1 \right). \tag{11}
\]

However, the errors still exist and remain nonnegligible even with such compensation. It is often attributed to transient dynamics [30], temporal jitter of spike trains [28], or unevenness error [37] without dedicated optimization. However, we contend that it is more precise and tractable to attribute such error to an incomplete representation of residual membrane potential rather than the irregular discharge of spikes in IF models.

To illustrate this, we handcraft three examples under uniform and irregular spike distribution, respectively, in Fig. 2 as follows.

**Example 1 (Regular Spike Emissions With Undervalued Rate):** All the input neurons emit spikes regularly with an interval of two time steps in Fig. 2(a). However, the output of floor-clipping function \( g_{\theta,T}(\overline{v}) \) is 0, while the real spike rate \( r = f_{0,T}(\overline{v}, \epsilon) \) is \((1/6)\). The spike rate is undervalued with negative residual potential even under such uniform spike afferents.

**Example 2 (Irregular Spike Emissions With Overvalued Rate):** The output neuron is in a saturation state with a real spike rate \( r = (2/6)\), while the expectation \( g_{\theta,T}(v) \) is \((3/6)\) in the example of Fig. 2(b). The spike rate is overvalued with overflow residual potential.

**Example 3 (Irregular Spike Emissions With Correct Rate):** The output neuron is in a state that yields the correct output rate \( r = g_{\theta,T}(v) = (1/6)\) as shown in the example of Fig. 2(c). The spike rate is estimated appropriately with feasible residual potential although the afferent spikes are irregular.

Therefore, the estimated spike rate may either surpass or fall below the actual rate of IF neurons in both cases, as shown in Fig. 2(a) and (b). In addition, the spike rate can also be evaluated exactly with irregular spike afferents [Fig. 2(c)]. The fundamental cause of the error is that the IF neuron with soft-reset mechanism cannot respond to the residual membrane potential out of \([0, \theta]\). Then, we refer to the kind of error as residual membrane potential representation error, RPE in short. It measures the divergence between the real
A. Perspective I (Divergence of Operators)

We generate 100 periodically injected sequences \( \mathcal{V} = \{ v_1[r], v_2[r], \ldots, v_{100}[r] \} \), where each sequence \( v[i] \) is sampled with ten time steps from sine functions of different frequencies. By injecting each sequence as current into the single spiking neuron, we obtain the real spike rate \( r \) through iterative equations [(2) and (3)]. Meanwhile, we adopt the quantization-clipping function \( g_{0,T}(\theta, v[i]) \) to estimate the response spike rate \( \hat{a} \). As shown in Fig. 5(a), there is a prominent bias compared with the identical mapping between \( \hat{a} \) and \( r \) even in the ideal condition which illustrates the fundamental divergence of operators between \( g_{0,T}(\mathcal{V}, \epsilon) \) and \( f_{\theta,T}(\mathcal{V}, \epsilon) \), i.e., RPE, on the single neuron.

\[ \text{RPE} = \| x_f - x_r \|_2 / \| x_f \|_2 \] 

B. Perspective II (Error Accumulation)

We calculate the cosine similarities (\( \text{sim}(x_1, x_2) = x_1^T x_2 / (\| x_1 \|_2 \| x_2 \|_2) \)) of activations between the source ANN and QC-ANN on 1024 samples to measure the error accumulation of RPE. This analysis serves to quantify the accumulation of RPE. Upon comparing the similarity curves of different layers [see Fig. 5(b)], we observe that shallower layers tend to exhibit higher similarities in general. It demonstrates that divergence of operations (RPE) in former layers brings cumulative errors through subsequent layers. In addition, it is observed that the similarities gradually increase with the growth of the time step \( T \). This phenomenon happens because, in this case, the residual term \( \epsilon^l = (\hat{u}[T] - \hat{u}[0]) / (T \theta^l) \) diminishes progressively.

C. Perspective III (Performance Gap)

To observe the performance gap resulting from cumulative errors, Fig. 5(c) displays the accuracy curves of source ANN, QC-ANN, and SNN over various time steps. The performance gap between QC-ANN and converted SNN demonstrates the conversion loss caused by neglected RPE about \( \epsilon \), while the difference between QC-ANN and source ANN indicates the impact of QE and CE. It is worth noting that the error resulting from RPE generally makes greater contributions to the overall conversion error from source ANN to SNN than QE and CE.

The theory and experiments in the analysis above consistently show that RPE works in a complementary manner with respect to QE and CE. In addition, as illustrated in Fig. 5(c), the neglected RPE gives a wider optimization space for few time steps. Therefore, this phenomenon motivates us to rethink the conversion errors and redesign the algorithm exploiting the RPE.

V. Method

Based on the above error analysis, we propose a two-stage conversion scheme (Fig. 6) toward lossless conversion by minimizing QE, CE, and RPE in stages. Overall, we start from a source neural network using the clipping function \( \sigma(x) \) as the activation function, then transfer it to a neural network with trainable quantization-clipping activation function \( g_{0,T}(x) \), called QC-ANN later. Therefore, the QE and CE are optimized with the fine tuning of the QC-ANN. Finally, the
layer-wise calibration on weights and initial membrane potential is adopted to optimize the RPE based on the activation divergence between \( g_{θ,T}(x) \) and \( f_{θ,T}(\xi, t_i^e) \).

A. Stage-I: QC-Finetuning With Trainable Threshold

The principle of the first stage is to minimize the QE and CE (right part of Fig. 1) and to take the quantization and clipping property of estimated activation function \( g_{θ,T}(x) \) for SNNs into ANN training. Therefore, we introduce the QC-ANN with trainable thresholds \( t_i^f \) as the intermediate network.

1) Convert ANNs Into QC-ANNs: To avoid the high computational cost due to training from scratch with QC activation function \( g_{θ,T}(x) \) under different \( T \), we start with training source ANNs with activation \( σ_θ(x) = \text{clip}(1/θ, x, 0, 1) \) unrelated to the specified time steps \( T \). Then, we convert the source ANN into QC-ANN by simply copying the network parameters and replace the corresponding activation functions. Here, we adopt a similar shift term \( Δx = θ/(2T) \) proposed in [31] to minimize the distance \( ||σ_θ(x) - g_{θ,T}(x + Δx)||^2 \) under different time steps \( T \). Therefore, the floor term in \( g_{θ,T}(x) \) in QC-ANN is replaced by a round term

\[
g_{θ,T}(x) = \text{clip}\left(\frac{1}{θ} Q_{θ/T}(x), 0, 1\right)
\]

where

\[
Q_{Δ}(x) = Δ \cdot \text{round}\left(\frac{x}{Δ}\right).
\]

2) Fine Tuning With Quantization Noise: Although the optimal shift is applied, the QE and CE still exist and contribute to the conversion error. Therefore, we fine tune the QC-ANN after copying weights from source ANN and replace the activation function \( σ_θ(x) \) with \( g_{θ,T}(x) \). Notably, the round term \( Q_1(x) \) in \( g_{θ,T}(x) \) has null gradients, which means that the derivative of the input is 0 almost everywhere. To train QC-ANN with such an ill-defined function, we adopt the widely used straight through estimator (STE) [55] to estimate the gradients

\[
\frac{∂ Q_{Δ}(x)}{∂ x} = \frac{∂ Q_1(x)}{∂ x} = \frac{∂ |x|}{∂ x} = 1.
\]

Nevertheless, in the STE scheme, most weights are updated with unbiased gradients. It brings the larger bias with the increasing quantization step \( θ/T \), when the time step is relatively low \( (T < 8) \). Furthermore, the convergence of the QC-ANN and the fast-inference ability of converted SNNs are inevitably limited. Here, we propose to correct the biased gradients using the noisy quantization function \( Q_{Δ}(x) \) inspired by Fan et al. [56], which introduces quant-noise in model compression. Specifically, we replace a random fraction of \( Q_Δ(x) \) with the identity mapping \( x \) at each forward

\[
\hat{Q}_{Δ}(x) = x + \left( Q_{Δ}(x) - x \right) \cdot M
\]

where \( M \sim \text{Bernoulli}(p) \) is the random mask. Then, we further modify (12) during QC fine-tuning as follows:

\[
g_{θ,T}(x) = \text{clip}\left(\frac{1}{θ} \hat{Q}_{θ/T}(x), 0, 1\right).
\]

Thus, most weights in QC-ANN are updated with unbiased gradients boosting the convergence under large quantization steps.

B. Stage-II: Layer-Wise Calibration With BPTT

After copying the weights, biases, and thresholds trained in QC-ANN into the target SNN, we get an SNN without considering RPE. To optimize RPE, the core objective of the second stage is shown in the left part of Fig. 1, we adopt
a layer-wise calibration mechanism on weights $W$ and initial membrane potential $\hat{u}[0]$. First, we modify the real activation function $g_{0,T}(x)$ on spike rate accordingly with respect to the shift $\theta/(2T)$ adopted in $g_{0,T}(x)$

$$f_{0,T}(v, \epsilon) = \left( v + \frac{\theta}{2T} \right) / \theta - \epsilon.$$  

(16)

By calibrating $f_{0,T}(v, \epsilon)$ toward $g_{0,T}(v)$ further, we effectively narrow the performance gap without the need for specialized neuron design.

1) Coarse Calibration on Initial Membrane Potential: When (9) is not satisfied, the initial membrane potential $\hat{u}[0]$ at the $l$th layer plays a coarse adjustment role for the activation distribution as shown in (7). Considering the offline strategy of calibration-before-inference on $N$ samples, the core for calibrating $\hat{u}[0]$ lies in solving the following optimization equation:

$$\min_{\hat{u}[0]} \left\{ \sum_{i=1}^{N} \left( \frac{\hat{v}_i}{\theta} + \frac{\hat{u}[T] - \hat{u}[0]}{T\theta} - g_{0,T}(z_i) \right)^2 \right\}.$$  

(17)

Suppose $\hat{u}[T]$ is unrelated to $\hat{u}[0]$, then the problem turns into a linear form and the corresponding solution is

$$\hat{u}[0] = \frac{T\theta}{N} \sum_{i=1}^{N} \left( g_{0,T}(z_i) - f_{0,T}(\hat{v}_i, \epsilon_i; \hat{u}[0] = 0) \right)$$

$$= \frac{T\theta}{N} \sum_{i=1}^{N} (\hat{u}_i - r_i).$$  

(18)

By the way, we can roughly correct the output expectation of the target SNN to match that of the QC-ANN. In addition, CC has a relatively low calculation overhead that is proportionate to inference on $N$ calibration samples.

2) Fine Calibration (FC) Considering Temporal Dynamics: To calibrate the RPE more precisely, we measured the KL divergence between the target activation $\hat{a}_t^l$ and real spike rate $r_t^l$ in the recognition task

$$\min_W \mathcal{L}(r_t^l, \hat{a}_t^l) = D_{KL}(\hat{a}_t^l \| f_{\theta,T}(\hat{v}(W), \epsilon(W))).$$  

(19)

Similarly, the MSE loss is adopted to improve the numerical precision of coordinate regression in object detection. Although $\hat{v}$ is the linear combination of $W^l$, $\epsilon^l$ is a superposition of nondifferentiable Heaviside functions $\Theta(x)$.

In order to solve such nonconvex and nondifferentiable optimization problems, we adopt a rectangular surrogate function $h(u)$ [54] to smooth the illness gradient $(\partial \Theta(u)/\partial u)$

$$\frac{\partial \Theta(u)}{\partial u} \approx h(u) = \frac{1}{\alpha} \text{sign}(|u - \theta| < \frac{\alpha}{2})$$  

(20)

where $\alpha$ controls the smoothness degree for $(\partial \Theta(u)/\partial u)$. Then, we obtain the gradients of weights with backpropagation-through-time (BPTT)

$$\frac{\partial \mathcal{L}}{\partial W_j^l} = \frac{1}{T} \frac{\partial \mathcal{L}}{\partial r_t^l} \sum_{t=1}^{T} \sum_{i=1}^{l} \frac{\partial \hat{s}_j^l[t]}{\partial u_t^l[t]} \delta_{j-1}[t]$$

$$= \frac{1}{T} \frac{\partial \mathcal{L}}{\partial r_t^l} \sum_{t=1}^{T} \frac{\partial \hat{s}_j^l[t]}{\partial u_t^l[0]}$$

with $S' = [s_0', 1, \ldots, s_T]$.

Algorithm 1: Algorithm for the two-stage ANN-SNN Conversion

**Input** : Source ANN model $\mathcal{F}_{\text{ANN}}(W; x)$ with activation function $\sigma(x) = \text{clip}(1/(\theta/x), 0, 1)$; Dataset $D$; Time step $T$; Total epoch $E$

**Output**: SNN model $\mathcal{F}_{\text{SNN}}(W; x)$

/* Stage-I: QC-Finetuning */
1. Merge BN layers into weights and bias.
2. For each layer $\mathcal{F}_{\text{ANN}}(W; x)$ in $\mathcal{F}_{\text{ANN}}(W; x)$ do
3. if $\mathcal{F}_{\text{ANN}}(W; x)$ is instance of activation function $\sigma_0(x)$ then
4. replace $\sigma_0(x)$ with $g_{0,T}(x)$ (Eq. 15) with the shared parameter $\theta$
5. Finetune the obtained $\mathcal{F}_{\text{QC-ANN}}(W; x)$ with $g_{0,T}(x)$ as activation function on $D$
6. Copy weights from $\mathcal{F}_{\text{QC-ANN}}(W; x)$ into the SNN $\mathcal{F}_{\text{SNN}}(W; x)$ and introduce the shift $\theta/2T$ (Eq. 16)
/* # Stage-II: Layer-wise Calibration with BPTT */
7. Sample a calibration subset $\hat{D}$ from $D$
8. For $l \leftarrow 1$ to $\mathcal{F}_{\text{ANN}}(W; x).\text{layers}$ do
9. Fetch the activation $\hat{a}_t^l$ of $\mathcal{F}_{\text{QC-ANN}}(W; x)$ on $\hat{D}$
10. Fetch the output $r_t^l$ of $\mathcal{F}_{\text{SNN}}(W; x)$ on $\hat{D}$
11. Calibrate initial membrane potential $\hat{u}_0[0]$ (Eq. 18)
12. For $l \leftarrow 1$ to $\mathcal{F}_{\text{ANN}}(W; x).\text{layers}$ do
13. For $e \leftarrow 1$ to $E$ do
14. For $b \leftarrow 1$ to $\mathcal{F}_{\text{SNN}}(W; x).\text{batches}$ do
15. Fetch the activation $\hat{a}_t^l$ of $\mathcal{F}_{\text{QC-ANN}}(W; x)$ on calibration samples $\hat{D}_b$
16. Fetch the input $S' = [s_0', 1, \ldots, s_T]$ of $\mathcal{F}_{\text{SNN}}(W; x)$ on calibration samples $\hat{D}_b$
17. Feedforward on the spiking layer $\mathcal{F}_{\text{SNN}}$ with $S'$ and calculate the spike rate $r_t^l$
18. Calculate the gradients $(\partial \mathcal{L}/\partial W)$ and $(\partial \mathcal{L}/\partial U[0])$ (Eq. 21)
19. Update $W$ and $U[0]$
20. Return $\mathcal{F}_{\text{SNN}}(W; x)$

$$\frac{\partial s_j^l[t^*]}{\partial u_t^l[t]} = \begin{cases} h(u_t^l[t^*]), & \text{if } t = t^* \\ \frac{\partial s_j^l[t^*]}{\partial u_t^l[t]} (1 - \theta_t^l h(u_t^l[t])), & \text{if } t < t^*. \end{cases}$$  

(21)

Compared with the direct training with surrogate gradients using BPTT, our calibration methods here only fine-tune and backward on one layer which means that the spatial complexity is $O(T \cdot \max(n_i))$ rather than $O(T \cdot \sum n_i)$, where $n_i$ is the neurons number at the $l$th layer in ANNs. Moreover, it is much easier and more computationally efficient to converge by fine-tuning on a pretrained layer rather than retraining from scratch on a large network presented in surrogate gradient methods. Empirically, it is enough for the calibration with
several hundred of samples. The algorithm is illustrated in the pseudocode of Algorithm 1.

VI. EXPERIMENTS

In this section, we first evaluate the effectiveness and efficiency of the proposed method compared to other state-of-the-art conversion algorithms for image recognition tasks on CIFAR-10, CIFAR-100, and ImageNet datasets. Each component of the design is validated with a comprehensive ablation study. Subsequently, both sample efficiency and energy efficiency are examined in different time steps. Furthermore, we investigate spike-based object detection with the proposed two-stage pipeline in shortened time steps.

A. Implementation Details

1) Training of Source ANN: To obtain a high-performance source ANN trained with \( \sigma_0(x) = \text{clip}((1/\theta)x, 0, 1) \). In general, there are three approaches: 1) training an ANN with clipping function \( \sigma_0(x) \) directly as done in [31]; 2) training an ANN with ReLU function and then constrain the activations into \([0, 1]\) with weight-threshold balancing technique [29], [30], [32]; and 3) training an ANN with the rescaled clipping function \( \sigma_0(x) = \text{clip}(x, 0, \theta) [38] \) and then fusing the scale factor \( \theta \) into weights after training. In experiments, we adopt the first method for CIFAR-10. For CIFAR-100 and ImageNet, we use the third method to achieve higher performance.

2) Experimental Settings: For CIFAR-10 and CIFAR-100 datasets, we crop images into \( 32 \times 32 \) with the padding of 4 pixels and flip them horizontally at random. In addition, as done in [26] and [37], we adopt the data augmentation of Cutout [57] and AutoAugment [58] to increase the generalization ability of source ANNs. For ImageNet, we adopt the standard preprocessing pipeline and crop images into \( 224 \times 224 \).

Table I provides a comprehensive comparison of data augmentation, encoding, and readout techniques employed in the proposed approach and other best-performing methods.

For training source ANN, we initialize the learning rate as 0.1 and then update it through the SGD optimizer with a momentum of 0.9. Meanwhile, the weight decay is set to \( 5e^{-4} \) to regularize the network. In addition, the learning rate is set as \( 1e^{-4} \) further to finetune QC-ANN. For CIFAR-10, the probability of quantization noise \( p \) is set as 0.1 and 0.2 when \( T \leq 4 \). Otherwise, no noise is injected into QC-ANNs. The \( p \) is set as 0.1 across all time steps on ImageNet and CIFAR-100 datasets. For Stage II, we observed that the errors tend to vary across different layers. Therefore, to address this issue and facilitate layer-wise calibration with BPTT presented in Stage-II, we opted to use the Adam optimizer with an adaptive learning rate. The initial learning rate is configured as \( 5e^{-4} \) and \( 1e^{-4} \) for recognition and detection tasks, respectively. The weight decay is set as 0. Moreover, we also adopt layer-wise BPTT calibration on the initial potential for the ImageNet dataset. For energy analysis, we sample 1024 samples and calculate the mean value and standard deviation of synaptic operations (SOPs). Early stopping is also used to alleviate overfitting with a tolerance of 20 epochs.

3) Evaluation Metrics: For the task of object detection, we report the mean average precision (mAP) with different localization requirements. As an illustration, \( \text{AP}@0.75 \) denotes the mAP score which considers the predictions with a correct classification and localization IoU \( \geq 0.75 \) as the true positive. Furthermore, according to the size of the ground truth, the mAP is divided into that of small objects \( \text{AP}_S \), middle objects \( \text{AP}_M \), and large objects \( \text{AP}_L \).

B. Ablation Study

In this section, we conduct a series of ablation studies and show the proposed method reduces the conversion loss in a complementary manner. Specifically, we test ResNet-20 on CIFAR-10 from \( T = 2 \) to 128 under six conditions: Just Copy Weights, Only Stage-I, Only Stage-II, Stage-I + Stage-II, Stage-I + CC, and Stage-I + FC.

As shown in Fig. 7(a), both Only Stage-I and Only Stage-II improve the performance effectively, which is especially impressive when \( T \leq 32 \). In addition, Stage-I + Stage-II with dedicated optimization obtains the best results across all time steps which indicates the synergic effect of the two stages. Comparing the two stages individually, we find Stage-I optimizing the QE and CE achieves better results when \( T < 4 \). Otherwise, Stage-II optimizing RPE mainly obtains higher...
performance. It is interpretable as the quantization factor \( \theta/T \) and the QE naturally grow as \( T \) decreases. Therefore, the performance degradation stemmed from the QE plays a dominant role compared with RPE, when \( T \) is extremely low. Surprisingly, Only Stage-II only needs 8 time steps to match the performance of Stage-I + Stage-II, whereas Only Stage-I requires 32 time steps. It demonstrates that optimizing RPE is more efficient in low-latency conversion in general. So simply adopting layer-wise Stage-II with a handful of samples is enough for conversion on resource-constrained devices. Notably, there is a strange accuracy drop for Only Stage-I when \( T = 4 \). It may come from increasing RPE which is not optimized by Stage-I. As shown in the curve of Stage-I + Stage-II, the drop is eliminated by introducing Stage-II (minimizing RPE). This indeed supports the efficacy of minimizing RPE.

To understand the effect of each component in Stage-II further, we do coarse calibration (CC) and fine calibration separately after Stage-I [Fig. 7(b)]. In general, both components improve performance consistently. In particular, we find Stage-I + FC achieves extremely close results to Stage-I + Stage-II. And, it even outperforms Stage-I + Stage-II when \( T = 4 \). Nevertheless, CC still provides a lightweight solution without gradient computation and a proper initialization for fine calibration.

C. Trade-Off Between Accuracy and Inference Delay

We compare the proposed method with the state-of-the-art ANN–SNN conversion methods on CIFAR-10, CIFAR-100 (Table II), and ImageNet (Table III) datasets. Here, in order to better compare the performance with different methods, we provide reproduced experimental results under few time steps in the format of "A/B." In general, the proposed method achieves the best result nearly across all time steps. For ultralow latency \( T = 2 \) and \( T = 4 \), our method achieves promising performance improvements of 14.53% and 2.84% with ResNet-18 on the CIFAR-10 Dataset, respectively. Notably, SNNs perform better than ANNs when time step increases up to 128, as shown in Table II. The identified RPE in this work explains the underlying cause. The spike rate is used in conversion to approximate ANN activation. Typically, it is difficult to eliminate the nondeterministic component (i.e., RPE) of approximation error. Thus, RPE serves as noise when using the firing rate of SNNs to replace the ANN activations (8). The noise becomes the optimizing objective in Stage-II, and weights and initial potential are fine tuned against such noise, so that the RPE brings additional generalization ability to convert SNNs which their ANN counterparts do not have. The superiority seems to vanish when \( T \geq 8 \) compared with the most recent work [37]. However, when considering large-scale
TABLE III

| Method          | Architecture | ANN Acc. | $T = 4$ | $T = 8$ | $T = 16$ | $T = 32$ | $T = 64$ | $T = 128$ | $T = 256$ |
|-----------------|--------------|----------|---------|---------|----------|----------|----------|----------|----------|
| TSC [59]        |              | 70.64    | -       | -       | -        | -        | -        | 55.65    |
| QCPS [37]       |              | 74.32    | -       | -       | 59.35    | 69.37    | 72.35    | 73.15    | 73.37    |
| RMP [6]         |              | 74.54/70.64 | 0.10%  | 0.10%  | 0.10%    | 1.23%    | 32.12%   | 63.29%   | 71.20/55.65 |
| Norm-on-Shortcut [45] ResNet-34 |              | 74.54/70.88 | 0.13%  | 0.20%  | 0.70%    | 7.65%    | 48.67%   | 68.55%   | 72.75%   |
| TCL [38]        |              | 73.43/70.85 | 0.09%  | 0.10%  | 0.18%    | 1.00%    | 19.96%   | 93.00%   | 72.57%   |
| OPT [31]        |              | 73.43/75.66 | 0.10%  | 0.10%  | 0.15%    | 4.69/0.09 | 47.22/0.12 | 68.23/0.19 | 72.37/47.11 |
| CAP [26]        |              | 75.65/75.66 | 1.68%  | 23.62% | 52.72%   | 67.78/64.54 | 72.71/71.12 | 74.40/74.35 | 74.92/74.61 |

Ours ResNet-34 73.43 55.71 61.20 67.77 71.66 72.65 73.37 73.45

| Method          | Architecture | ANN Acc. | $T = 4$ | $T = 8$ | $T = 16$ | $T = 32$ | $T = 64$ | $T = 128$ | $T = 256$ |
|-----------------|--------------|----------|---------|---------|----------|----------|----------|----------|----------|
| TSC [59]        |              | 73.49    | -       | -       | -        | -        | -        | -        | -        |
| RMP [6]         |              | 74.53/73.49 | 0.08%  | 0.09%  | 0.13%    | 0.32%    | 9.53%    | 50.89%   | 68.78/48.32 |
| OPT [31]        |              | 74.88/73.56 | 0.10%  | 0.11%  | 0.21%    | 1.10/0.114 | 22.75/0.118 | 64.71/0.122 | 72.81/1.81 |
| CAP [26]        |              | 74.75/73.56 | 5.65%  | 25.70% | 56.25%   | 66.0/63.64 | 70.33/70.69 | 72.43/73.32 | 73.54/74.23 |
| SNM [39]        |              | 73.18    | -       | -       | -        | 64.78    | 71.50    | 72.86    | -        |
| Burst [41]      |              | 74.27    | -       | -       | 50.97    | 68.47    | 72.85    | 73.97    | 74.22    |
| QCPS [37]       |              | 74.29    | -       | 6.25%   | 36.02    | 64.70    | 72.47    | 74.24    | 74.62    |
| OPI [48]        |              | 74.85    | -       | 59.95%  | 62.51    | 70.13    | 73.44    | 74.68    | 74.93    |

Ours VGG-16 74.88 59.95 62.51 70.13 73.44 74.68 74.93 74.94

datasets like ImageNet, the proposed method still improves the state-of-the-art of accuracy-delay trade-off sustainedly as explicated in Table III. For instance, the inference delay is shrunk by 4× into four time steps on ImageNet with VGG16 architecture while still obtaining significant performance improvements (8.98% top-1 at least, 59.95% versus 50.97%) compared with the existing literature. Notably, as we do not apply the CollorJitter data augmentation as done in [26] and [48], the result of source ANN on ImageNet and the best SNN ($T \geq 256$) is slightly lower than the best results reported [26]. Even though, our method still outperforms previous works and achieves 74.68% top-1 accuracy under relatively long time steps ($T = 64$). A full comparison of data augmentation methods, encoding, and decoding methods with the best-performing competitors is provided in Table I.

It is worth noting that different methods use different source ANN activation functions (e.g., clipping function in TCL [38]) and different training methods (e.g., customized auxiliary loss for RNL layer in [46]). Hence, achieving exactly the same ANN performance, even with the same data preprocessor and optimizer is challenging. To further eliminate the influence of the source ANN, we assess the relative performance losses $\Delta_{\text{ANN}} = (\text{Acc}_{\text{ANN}} - \text{Acc}_{\text{SNN}})/\text{Acc}_{\text{ANN}}$ at $T = 32$, $T = 64$, and $T = 128$, as demonstrated in Table IV. In general, the proposed method significantly reduces the conversion loss and enhances the performance of target SNNs. Moreover, as shown in Table V, our method is also competitive compared with the state-of-the-art direct training method characterized by temporal credit assignment under only four time steps on both CIFAR datasets. All those results illustrate the effectiveness of optimizing threefold errors simultaneously.

D. Effect of Different Sample Numbers

As shown in Table VI, we investigate the impact of calibration sample numbers on CIFAR-10 using ResNet-34 and VGG-16 architectures. The initial line shows the baseline results without calibration. Interestingly, a small number of calibration samples yield significant enhancements. For instance, calibration on just 32 samples elevates the performance of spiking ResNet-18 by 25.42% when executed under four time steps, demonstrating the sample efficiency of the proposed calibration approach, particularly for low-latency conversion. It is worth mentioning that inadequate sample numbers may cause a certain decline in performance due to the randomness of BPTT calibration. In our experience, using 64 calibration samples leads to considerable performance improvement under low inference steps. During actual deployment, an appropriate calibration number can be obtained further by analyzing the regression curve of performance gains with respect to calibration samples.

Authorized licensed use limited to the terms of the applicable license agreement with IEEE. Restrictions apply.
TABLE V

COMPARISON WITH DIRECT TRAINING WITH SPIKE-BASED BACKPROPAGATION

| Method     | Architecture | Time Steps | Accuracy (%) |
|------------|--------------|-----------|--------------|
| CIFAR-10   |              |           |              |
| BRP [59] †| 2D-CSNN      | -         | 57.08        |
| STBP [54]  | CIFARNNet    | 12        | 90.53        |
| Hybrid [60]| ResNet-20    | 250       | 92.22        |
| DIET-SNN [61]| ResNet-20    | 5         | 91.78        |
| TSSL [19]  | CIFARNNet    | 5         | 91.41        |
| STBP-tdBN [62]| ResNet-4     | 4         | 92.92        |
| Ours       | ResNet-20    | 4         | 93.27        |
|            | VGG-16       |           | 94.06        |

CIFAR-100

| Method     | Architecture | Time Steps | Accuracy (%) |
|------------|--------------|-----------|--------------|
| Hybrid [60]| VGG-11       | 125       | 67.87        |
| DIET-SNN [61]| VGG-16      | 5         | 69.67        |
| STBP-tdBN [62]| ResNet-19    | 4         | 70.86        |
| Ours       | ResNet-18    | 4         | 75.98        |
|            | VGG-16       |           | 70.08        |

The symbol † indicates that the method incorporates a biologically plausible pseudo-backpropagation approach, leveraging spike-timing-dependent plasticity (STDP).

TABLE VI

EFFECT OF THE NUMBER N OF CALIBRATION SAMPLES, THE NUMBER IN PARENTHESES IMPLIES THE ACCURACY OF THE SOURCE ANN

| # samples | ResNet-18 (96.41) | VGG-16 (95.73) |
|-----------|-------------------|----------------|
|           | T = 2             | T = 4          | T = 8          |
|           | T = 2             | T = 4          | T = 8          |
| 0         | 76.01             | 66.52          | 84.17          |
| 2         | 75.55             | 32.20          | 74.47          |
| 4         | 83.46             | 80.26          | 90.42          |
| 8         | 85.79             | 87.67          | 93.21          |
| 16        | 87.89             | 90.47          | 94.31          |
| 32        | 88.75             | 91.94          | 94.89          |
| 64        | 89.56             | 92.40          | 94.85          |
| 128       | 89.94             | 92.18          | 95.07          |
| 256       | 89.51             | 92.77          | 95.29          |
| 512       | 89.89             | 92.96          | 95.31          |
| 1024      | 90.49             | 93.23          | 95.35          |

E. Energy Efficiency and Sparsity

In this section, we count the total SOPs to estimate the computation overhead of SNN compared with their ANN counterparts [9]. Specifically, synaptic operations in SNNs exhibit variability influenced by spike sparsity with sparse accumulation (AC). In contrast, synaptic operations involving multiplication and accumulation (MAC) in ANNs remain constant within a defined network structure. Here, we measure 32-bit float-point AC and MAC by $\alpha_{AC} = 0.9$ pJ and $\alpha_{MAC} = 4.6$ pJ per operation individually as done in [63]. Notably, the estimation is extremely conservative and the energy consumption of SNNs on specified hardware design can be reduced by $12 \times$ to $77 \text{ fJ/SOP}$ [64].

From Table VII, SNN is more SOP efficient with sparse spike communications than the ANN in the same ResNet-18 architecture. The energy efficiency is more significant considering the operation degrading from MAC to AC. For instance, the conversion is nearly lossless (0.38% accuracy loss) when $T = 8$ with only 16% energy consumption compared with the ANN counterpart. For edge devices with stringent power requirements, the energy consumption is downgraded to 140 $\mu$J further (15.7 $\times$ energy saving with 73%+ performance on CIFAR-100). We further compare the energy consumption with the state-of-the-art methods [26], [37], [48]. In this analysis, we ensured that different methods were reproduced under the same conditions, and we used the same set of 1024 samples to estimate the power consumption. As shown in Table VII, although our approach consumes slightly more energy compared with OPI [48], it is justified by the significant performance gains achieved at few time steps. Specifically, our method reduces the relative performance loss ($\Delta_{Acc}$) by 65.89% under two time steps.

TABLE VII

COMPARING THE ENERGY COST UNDER DIFFERENT TIME STEPS ON CIFAR-100. THE NUMBERS IN PARENTHESES REPRESENT RELATIVE ACCURACY LOSSES DURING CONVERSION

| Metrics       | Method     | ANN | T = 2 | Accuracy (%) |
|---------------|------------|-----|-------|--------------|
|                | OPI [48]   |     |       | 79.43        |
|                | CAP [26]   |     |       | 78.89        |
|                | QCFS [37]  |     |       | 77.15        |
|                | Ours       |     |       | 72.92        |
|                | GSOP ($\times$ 1e-3) | |       | Ours: 72.92 |
|                | OPI [48]   |     |       | 48.02        |
|                | CAP [26]   |     |       | 48.02        |
|                | QCFS [37]  |     |       | 48.02        |
|                | Ours       |     |       | 48.02        |
|                | Energy ($\mu$J) |    |       | 75.1 $\pm$ 3.6 |
|                | OPI [48]   |     |       | 220.87       |
|                | CAP [26]   |     |       | 220.87       |
|                | QCFS [37]  |     |       | 220.87       |
|                | Ours       |     |       | 220.87       |
|                | Energy Ratio (%) | |       | Ours: 75.1 $\pm$ 3.6 |
|                | OPI [48]   |     |       | 100.0 $\pm$ 16.0 |
|                | CAP [26]   |     |       | 100.0 $\pm$ 16.0 |
|                | QCFS [37]  |     |       | 100.0 $\pm$ 16.0 |
|                | Ours       |     |       | 100.0 $\pm$ 16.0 |

TABLE VIII

SCALING THE PROPOSED METHOD TO ResNeXt29-2 $	imes$ 64D AND MobileNetV2 ARCHITECTURES

| Method       | VGG-16 (95.62) | ResNet-18 (96.81) |
|--------------|---------------|------------------|
| T = 4        | T = 8         | T = 4            | T = 8       |
| Baseline     | 27.60         | 49.74            | 15.59       | 29.36       |
| Stage-I      | 79.79         | 92.43            | 26.16       | 54.17       |
| Stage-II     | 82.96         | 83.92            | 60.42       | 85.63       |
| Stage-I + Stage-II | 90.02       | 94.33            | 75.64       | 91.49       |

TABLE IX

SCALING THE PROPOSED METHOD TO Leaky ReLU

| Method       | VGG-16 (95.62) | ResNet-18 (96.81) |
|--------------|---------------|------------------|
| T = 4        | T = 8         | T = 4            | T = 8       |
| Baseline     | 47.77         | 71.80            | 71.46       | 87.49       |
| Stage-I      | 78.83         | 82.47            | 88.27       | 91.55       |
| Stage-II     | 86.02         | 91.48            | 85.44       | 92.52       |
| Stage-I + Stage-II | 91.76     | 93.38            | 91.23       | 93.93       |

F. Extensibility for Complex Structures

To demonstrate that our method can be applied to more complex models, we conduct ablation studies on the complex
 improving the performance of spike-based object detectors through the proposed method

| Methods                        | Time Step | AP@0.50:0.95 | AP@0.50 | AP@0.75 | AS | AM | AL |
|-------------------------------|-----------|--------------|---------|---------|----|----|----|
| Spiking YOLO [32]             | 20        | 0.2          | 0.7     | 0.1     | 0.2 | 0.5 | 0.1 |
| Stage-I w/o Stage-II          | 20        | 7.3          | 19.7    | 3.1     | 3.6 | 6.8 | 8.7 |
| Stage-I + Stage-II            | 20        | 28.5         | 64.1    | 19.5    | 6.2 | 16.2 | 33.5 |
| Spiking YOLO [32]             | 40        | 7.9          | 22.3    | 3.1     | 2.9 | 5.2 | 10.3 |
| Stage-I w/o Stage-II          | 40        | 20.1         | 48.5    | 11.6    | 4.6 | 13.0 | 24.4 |
| Stage-I + Stage-II            | 40        | 30.7         | 66.4    | 22.5    | 7.2 | 17.8 | 36.1 |
| Spiking YOLO [32]             | 2000      | 26.9         | 59.9    | 19.1    | 6.7 | 15.5 | 31.6 |
| Spiking YOLO [32]             | 60        | 16.5         | 41.6    | 8.2     | 5.3 | 9.2  | 20.1 |
| Stage-I w/o Stage-II          | 60        | 25.7         | 58.7    | 16.6    | 5.7 | 15.6 | 30.6 |
| Stage-I + Stage-II            | 60        | 31.2         | 66.8    | 23.2    | 7.1 | 17.9 | 36.7 |

Fig. 8. Comparison of detection results on a single sample with increasing time steps.

The results of this adaptation are showcased in Table IX, where we observe a significant improvement in the performance of the converted SNN by stages. In addition, the latency is dramatically reduced to eight time steps, marking a substantial advancement compared to the typical state-of-the-art method [40] for conversion with the leaky ReLU activation function, which reports latencies in the hundreds of time steps.

G. Improvements on Object Detection

In this section, we investigate the effect of the proposed method on a more challenging detection task that involves the coordinate regression of bounding boxes and so requires higher numerical precision than object recognition to approximate the accurate coordinates. For a fair comparison, we implement the well-known spike-based object detector (Spiking YOLO [32]) in the same architecture which achieves better performance than that reported in the original paper. As depicted in Table X, the proposed method improves continuously the performance of spike-based detection across different object scales and time steps on the PASCAL VOC dataset. Meanwhile, the detection delay is reduced significantly by at least 25 times to 40 time steps for nearly lossless conversion while Spiking YOLO typically needs thousands of time steps. Moreover, Stage-I + Stage-II under few time steps (e.g., 20 time steps) outperforms Stage-I w/o Stage-II by a large margin, especially on the large and middle object detection (33.5% versus 8.7% and 16.2% versus 6.8%). It further illustrates the effect of Stage-II and the importance of optimizing RPE in addition to QE and CE.

Furthermore, we show the detection results on a sample image with increasing time steps (Fig. 8). The far left gives the result of ANN-based Tiny YOLO. The top row displays the detection results of Spiking YOLO [32]. In this example, no object can be detected until the time step grows to 100.
By optimizing QE and CE, Stage-I shortens the detection delay to 40 time steps although the numerical precision is still not high. The delay is further reduced to 20 time steps and higher detection accuracy is attained by further optimizing RPE in Stage-II. We also draw the detection results of each class of objects, as shown in Fig. 9.

VII. CONCLUSION AND DISCUSSION

In this work, instead of merely optimizing quantization and clipping error as in the previous works, this work first explicitly identifies the errors from residual potential which exhibit a more substantial role in low-latency conversion. By including an additional fine-tuning stage to specifically minimize this source of error, this study presents a novel two-stage converting strategy and obtains high-accuracy and low-latency SNNs. The introduced calibration mechanism considering temporal dynamics in SNNs effectively smooths the information loss of knowledge transfer between ANNs and SNNs. Experiments on large-scale recognition and detection tasks demonstrate our method improves both performance and latency greatly with attractive power conservation compared with ANNs, which would facilitate the future application of SNNs on resource-constrained edge devices.

The proposed conversion method has limitations in processing time-domain data with RNN-like and attention-based architectures. In future work, we expect to adapt the concept of temporal calibration to other types of network structures, such as LSTM and transformer. In addition, we consider conversion methods based on temporal coding with BPTT calibration, aiming for ultimate energy efficiency and minimized latency. It is important to note that the proposed conversion method is not applicable to online or on-chip learning, as the conversion algorithms generally do not target online or on-chip learning with neuromorphic uses. Instead, they are widely applicable in efficient inference with lightweight parameter tuning. Therefore, how to realize transfer learning from ANNs to SNNs in a flexible, local, and online way is a worthwhile avenue for future research, especially in the context of deployment on neuromorphic chips.

REFERENCES

[1] A. L. Hodgkin and A. F. Huxley, “A quantitative description of membrane current and its application to conduction and excitation in nerve,” J. Physiol., vol. 117, no. 4, p. 500, 1952.

[2] W. Gerstner, R. Ritz, and J. L. van Hemmen, “Why spikes? Hebbian learning and retrieval of time-resolved excitation patterns,” Biol. Cybern., vol. 69, nos. 5–6, pp. 503–515, Oct. 1993.

[3] W. Maass, “Networks of spiking neurons: The third generation of neural network models,” Neural Netw., vol. 10, no. 9, pp. 1659–1671, Dec. 1997.

[4] K. Roy, A. Jaiswal, and P. Panda, “Towards spike-based machine intelligence with neuromorphic computing,” Nature, vol. 575, no. 7784, pp. 607–617, Nov. 2019.

[5] D. V. Christensen et al., “2022 Roadmap on neuromorphic computing and engineering,” Neurocomput. Eng., vol. 2, no. 2, 2022, Art. no. 022501.

[6] B. Han, G. Srinivasan, and K. Roy, “RMP-SNN: Residual membrane potential neuron for enabling deeper high-accuracy and low-latency spiking neural network,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2020, pp. 13555–13564.

[7] E. Painkras et al., “SpinNNaker: A 1-W 18-core system-on-chip for massively-parallel neural network simulation,” IEEE J. Solid-State Circuits, vol. 48, no. 8, pp. 1943–1953, Aug. 2013.

[8] M. Davies et al., “Advancing neuromorphic computing with Loihi: A survey of results and outlook,” Proc. IEEE, vol. 109, no. 5, pp. 911–934, May 2021.

[9] P. A. Merolla et al., “A million-spiking-neuron integrated circuit with a scalable communications network and interface,” Science, vol. 345, no. 6197, pp. 668–673, Aug. 2014.

[10] L. Deng et al., “Rethinking the performance comparison between SNNS and ANNS,” Neural Netw., vol. 121, pp. 294–307, Jan. 2020.

[11] Y. Wu, L. Deng, G. Li, J. Zhu, and L. Shi, “Spatio-temporal backpropagation for training high-performance spiking neural networks,” Frontiers Neurosci., vol. 12, p. 531, May 2018.

[12] S. B. Shrestha and G. Orchard, “SLAYER: Spike layer error reassignment in time,” in Proc. Adv. Neural Inf. Process. Syst., vol. 31, 2018, pp. 1419–1428.

[13] E. O. Nefci, H. Mostafa, and F. Zenke, “Surrogate gradient learning in spiking neural networks: Bringing the power of gradient-based optimization to spiking neural networks,” IEEE Signal Process. Mag., vol. 36, no. 6, pp. 51–63, Nov. 2019.

[14] P. Gu, R. Xiao, G. Pan, and H. Tang, “STCA: Spatio-temporal credit assignment with delay in deep spiking neural networks,” in Proc. IJCAI, Aug. 2019, pp. 1366–1372.

[15] W. Fang, Z. Yu, Y. Chen, T. Huang, T. Masquelier, and Y. Tian, “Deep residual learning in spiking neural networks,” in Proc. Adv. Neural Inf. Process. Syst., vol. 34, 2021, pp. 21056–21069.

[16] S. M. Bothe, J. N. Kok, and J. A. La Poutré, “SpikeProp: Backpropagation for networks of spiking neurons,” in Proc. ESANN, Bruges, Belgium, vol. 48, Feb. 2000, pp. 419–424.

[17] J. Kim, K. Kim, and J.-I. Kim, “Unifying activation- and timing-based learning rules for spiking neural networks,” in Proc. Adv. Neural Inf. Process. Syst., vol. 33, 2020, pp. 19534–19544.

[18] Q. Yu, H. Li, and K. C. Tan, “Spike timing or rate? Neurons learn to make decisions for both through threshold-driven plasticity,” IEEE Trans. Cybern., vol. 49, no. 6, pp. 2178–2189, Jun. 2019.

[19] W. Zhang and P. Li, “Temporal spike sequence learning via backpropagation for deep spiking neural networks,” in Proc. Adv. Neural Inf. Process. Syst., vol. 33, 2020, pp. 12022–12033.

[20] A. Taherkhani, A. Belatreche, Y. Li, and L. P. Maguire, “A supervised learning algorithm for learning precise timing of multiple spikes in multilayer spiking neural networks,” IEEE Trans. Neural Netw. Learn. Syst., vol. 29, no. 11, pp. 5394–5407, Nov. 2018.

[21] M. Zhang et al., “Rectified linear postsynaptic potential function for backpropagation in deep spiking neural networks,” IEEE Trans. Neural Netw. Learn. Syst., vol. 33, no. 5, pp. 1947–1958, May 2022.

[22] J.-P. Pfister, T. Toyoizumi, D. Barber, and W. Gerstner, “Optimal spike-timing-dependent plasticity for precise action potential firing in supervised learning,” Neural Comput., vol. 18, no. 6, pp. 1318–1348, Jun. 2006.

[23] B. Gardner, I. Sporea, and A. Grüning, “Learning spatiotemporally encoded pattern transformations in structured spiking neural networks,” Neural Comput., vol. 27, no. 12, pp. 2548–2586, Dec. 2015.

[24] C. Ma, R. Yan, Z. Yu, and Q. Yu, “Deep spike learning with local classifiers,” IEEE Trans. Cybern., vol. 53, no. 5, pp. 3363–3375, May 2023.

[25] Q. Zhan, G. Liu, X. Xie, G. Sun, and H. Tang, “Effective transfer learning algorithm in spiking neural networks,” IEEE Trans. Cybern., vol. 52, no. 12, pp. 13323–13335, Dec. 2022.

[26] Y. Li, S. Deng, X. Dong, R. Gong, and S. Gu, “A free lunch from ANN: Towards efficient, accurate spiking neural networks calibration,” in Proc. Int. Conf. Mach. Learn., 2021, pp. 6316–6325.

[27] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, “ImageNet: A large-scale hierarchical image database,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., Jun. 2009, pp. 248–255.

[28] J. Wu, Y. Chua, M. Zhang, G. Li, H. Li, and K. C. Tan, “A tandem learning rule for effective training and rapid inference of deep spiking neural networks,” IEEE Trans. Neural Netw. Learn. Syst., vol. 34, no. 1, pp. 446–460, Jan. 2023.
S. Deng and S. Gu, “Optimal conversion of conventional artificial neural networks to spiking neural networks,” in Proc. Int. Conf. Learn. Represent., 2021.

S. Kim, S. Park, B. Na, and S. Yoon, “Spike-yolo: Spiking neural network for energy-efficient object detection,” in Proc. AAAI Conf. Artif. Intell., Apr. 2020, vol. 34, no. 7, pp. 11270–11277.

W. Tan, D. Patel, and R. Koznza, “Strategy and benchmark for converting deep Q-networks to event-driven spiking neural networks,” in Proc. AAAI Conf. Artif. Intell., 2021, vol. 35, no. 11, pp. 9816–9824.

Y. Luo et al., “SiamesNN: Siamese spiking neural networks for energy-efficient object tracking,” in Proc. Int. Conf. Artif. Neural Netw. Chern, Switzerland: Springer, 2021, pp. 182–194.

Y. Kim, Y. Venkatesha, and P. Panda, PrivateSNN: Privacy-preserving spiking neural networks,” in Proc. AAAI Conf. Artif. Intell., 2022, vol. 36, no. 1, pp. 1192–1199.

Z. Yan, J. Zhou, and W.-F. Wong, “Near lossless transfer learning for spiking neural networks,” in Proc. AAAI Conf. Artif. Intell., 2021, vol. 35, no. 12, pp. 10577–10584.

T. Bu, W. Fang, J. Ding, P. Dai, Z. Yu, and T. Huang, “Optimal ANN-SNN conversion for high-accuracy and ultra-low-latency spiking neural networks,” in Proc. Int. Conf. Learn. Represent., 2022.

N.-D. Ho and I.-J. Chang, “Tcl: An ANN-to-SNN conversion with trainable clipping layers,” in Proc. 58th ACM/IEEE Design Autom. Conf. (DAC), Dec. 2021, pp. 793–798.

Y. Gao et al., “Solving neuron with memory: Towards simple, accurate and high-efficient ANN-SNN conversion,” in Proc. 31st Int. Joint Conf. Artif. Intell., Jul. 2022, pp. 2501–2508.

Q. Yu, C. Ma, S. Song, G. Zhang, J. Dang, and K. C. Tan, “Constructing accurate and efficient deep spiking neural networks with double-threshold and augmented schemes,” IEEE Trans. Neural Netw. Learn. Syst., vol. 33, no. 4, pp. 1764–1766, Apr. 2022.

Y. Li and Y. Zeng, “Efficient and accurate conversion of spiking neural network with burst spikes,” 2022, arXiv:2204.13271.

Y. Cao, Y. Chen, and D. Khasla, “Spiking deep convolutional neural networks for energy-efficient object recognition,” Int. J. Comput. Vis., vol. 113, no. 1, pp. 54–66, May 2015.

A. Sengupta, Y. Ye, R. Wang, C. Liu, and K. Roy, “Going deeper in spiking neuronal networks: VGG and residual architectures,” Frontiers Neurosci., vol. 13, p. 95, Mar. 2019.

Y. Xu, H. Tang, J. Xing, and H. Li, “Spike trains encoding and threshold rescaling method for deep spiking neural networks,” in Proc. IEEE Symp. Ser. Comput. Intell. (SSCI), Nov. 2017, pp. 1762–1768.

Y. Hu, H. Tang, and G. Pan, “Spike deep residual networks,” IEEE Trans. Neural Netw. Learn. Syst., vol. 34, no. 8, pp. 5200–5205, Aug. 2023.

J. Ding, Z. Yu, Y. Tian, and T. Huang, “Optimal ANN-SNN conversion for fast and accurate inference in deep spiking neural networks,” in Proc. IJCAI, Aug. 2021, pp. 2328–2336.

Y. Wang, Y. Xu, R. Yan, and H. Tang, “Deep spiking neural networks with binary weights for object recognition,” IEEE Trans. Cognit. Develop. Syst., vol. 13, no. 3, pp. 514–523, Sep. 2021.

T. Bu, J. Ding, Z. Yu, and T. Huang, “Optimized potential initialization for low-latency spiking neural networks,” in Proc. AAAI Conf. Artif. Intell., 2022, pp. 11–20.

C. Stockl and W. Maass, “Optimized spiking neurons can classify images with high accuracy through temporal coding with two spikes,” Nature Mach. Intell., vol. 3, no. 3, pp. 230–238, Mar. 2021.

B. Han and K. Roy, “Deep spiking neural network: Energy efficiency through time based coding,” in Proc. Eur. Conf. Comput. Vis. Cham, Switzerland: Springer, 2020, pp. 388–404.

S. Hwang et al., “Low-latency spiking neural networks using precharged membrane potential and delayed evaluation,” Frontiers Neurosci., vol. 15, Feb. 2021, Art. no. 629000.

B. Jacob et al., “Quantization and training of neural networks for efficient integer-arithmetic-only inference,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit., Jun. 2018, pp. 2704–2713.
Shuang Lian received the B.E. degree in applied mathematics and the M.S. degree in applied statistics from the School of Mathematical Sciences, Dalian University of Technology, Dalian, China, in 2017 and 2019, respectively. He is currently pursuing the Ph.D. degree with the College of Computer Science and Technology, Zhejiang University, Hangzhou, China.

His research interests include neuromorphic computing and learning algorithms in spiking neural networks (SNNs).

Xiaoxin Cui (Member, IEEE) received the B.Sc. degree in cybernetion from Beihang University, Beijing, China, in 2002, and the Ph.D. degree in microelectronic from Peking University, Beijing, in 2007.

She is currently a Professor with the School of Integrated Circuits, Peking University. Her research interests include neuromorphic computing and neuromorphic computing chips, energy-efficient AI accelerators, and hardware security.

Rui Yan (Member, IEEE) received the B.E. and M.S. degrees from the Department of Mathematics, Sichuan University, Chengdu, China, in 1998 and 2001, respectively, and the Ph.D. degree from the Department of Electrical and Computer Engineering, National University of Singapore, Singapore, in 2006.

She was a Post-Doctoral Research Fellow with The University of Queensland, Brisbane, QLD, Australia, from 2006 to 2008, and a Research Scientist with the Institute for Infocomm Research, A*STAR, Singapore, from 2009 to 2014. She is currently a Professor with the College of Computer Science, Zhejiang University of Technology, Hangzhou, China. Her research interests include intelligent robots, brain-inspired computing, and cognitive systems.

Huajin Tang (Senior Member, IEEE) received the B.E. degree from Zhejiang University, Hangzhou, China, in 1998, the M.E. degree from Shanghai Jiao Tong University, Shanghai, China, in 2001, and the Ph.D. degree from the National University of Singapore, Singapore, in 2005.

He was a System Engineer with ST Microelectronics, Singapore, from 2004 to 2006. From 2006 to 2008, he was a Post-Doctoral Fellow with the Queensland Brain Institute, The University of Queensland, Brisbane, QLD, Australia. Since 2008, he has been the Head of the Robotic Cognition Laboratory, Institute for Infocomm Research, A*STAR, Singapore. He is currently a Professor with the College of Computer Science and Technology, Zhejiang University. His research interests include neuromorphic computing and robotic cognition.

Dr. Tang was a recipient of the 2016 IEEE Outstanding IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS (TNNLS) Paper Award, 2019 IEEE Computational Intelligence Magazine Outstanding Paper Award, and the 2023 Neural Networks Best Paper Award. He was an Associate Editor of IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS, Frontiers in Neuroromorph Engineering, and Neural Networks. He is an EIC of IEEE TRANSACTIONS ON COGNITIVE AND DEVELOPMENTAL SYSTEMS, and also a Board of Governor Member of the International Neural Networks Society.