Are Conventional SNNs Really Efficient? A Perspective from Network Quantization

Guobin Shen[1,2], Dongcheng Zhao[1], Tenglong Li[1,3], Jindong Li[1,3] Yi Zeng[1,2,3,†]
[1] BrainCog Lab, CASIA, [2] School of Future Technology, UCAS, [3] School of Artificial Intelligence, UCAS
{shenguobin2021, zhaodongcheng2016, litenglong2023, lijindong2022, yi.zeng}@ia.ac.cn

Abstract

Spiking Neural Networks (SNNs) have been widely praised for their high energy efficiency and immense potential. However, comprehensive research that critically contrasts and correlates SNNs with quantized Artificial Neural Networks (ANNs) remains scant, often leading to skewed comparisons lacking fairness towards ANNs. This paper introduces a unified perspective, illustrating that the time steps in SNNs and quantized bit-widths of activation values present analogous representations. Building on this, we present a more pragmatic and rational approach to estimating the energy consumption of SNNs. Diverging from the conventional Synaptic Operations (SynOps), we champion the "Bit Budget" concept. This notion permits an intricate discourse on strategically allocating computational and storage resources between weights, activation values, and temporal steps under stringent hardware constraints. Guided by the Bit Budget paradigm, we discern that pivoting efforts towards spike patterns and weight quantization, rather than temporal attributes, elicits profound implications for model performance. Utilizing the Bit Budget for holistic design consideration of SNNs elevates model performance across diverse data types, encompassing static imagery and neuromorphic datasets. Our revelations bridge the theoretical chasm between SNNs and quantized ANNs and illuminate a pragmatic trajectory for future endeavors in energy-efficient neural computations.

1. Introduction

Spiking Neural Networks (SNNs) [24], rooted in a computational framework that draws inspiration from biological neural processes, present a compelling counterpoint to the established paradigms of traditional Artificial Neural Networks (ANNs). This modality signifies a substantial divergence from the analog and continuous activation regimes that characterize ANNs. Predicated on an event-driven operational principle, SNNs are inherently poised for reduced energy consumption, a trait that is instrumental in the ongoing evolution of neuromorphic computing platforms [17]. Building on the energy-efficient principles inherent to SNNs, pioneering developments in neuromorphic chips like TrueNorth [25] and Loihi [4] have underscored their significant potential for reducing energy consumption in comparison with traditional computing approaches. These platforms capitalize on the efficiency of SNNs to offer optimized solutions for energy-constrained applications, advancing the frontiers of computing technology.

Despite the inherently low power consumption of SNNs, their deployment in edge computing devices such as mobile phones and wearable technology remains notably infrequent. Despite progress in neuromorphic technology, modern SNN configurations remain deeply rooted in the methodologies of deep ANNs. These SNN designs typically rely on optimistic energy usage projections that overlook the tangible overheads associated with deploying hardware in real-world environments [18]. This approach can distort the comparison with traditional ANNs, especially when such ANNs have been refined for energy-efficient operation. To address the severe energy and computational constraints, the traditional machine learning domain has introduced a suite of model optimization techniques, including quantization [19, 20, 22] and network pruning [9, 10, 30]. These methods effectively reduce the computational and memory requirements. Such adaptability ensures that sophisticated neural models can be deployed even in resource-limited settings with minimal compromise on performance. Conventionally, SNNs are often compared to ANNs that have not undergone similar optimizations [28, 31, 36, 44]. Although such comparisons may highlight the energy efficiency of SNNs, they do not fairly consider the significant advancements in ANN energy-efficient optimization. Consequently, this prevalent comparative framework fails to substantiate the computational efficacy of SNNs in a comprehensive manner and does not adequately acknowledge the considerable enhancements achieved through contemporary ANN inference optimization methodologies.

[†]Corresponding Author.

In particular, the advent of surrogate gradient methods [2] has precipitated new advancements in the field of deep SNNs, pivoting the focus towards enhancing network performance concurrently with the reduction of network latency [31, 37, 41]. Incorporating advanced methodologies has precipitated a notable reduction in latency within deep SNNs, culminating in scenarios where the processing is streamlined to necessitate a mere one or two time steps. This observation raises a compelling inquiry: Does an SNN, functioning within the confines of a single time step, essentially mirror the characteristics of an ANN wherein the activation is quantized? Our analysis scrutinizes the efficiency of SNNs through the prism of neural network quantization, allowing us to elucidate the nuanced discrepancies between SNNs and their quantized ANN counterparts. We undertake a thorough and methodical examination of the computational efficacy of SNNs, with an emphasis on their practical application within real-world contexts, transcending beyond the confines of theoretical computational advantages. Our contribution can be summarized as follows:

- We introduce a unified framework for quantitatively analyzing SNNs and ANNs, demonstrating that an SNN with simulation step T is comparable to a T-bit quantized ANN. This emphasizes the association between SNNs and ANNs in terms of computational complexity.
- We define the "Bit Budget" to measure synaptic operation complexity and present two metrics, S-ACE and NS-ACE, to assess computational demand across hardware platforms. Time steps, spike patterns, and weights are taken into account for the computational overhead. These metrics inform the development of the strategy and emphasize the significant impact of model weight quantification on computation and storage
- Our validation across varied neural tasks and architectures highlights the potential for our findings to bridge SNN research with broader deep learning advances and guide future explorations in the field.

2. Related Work

Spiking neural networks, heralded as the third generation of neural networks, are engineered to emulate the intricate information-processing mechanism of the human brain. Maass [24] was seminal in delineating the contrast between SNNs and their ANN counterparts, explicitly highlighting the former's proficiency in discrete-time, event-driven processing. This paradigm shift was further catalyzed by the work [31, 34, 35, 37–39] and innovative Spiking Transformer architectures [36, 44]. These models are increasingly becoming cornerstones in domains like image recognition, natural language processing, and robotics control [32, 42, 45], underscoring the SNNs' inherent ability to execute computationally demanding tasks in an energy-efficient manner, presenting a sustainable alternative to conventional ANNs.

Optimization of neural network efficiency, particularly in terms of computational overhead and resource utilization, is a critical aspect of advancing deep learning technology. Notable techniques such as network pruning and quantization have dramatically diminished the computational burden and storage requirements traditionally associated with neural networks. Pruning strategies streamline network architectures by systematically eliminating superfluous connections and neurons. Han et al. [11] illustrated the over-parameterization in deep learning models and optimized them through pruning. This approach evolved, giving rise to structured [15, 21] and unstructured pruning methods [12]. Quantization, on the other hand, concentrates on minimizing the precision of the numerical representation of neural weights and activations, substantially saving on both storage and computational intensity. Zhou et al. [43] showcased the potential of quantized neural networks. Extreme cases, like Binary Neural Networks (BNNs), which only use single-bit representations, have also been explored [3]. Thereon, the domain has witnessed extensive research, focusing on balancing quantization and performance [26].

Given the intertwined trajectories of SNNs, neuromorphic hardware, and ANNs, there is an evident need for a comprehensive evaluation metric that accounts for practical hardware constraints. However, current measures of the computational complexity of SNNs using the number of Synaptic Operations (SOPs) [13] only take into account the effects of firing rate and time step, making it challenging to meet this burden. Our research suggests that SNNs and ANNs can be equated to circuit complexity. Recognizing this equivalence provides an avenue to analyze SNN energy consumption in line with ANNs, necessitating an inclusive energy-efficiency evaluation approach. Embracing techniques from ANNs, especially quantization can further amplify the energy-saving potential of SNNs. Informed by this holistic perspective, we introduce a universal framework. Our approach delves deep into the facets of weight precision, activation function choices, and the unique temporal resolution aspect of SNNs. This framework is designed to bridge theoretical postulations with practical implementations, championing both energy efficiency and computational excellence.

3. Method

3.1. T-step SNNs and T-bit Quantized ANNs

Both SNNs and Quantized ANNs (QANNs) enhance the efficiency of network inference through discrete representations. However, literature rarely discusses the differences and connections between them. We analyze both from a unified perspective.

An SNN layer operates across three temporal phases,

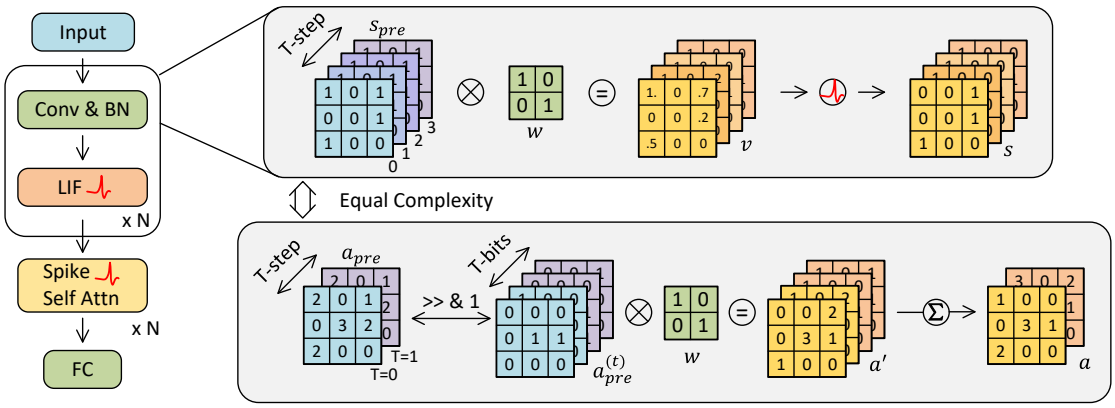


Figure 1. With the same number of feature bits, SNNs and quantized ANNs have the same complexity of representation.

delineated in Eq. 1. The input spikes (s_{pre}) from the antecedent layer are weighted and transformed into currents that modulate the neuron's membrane potential (v_t). If this potential exceeds a certain threshold (v_{th}), it resets after a spike, encapsulated by the Heaviside step function ($H(\cdot)$), reflecting the all-or-none nature of neuronal firing. v_{rst} is the reset potential, τ is the time constant, and w_j is the synaptic weight for the j^{th} presynaptic neuron.

$$\begin{aligned} v_t &= \frac{1}{\tau} v_{t-1} + \sum_j w_j s_{\text{pre}}, \\ &\text{if } v > v_{th} \text{ then } v \leftarrow v_{rst}, \\ s &= H(v - v_{th}) \end{aligned} \tag{1}$$

QANNs mirror SNNs in structure but not in temporality, operationalized through Eq. 2. The activations (a_{pre}) are quantized and multiplied by synaptic weights, aggregated, and then scaled to produce the discretized output activations (a). a' denotes the weighted sum of pre-activation values, and a the quantized activations, with Δ_a serving as the quantization step.

$$a' = \sum_j w_j a_{pre}, \qquad a = \lfloor \frac{a'}{\Delta_a} \rfloor \times \Delta_a \tag{2}$$

SNNs and QANNs' computational equivalence is captured in Eq. 3, where the forward pass in QANNs is deconstructed into a summation over T quantization levels, akin to SNNs' summation over T time steps.

$$a = \sum_j w_j a_{pre} = \sum_j w_j \sum_{t=0}^{T} 2^t a_{pre}^{(t)} = \sum_{t=0}^{T} 2^t \sum_j w_j a_{pre}^{(t)} \tag{3}$$

In this framework, $a_{pre}^{(t)}$ represents the t^{th} bit of the binary representation of a_{pre}, illustrating the decomposition of QANN forward computation into bit-wise operations, which closely resemble SNN's temporal spike integrations, as shown in Fig. 1.

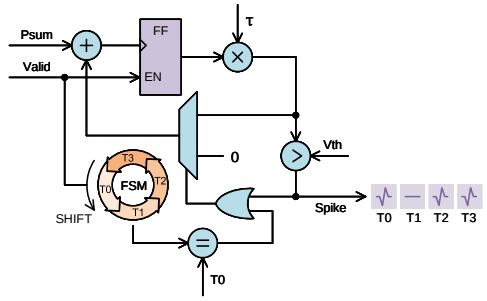
The above discussion demonstrates that the Generalized Matrix Multiplication (GeMM) process of input activation values and weights in QANNs can be converted to the multiplication of binary spikes with weights, which has a similar representation to SNNs. The distinction arises in the subsequent integration stage; SNNs accumulate results over time, while QANNs do so across bit-widths. Below, we explore the hardware implementations for these integrations and their parallels.

Fig. 2 illustrates the input integration for neurons with varying bit allocations, given a constant synaptic weight bit-width. The `P_sum` is the weighted presynaptic input, and `Valid` signals its readiness for neuronal processing. `Last` indicates the final time step, used alongside `Valid` to capture the quantized spike vector output. A Finite State Machine (FSM) controls the timing, resetting neuron potential at the start. The subfigures (a)-(c) demonstrate the membrane potential integration for different spike-time configurations, such as 1/4, 4/1, and 2/2. This setup allows for unfolding multiple input currents for parallel processing. Hence, we assert that varying bit allocations do not significantly alter hardware implementation or complexity.

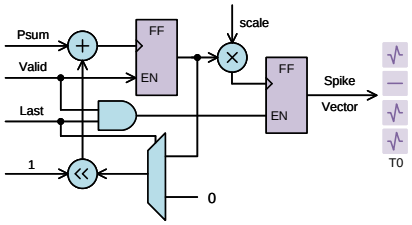
3.2. A Generalized Framework towards QSNNs

This subsection introduces a generalized framework for Quantitative SNNs (QSNNs). The framework aims to provide a cohesive understanding of weight quantization, time steps, spike pattern transformation, and energy consumption evaluation.

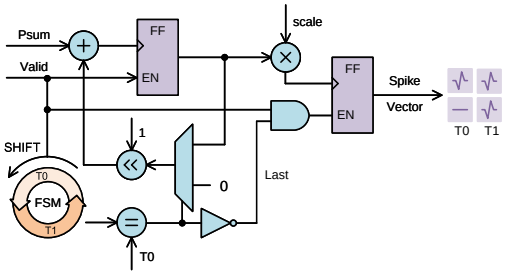
Bit Budget: A More Atomistic Measure In conventional approaches to calculating the computational overhead of SNNs, synaptic operations (SOPs) are often utilized as the fundamental unit of measurement. SOPs typically assume that synaptic weights are fixed-precision floating-point numbers and estimate energy costs based on the op-



(a) 1-bit spike, 4-time step neural integration unit.



(b) 4-bit spike, 1-time step neural integration unit.



(c) 2-bit spike, 2-time step neural integration unit.

Figure 2. Comparison of different allocation of bit budgets.

erations between binary spikes and continuous synaptic weights on specific neuromorphic hardware. This methodology presupposes an ideal operation of the designed networks on particular hardware. Furthermore, this method only considers the impact of spike rate and time step on energy consumption while neglecting the potential influence of weight bit-width and spike patterns. This oversight complicates the unification of SNNs and QANNs under a common framework. Therefore, we propose refining SOPs into a more fundamental unit, the Bit Budget, which we define as follows:

$$\mathrm{BB} = T \cdot b_w \cdot b_s \tag{4}$$

As depicted in Eq. 4, we consider the following three factors in tandem to estimate the computational overhead of a single synapse: the time step (T), the weight bit-width (b_w), and the bit-width of the spike (b_s), also referred to as spike patterns. This presents a more flexible degree of freedom compared to previous works, which only considered the impact of the time step and assumed fixed bit-width for synaptic weights, along with a limited binary representation for spikes. To validate the effectiveness of this metric, we used an Field Programmable Gate Arrays (FPGA) platform to analyze the bit budget corresponding to different devices and their synaptic energy consumption, as illustrated

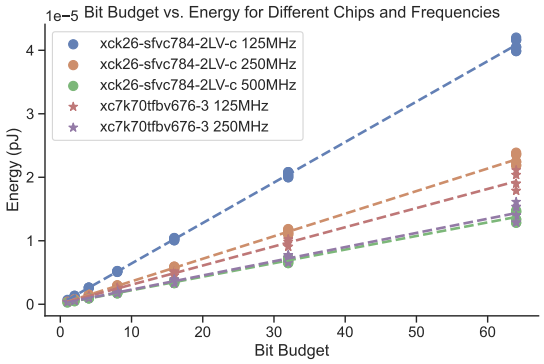


Figure 3. Bit Budget versus energy consumption for individual synaptic operations.

in Fig. 3. The figure reveals a noticeable linear relationship between the bit budget and synaptic energy consumption, affirming the bit budget's validity. Next, we will revisit SNNs and optimize the computational overhead of SNNs, guided by the bit budget.

Quantification of Weights In the realm of quantized neural networks, weight quantization is pivotal for optimizing storage and computational overhead without significantly sacrificing performance. This is essential for both ANNs and SNNs, especially when pursuing efficient hardware implementations. However, despite the importance of weight quantization in SNNs, it has not received as much attention as in QANNs.

The quantization function is shown in Eq. 5, which transforms continuous or high-precision floating-point weights w into a discrete integer set w_q with reduced bit-width. This function is governed by a quantized step Δ, a critical parameter that balances information fidelity against compression level.

$$w_q = \lfloor \frac{w}{\Delta} \rceil \times \Delta \tag{5}$$

$\lfloor \cdot \rceil$ is the standard rounding operation. Upon quantization, each original weight w is supplanted by a quantized value w_q, altering network dynamics. The core challenge is to turn Δ and the quantization scheme such that the quantized network's output y_q, remains as close as possible to that of the unquantized network y, despite the inherent loss of precision. The similarities between SNN and QANN architectures allow for the cross-application of quantization strategies, as outlined in Section 3.1. Our contribution does not delve into the complexities of specific quantization techniques; rather, it presents a unified framework for QSNNs. We utilize a symmetric quantization approach, with Δ set to 2^{T-1}, ensuring that the quantized weights w_q fall within the range $[-1, 1]$.

Step-State Bit Allocation Our preliminary research revealed the functional equivalence between T-step SNNs

and T-bit quantized state ANNs. Building upon this foundation, in the following sections, we will delve deeper into how SNNs, under a fixed bit budget, can strategically allocate bits to the representation of spike patterns and time steps to enhance overall performance.

In conventional SNNs, neuronal responses are binary, leveraging the temporal dimension to encode rich information. In contrast, QANNs encode data within quantized state bits, devoid of a distinct temporal dimension. A proposed hybrid model, therefore, seeks to synergize these approaches, judiciously distributing bits across time steps and spike patterns. For example, rather than dedicating all bits to spike patterns or temporal neuronal activity, a model might apportion its bit budget to improve temporal resolution (by increasing time steps) while using the remainder for enhanced state representation (through elevated spike patterns). This strategy promises a more refined balance between temporal granularity and state complexity, potentially unveiling new operational modes and efficiencies beyond the reach of SNNs or QANNs.

The concept of bit allocation in SNNs initially stems from the study of burst neurons [33]. The burst neuron model categorizes neuronal activity into three distinct states: resting, spiking, and bursting, demonstrating its advanced capabilities. Building upon this, we have developed an enhanced model in which neurons dynamically allocate computational resources or Bit Budgets between the temporal domain and spike type. This is achieved by modulating time steps and various spike patterns, enabling neurons to emit a collection of different states rather than mere binary signals.

In the proposed model, the allocation of bits is more flexible and is not strictly confined to representing either the time steps or the quantization levels of the neuron's activity. Instead, it permits a dynamic distribution of computational resources:

$$\begin{aligned} v_t &= \frac{1}{\tau} v_{t-1} + \sum_j w_j s_{\text{pre}}, \quad t \in \{1, 2, ..., \lfloor \frac{T}{N} \rfloor\} \\ &\text{if } v > v_{th}, \text{ then } v \leftarrow v_{rst}; \end{aligned} \tag{6}$$

$$s = \lfloor \frac{v}{v_{th}} \rfloor \times v_{th} \tag{7}$$

As shown in Eq. 7, the neuron's membrane potential (v) at each time step (t) is influenced by its previous state and the current synaptic inputs (s_{pre}), weighted by the synaptic weights (w_j). When the membrane potential exceeds the threshold, the neuron fires a spike, after which the neuron's membrane potential is reset. The potential quantization determines the emitted spike pattern (s). This quantization allows the spike to carry more information, utilizing the bit budget effectively.

This mechanism introduces an adaptable allocation strategy. Part of the bit budget is used to increase the number of time steps, enriching the temporal resolution, while the rest is employed to enhance the spike patterns via finer quantization levels. Such balanced allocation provides more nuanced control over the interplay between time-sensitive dynamics and the richness of state information, paving the way for more efficient and adaptable neural computations in QSNNs.

Computational Effort Estimates for SNNs In light of these limitations and inspired by the computational paradigm in [40], we propose two advanced metrics aimed at a more holistic and hardware-independent evaluation of computational expenses based on the bit budget: the Synaptic Arithmetic Computation Effort (S-ACE) and the Neuromorphic Synaptic Arithmetic Computation Effort (NS-ACE). These metrics are specifically tailored to assess the resource expenditure of SNNs in both generic and neuromorphic computing environments, offering broader applicability.

The S-ACE is calculated with Eq. 8:

$$\text{S-ACE} = \sum_{w \in W,\ s \in S} n_{w,s} \cdot \text{BB} \tag{8}$$

In Eq. 8, $n_{w,s}$ denotes the count of multiply-accumulate operations (MACs) for a b_w-bit number and a b_s-bit number extracted directly from the neural network's architecture. W and S constitute the sets containing all bit-widths engaged during the neural network's inference stage.

On the contrary, NS-ACE integrates the neuron's firing rate (fr_s) into its computations to more accurately reflect the energy dynamics inherent to neuromorphic hardware systems. This model is predicated on the principle that energy consumption occurs exclusively during spike events. The formulation for NS-ACE is as follows:

$$\text{NS-ACE} = \sum_{w \in W,\ s \in S} fr_s \cdot n_{w,s} \cdot \text{BB} \tag{9}$$

In Eq. 9, the term fr_s represents the firing rate of neurons, emphasizing the concept that in neuromorphic systems, the absence of spiking activity is associated with negligible energy expenditure.

Our introduction of S-ACE and NS-ACE marks a significant stride toward a more refined, realistic, and hardware-agnostic evaluation of computational costs in SNNs. By integrating the complexities inherent in real-world applications, these metrics offer a versatile and comprehensive framework for researchers and engineers working in the SNN landscape.

4. Experiments

To assess the performance of QSNNs and determine their comparative effectiveness against conventional neural net-

work models across various scenarios, we conducted a comprehensive evaluation using a range of datasets. Our analysis included traditional static datasets such as ImageNet [5] and CIFAR10/100 [14], along with a collection of neuromorphic datasets like CIFAR10-DVS [16], DVS128 Gesture [1], and N-Caltech101 [27]. For detailed information about these datasets and the experimental settings, please refer to the Appendix A.

Our experimental analysis centers on the spike patterns and weight quantization of SNNs, aiming to dissect the subtle differences and interconnections between SNNs and ANNs. It is worth mentioning that our research findings can be integrated with contemporary neural network quantization techniques to further enhance model performance. However, while we have achieved state-of-the-art results in several SNN benchmarks, the focus of our study is not to establish new performance benchmarks for SNNs. Instead, our goal is to enrich the understanding of how SNNs function under quantization constraints and their relationship with their non-spiking counterparts.

4.1. Evaluation on Static Datasets

As seen in Tab. 1, we examine the influence of bit allocation, model size, and accuracy on various SNN structures, including convolutional and Transformer variants on the ImageNet dataset. Our findings indicate that for static images, allocating bits to enhance spike patterns, rather than increasing time steps, leads to superior performance due to the static nature of these images. This approach, favoring dense spike patterns, proves more efficient in information transmission while maintaining a consistent computational budget, as quantified by the S-ACE and NS-ACE metrics. Additionally, Tab. 1 reveals that reducing weight bit-width results in a reduction of parameters and variations in performance.

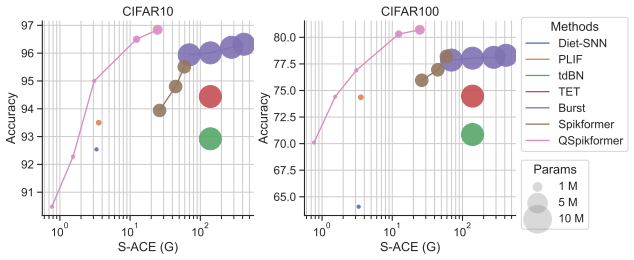


Figure 4. Accuracy, S-ACE and model parameters on the CIFAR dataset. S-ACE is expressed in logarithmic coordinates.

The results in Tab. 2 have shown the effects of allocating bit budgets between spike patterns and time steps on CIFAR10/100 datasets while keeping weight and bit-widths consistent. The optimal model performance on CIFAR10/100 is achieved by limiting time steps to 1 or 2 and maximizing the spike pattern bit-width, which is consistent with the results on the ImageNet dataset. A detailed exposition is available in Appendix A.4. Furthermore, Fig. 4 graphically represents the relationships among accuracy, S-ACE (logarithmically scaled), and model parameters, illustrating the unique balances each method strikes between accuracy, computational cost, and model size. Our methodology demonstrates a proficient equilibrium of these pivotal factors.

4.2. Evaluation on Neuromorphic Datasets

For neuromorphic datasets, we utilize the model outlined in Tab. 2 and undertake a comprehensive analysis to evaluate the influence of bit distribution between spike patterns and time steps on model performance, maintaining a uniform weight bit-width. For the sake of simplicity, we set the weight bit-width b_w to 1.

Tab. 3 reveals that adjusting the bit allocation between spike patterns and time steps results in considerable performance trade-offs. However, the effect of weight bit-width on model performance appears less substantial, possibly due to the constrained size of the neuromorphic datasets, where weight bit-width may not be a critical performance factor.

4.3. Bit Budget Allocation

The performance of SNN models is contingent upon the optimized interplay of weight bit-width (b_w), spike patterns (b_s), and the count of time steps (T). We conceptualize the cumulative product of these parameters as the Bit Budget, corresponding to the S-ACE for a single operation. The pivotal aspect of our research involves determining the optimal allocation of these variables within the constraints of a predetermined Bit Budget to augment the comprehensive performance of the model.

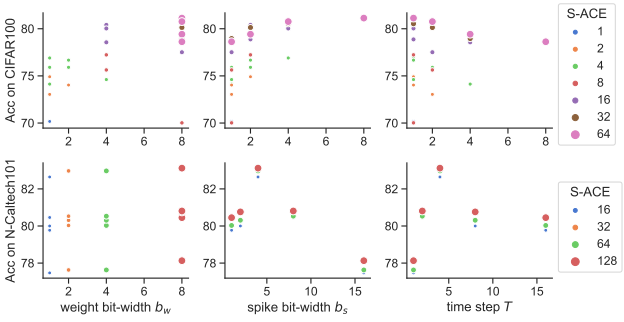


Figure 5. Relationship between different bit budget allocations and model performance.

As shown in Fig. 5, our trials on CIFAR100 and N-Caltech101 unveiled a clear pattern: boosting spike bit-width enhanced performance for static images, while increasing time steps did not. This aligns with the static nature of such images where temporal aspects are less significant. For neuromorphic data, a careful balance between spike bit-widths and time steps was vital. Setting time steps to $T = 4$

Table 1. Evaluation on ImageNet. In quantization, b_w represents the bit-width of weights, b_s denotes the bit-width of spike patterns, and T represents the time steps. SOPs refer to synaptic operations. When calculating S-ACE / NS-ACE for FP32 operations, we assume that these operations can be performed in BF16 without a loss of precision. ↑ means the higher the better, ↓ means the lower the better.

Methods	Architecture	Bit (↓) Budget	Quantization b_w / b_s / T	Params (M) (↓)	SOPs (G) (↓)	S-ACE (G) (↓)	NS-ACE (G) (↓)	Acc (%) (↑)
Hybrid training [29]	ResNet-34	4000	16 / 1 / 250	21.79	-	13608	-	61.48
TET [6]	SEW-ResNet-34	64	16 / 1 / 4	21.79	-	217.72	-	68.00
Spiking ResNet [13]	ResNet-50	5600	16 / 1 / 350	25.56	-	18952	-	72.75
tdBN [42]	Spiking-ResNet-34	96	16 / 1 / 6	21.79	-	326.60	-	63.72
SEW ResNet [7]	SEW-ResNet-34	64	16 / 1 / 4	21.79	3.96	217.72	63.14	67.04
	SEW-ResNet-50	64	16 / 1 / 4	25.56	4.33	215.62	69.03	67.78
Quantized SEW ResNet	SEW-ResNet-34	1	1 / 1 / 1	1.36	1.01	3.40	1.01	52.17
	SEW-ResNet-34	4	2 / 1 / 2	2.72	2.06	13.6	4.11	60.15
	SEW-ResNet-34	4	2 / 2 / 1	2.72	1.83	13.6	3.65	62.36
	SEW-ResNet-34	64	8 / 8 / 1	10.89	7.05	217	56.42	**70.13**
Spikformer [44]	Spikformer-8-384	64	16 / 1 / 4	16.81	5.07	299.6	80.89	70.24
	Spikformer-6-512	64	16 / 1 / 4	23.37	7.56	451.2	121.03	72.46
	Spikformer-8-512	64	16 / 1 / 4	29.68	9.95	516.4	159.16	73.38
	Spikformer-10-512	64	16 / 1 / 4	36.01	10.33	613.6	165.30	73.68
	Spikformer-8-768	64	16 / 1 / 4	66.34	20.01	1197	319.64	74.81
Quantized Spikformer	Spikformer-8-512	1	1 / 1 / 1	1.86	2.12	6.79	2.12	54.54
	Spikformer-8-512	4	2 / 2 / 1	3.71	3.83	26.52	7.65	63.16
	Spikformer-8-512	4	2 / 1 / 2	3.71	3.93	26.92	7.86	61.37
	Spikformer-8-512	8	4 / 2 / 1	7.42	3.59	52.90	14.36	70.87
	Spikformer-8-512	8	2 / 4 / 1	3.71	7.09	52.90	14.17	64.75
	Spikformer-8-512	32	4 / 8 / 1	7.42	13.69	210.1	54.76	76.83
	Spikformer-8-512	32	8 / 4 / 1	14.84	6.51	210.1	52.10	**79.37**

Table 2. Evaluation on CIFAR10/100.

Allocation b_w / b_s / T	Params (M) (↓)	S-ACE (G) (↓)	CIFAR10 Acc(↑)	CIFAR100 Acc (↑)
16 / 1 / 4	4.15	59.10	95.51	78.21
1 / 1 / 1	0.26	0.77	90.48	70.11
1 / 4 / 1	0.26	3.08	95.00	**76.90**
1 / 2 / 2	0.52	3.08	94.43	75.91
1 / 1 / 4	0.26	3.69	93.91	74.13
2 / 2 / 1	0.52	3.08	95.41	**76.67**
2 / 1 / 2	0.52	3.09	93.56	75.91
4 / 1 / 1	1.04	3.09	94.51	74.61
4 / 4 / 1	1.04	12.32	96.84	80.13
4 / 2 / 2	1.04	12.33	96.50	**80.71**
4 / 1 / 4	1.04	14.77	95.94	78.77
8 / 1 / 2	2.08	12.35	95.55	77.72
8 / 2 / 1	2.08	12.33	96.29	**80.00**

and judiciously allocating the remaining bit budget yielded positive results under limited model parameters and computational resources. A marked increase in model accuracy was observed with higher weight bit-widths, plateauing after a certain point. Interestingly, models with low-weight bit-widths, like 1-bit, still performed well, a fact often overlooked in prior research. Properly adjusting weight bit-width can significantly enhance inference efficiency and compact the model size, which is crucial for the inherent efficiency of SNNs.

4.4. Hardware Implementation

We have investigated various bit allocation strategies on FPGA platforms for efficient hardware implementations of neural networks. We examine the balance between spike pattern complexity, time steps, fps, and accuracy, constrained by 8-bit weight precision. As indicated in Tab. 4, the congruence between our FPGA-based experimental results and software simulation results underscores the robustness and universality of our methodologies across varied hardware environments. For detailed hardware experimental setup and more experimental results, please refer to the Appendix B.

Table 3. Evaluation of different Bit Budgets on neuromorphic datasets.

b_w / b_s / T	DVSC10 [16]	DVSG [1]	NCAL [27]
16 / 1 /16	80.7	98.3	80.23
1 / 1 / 16	**79.8** (+0.00)	96.67 (+0.00)	79.77 (+0.00)
1 / 2 / 8	79.3 (-0.50)	**98.48** (+1.81)	80.00 (+0.23)
1 / 4 / 4	63.1 (-16.7)	97.35 (+0.68)	**82.64** (+2.87)
1 / 8 / 2	43.0 (-36.8)	96.59 (-0.08)	80.46 (+0.69)
1 / 16 / 1	35.8 (-44.0)	95.45 (-1.22)	77.47 (-2.30)

Table 4. Hardware Implementation Results on CIFAR10 and DVS-CIFAR10

CIFAR10				
Device	Bit Budget	b_w / b_s / T	fps	Accuracy (%)
xck26	32	8 / 1 / 4	342.35	95.45
		8 / 2 / 2	343.52	95.84
		8 / 4 / 1	343.29	96.41
	64	8 / 1 / 8	155.30	95.47
		8 / 2 / 3	155.55	95.79
		8 / 4 / 2	155.47	96.43
		8 / 8 / 1	155.23	96.51
DVS-CIFAR10				
Device	Bit Budget	b_w / b_s / T	Latency	Accuracy (%)
xck26	128	8 / 1 / 16	49.58	80.30
		8 / 2 / 8	49.56	80.10
		8 / 4 / 4	49.58	69.13
		8 / 8 / 2	49.57	51.31

4.5. Visualization of Sparsity in SNNs

In neural networks, sparsity, characterized by inactive neurons and synapses, contributes to energy efficiency by reducing computational and memory requirements. This subsection explores the influence of various bit budget allocations on neuron sparsity within SNNs. Employing the CIFAR100 and DVS-Gesture datasets and utilizing single-bit weight models, we analyzed neuron firing rates across different allocations of bits to spike patterns and time steps. As depicted in Fig. 6, the results reveal that neuron sparsity was not notably prominent with single-bit spike patterns. Conversely, increasing bits for spike patterns while reducing them for time steps led to lower firing rates, indicating an increase in network sparsity.

The results indicate that allocating more bits to spike patterns than time steps can lead to more energy-efficient SNN operations, potentially without performance loss. This efficiency likely arises from a more effective use of the network's representational capabilities and decreased computational waste. Our findings highlight that SNN sparsity is not a fixed attribute but can be effectively managed through design, with the Bit Budget concept being key in optimizing both sparsity and performance. This observation calls into question the prevailing assumptions regarding intrinsic sparsity in SNNs and propels novel strategies for architecting efficient neural networks.

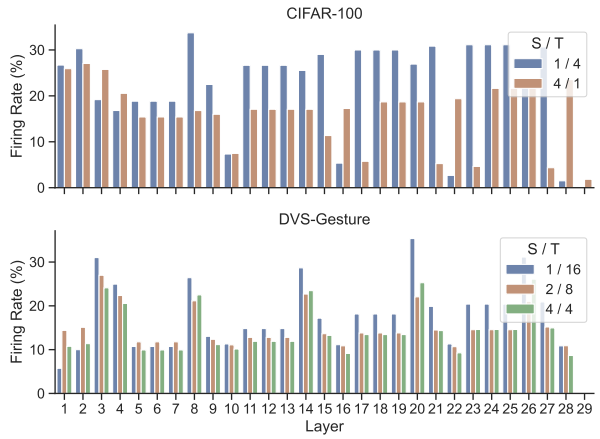


Figure 6. Comparison of firing rates of different layers corresponding to different bit allocation strategies between spike patterns and time steps.

5. Conclusion

The current trend in the design of deep SNNs favors simplified neuron models that aim to achieve a scale comparable to modern artificial neural networks. While these SNNs exhibit notable performance in specific contexts, they often struggle to surpass the performance benchmarks set by traditional ANNs. A critical insight from our research is that SNNs designed following the ANN paradigm inherently mirror the core characteristics of ANNs, akin to ANNs that employ activation value quantization. This revelation highlights the challenges faced by such design philosophies. Our in-depth analysis of SNNs accentuates the pivotal role of bit allocation strategies in regulating network sparsity and enhancing energy efficiency. We discern that sparsity in SNNs is not intrinsic but can be effectively modulated and optimized through meticulous design strategies.

Consequently, we introduce the Bit Budget concept as a systematic approach to enhance network representational capabilities while minimizing computational waste, thus balancing sparsity and performance. In our bit budgeting, prioritizing the state representation of SNNs over time steps has been shown to augment energy efficiency without compromising performance. Through simulations and FPGA experiments, we have validated the robustness and applicability of our approach across various computational settings. This research deepens the understanding of SNN dynamics and provides practical guidelines for building more efficient SNNs. It carves out new pathways for advancing SNNs, especially in developing energy-efficient systems for real-world applications.

Our findings challenge the prevailing notion that contemporary SNNs are intrinsically high-efficiency systems. We advocate for a paradigm that aligns harmoniously with neuroscience rather than pursuing efficiency as an end in itself. We aspire to see the field pivot towards a more integrated approach with neuroscience, fostering intelligent networks that are energy-efficient and deeply rooted in the principles of brain-inspired artificial intelligence.

Acknowledgements

This research is financially supported by a funding from Institute of Automation, Chinese Academy of Sciences (Grant No. E411230101).

References

[1] Arnon Amir, Brian Taba, David Berg, Timothy Melano, Jeffrey McKinstry, Carmelo Di Nolfo, Tapan Nayak, Alexander Andreopoulos, Guillaume Garreau, Marcela Mendoza, et al. A low power, fully event-based gesture recognition system. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7243–7252, 2017. 6, 7, 1

[2] Sander M Bohte. Error-backpropagation in networks of fractionally predictive spiking neurons. In *International conference on artificial neural networks*, pages 60–68. Springer, 2011. 2

[3] Matthieu Courbariaux, Itay Hubara, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Binarized neural networks: Training deep neural networks with weights and activations constrained to+ 1 or-1. *arXiv preprint arXiv:1602.02830*, 2016. 2

[4] Mike Davies, Narayan Srinivasa, Tsung-Han Lin, Gautham Chinya, Yongqiang Cao, Sri Harsha Choday, Georgios Dimou, Prasad Joshi, Nabil Imam, Shweta Jain, et al. Loihi: A neuromorphic manycore processor with on-chip learning. *Ieee Micro*, 38(1):82–99, 2018. 1

[5] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009. 6, 1

[6] Shikuang Deng, Yuhang Li, Shanghang Zhang, and Shi Gu. Temporal efficient training of spiking neural network via gradient re-weighting. *arXiv preprint arXiv:2202.11946*, 2022. 7, 3

[7] Wei Fang, Zhaofei Yu, Yanqi Chen, Tiejun Huang, Timothée Masquelier, and Yonghong Tian. Deep residual learning in spiking neural networks. *Advances in Neural Information Processing Systems*, 34:21056–21069, 2021. 7, 1

[8] Wei Fang, Zhaofei Yu, Yanqi Chen, Timothée Masquelier, Tiejun Huang, and Yonghong Tian. Incorporating learnable membrane time constant to enhance learning of spiking neural networks. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 2661–2671, 2021. 3

[9] Mitchell A Gordon, Kevin Duh, and Nicholas Andrews. Compressing bert: Studying the effects of weight pruning on transfer learning. *arXiv preprint arXiv:2002.08307*, 2020. 1

[10] Bing Han, Feifei Zhao, Yi Zeng, Wenxuan Pan, and Guobin Shen. Enhancing efficient continual learning with dynamic structure development of spiking neural networks. *arXiv preprint arXiv:2308.04749*, 2023. 1

[11] Song Han, Huizi Mao, and William J Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149*, 2015. 2

[12] Song Han, Jeff Pool, John Tran, and William Dally. Learning both weights and connections for efficient neural network. *Advances in neural information processing systems*, 28, 2015. 2

[13] Yangfan Hu, Huajin Tang, and Gang Pan. Spiking deep residual networks. *IEEE Transactions on Neural Networks and Learning Systems*, 2021. 2, 7

[14] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009. 6, 1

[15] Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. Pruning filters for efficient convnets. *arXiv preprint arXiv:1608.08710*, 2016. 2

[16] Hongmin Li, Hanchao Liu, Xiangyang Ji, Guoqi Li, and Luping Shi. Cifar10-dvs: an event-stream dataset for object classification. *Frontiers in neuroscience*, 11:309, 2017. 6, 7, 1

[17] Jindong Li, Guobin Shen, Dongcheng Zhao, Qian Zhang, and Yi Zeng. Firefly: A high-throughput hardware accelerator for spiking neural networks with efficient dsp and memory optimization. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 2023. 1

[18] Jindong Li, Guobin Shen, Dongcheng Zhao, Qian Zhang, and Yi Zeng. Firefly v2: Advancing hardware support for high-performance spiking neural network with a spatiotemporal fpga accelerator. *arXiv preprint arXiv:2309.16158*, 2023. 1

[19] Yanjing Li, Sheng Xu, Baochang Zhang, Xianbin Cao, Peng Gao, and Guodong Guo. Q-vit: Accurate and fully quantized low-bit vision transformer. *Advances in Neural Information Processing Systems*, 35:34451–34463, 2022. 1

[20] Yang Lin, Tianyu Zhang, Peiqin Sun, Zheng Li, and Shuchang Zhou. Fq-vit: Post-training quantization for fully quantized vision transformer. *arXiv preprint arXiv:2111.13824*, 2021. 1

[21] Zhuang Liu, Mingjie Sun, Tinghui Zhou, Gao Huang, and Trevor Darrell. Rethinking the value of network pruning. *arXiv preprint arXiv:1810.05270*, 2018. 2

[22] Zhenhua Liu, Yunhe Wang, Kai Han, Wei Zhang, Siwei Ma, and Wen Gao. Post-training quantization for vision transformer. *Advances in Neural Information Processing Systems*, 34:28092–28103, 2021. 1

[23] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017. 1

[24] Wolfgang Maass. Networks of spiking neurons: the third generation of neural network models. *Neural networks*, 10 (9):1659–1671, 1997. 1, 2

[25] Paul A Merolla, John V Arthur, Rodrigo Alvarez-Icaza, Andrew S Cassidy, Jun Sawada, Filipp Akopyan, Bryan L Jackson, Nabil Imam, Chen Guo, Yutaka Nakamura, et al. A million spiking-neuron integrated circuit with a scalable communication network and interface. *Science*, 345(6197):668–673, 2014. 1

[26] Markus Nagel, Rana Ali Amjad, Mart Van Baalen, Christos Louizos, and Tijmen Blankevoort. Up or down? adaptive rounding for post-training quantization. In *International Conference on Machine Learning*, pages 7197–7206. PMLR, 2020. 2

[27] Garrick Orchard, Ajinkya Jayawant, Gregory K Cohen, and Nitish Thakor. Converting static image datasets to spiking neuromorphic datasets using saccades. *Frontiers in neuroscience*, 9:437, 2015. 6, 7, 1

[28] Nitin Rathi and Kaushik Roy. Diet-snn: Direct input encoding with leakage and threshold optimization in deep spiking neural networks. *arXiv preprint arXiv:2008.03658*, 2020. 1

[29] Nitin Rathi, Gopalakrishnan Srinivasan, Priyadarshini Panda, and Kaushik Roy. Enabling deep spiking neural networks with hybrid conversion and spike timing dependent backpropagation. *arXiv preprint arXiv:2005.01807*, 2020. 7, 3

[30] Victor Sanh, Thomas Wolf, and Alexander Rush. Movement pruning: Adaptive sparsity by fine-tuning. *Advances in Neural Information Processing Systems*, 33:20378–20389, 2020. 1

[31] Guobin Shen, Dongcheng Zhao, and Yi Zeng. Backpropagation with biologically plausible spatiotemporal adjustment for training deep spiking neural networks. *Patterns*, 3(6), 2022. 1, 2

[32] Guobin Shen, Dongcheng Zhao, Yiting Dong, and Yi Zeng. Brain-inspired neural circuit evolution for spiking neural networks. *Proceedings of the National Academy of Sciences*, 120(39):e2218173120, 2023. 2

[33] Guobin Shen, Dongcheng Zhao, and Yi Zeng. Exploiting high performance spiking neural networks with efficient spiking patterns. *arXiv preprint arXiv:2301.12356*, 2023. 5, 1, 3

[34] Wenjie Wei, Malu Zhang, Jilin Zhang, Ammar Belatreche, Jibin Wu, Zijing Xu, Xuerui Qiu, Hong Chen, Yang Yang, and Haizhou Li. Event-driven learning for spiking neural networks. *arXiv preprint arXiv:2403.00270*, 2024. 2

[35] Jibin Wu, Chenglin Xu, Xiao Han, Daquan Zhou, Malu Zhang, Haizhou Li, and Kay Chen Tan. Progressive tandem learning for pattern recognition with deep spiking neural networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(11):7824–7840, 2021. 2

[36] Man Yao, Jiakui Hu, Zhaokun Zhou, Li Yuan, Yonghong Tian, Bo Xu, and Guoqi Li. Spike-driven transformer. *arXiv preprint arXiv:2307.01694*, 2023. 1, 2

[37] Yi Zeng, Dongcheng Zhao, Feifei Zhao, Guobin Shen, Yiting Dong, Enmeng Lu, Qian Zhang, Yinqian Sun, Qian Liang, Yuxuan Zhao, et al. Braincog: A spiking neural network based, brain-inspired cognitive intelligence engine for brain-inspired ai and brain simulation. *Patterns*, 4(8), 2023. 2

[38] Malu Zhang, Hong Qu, Ammar Belatreche, Yi Chen, and Zhang Yi. A highly effective and robust membrane potential-driven supervised learning method for spiking neurons. *IEEE transactions on neural networks and learning systems*, 30(1):123–137, 2018.

[39] Malu Zhang, Jiadong Wang, Jibin Wu, Ammar Belatreche, Burin Amornpaisannon, Zhixuan Zhang, Venkata Pavan Kumar Miriyala, Hong Qu, Yansong Chua, Trevor E Carlson, et al. Rectified linear postsynaptic potential function for backpropagation in deep spiking neural networks. *IEEE transactions on neural networks and learning systems*, 33 (5):1947–1958, 2021. 2

[40] Yichi Zhang, Zhiru Zhang, and Lukasz Lew. Pokebnn: A binary pursuit of lightweight accuracy. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12475–12485, 2022. 5

[41] Dongcheng Zhao, Guobin Shen, Yiting Dong, Yang Li, and Yi Zeng. Improving stability and performance of spiking neural networks through enhancing temporal consistency. *arXiv preprint arXiv:2305.14174*, 2023. 2

[42] Hanle Zheng, Yujie Wu, Lei Deng, Yifan Hu, and Guoqi Li. Going deeper with directly-trained larger spiking neural networks. In *Proceedings of the AAAI conference on artificial intelligence*, pages 11062–11070, 2021. 2, 7, 3

[43] Shuchang Zhou, Yuxin Wu, Zekun Ni, Xinyu Zhou, He Wen, and Yuheng Zou. Dorefa-net: Training low bitwidth convolutional neural networks with low bitwidth gradients. *arXiv preprint arXiv:1606.06160*, 2016. 2

[44] Zhaokun Zhou, Yuesheng Zhu, Chao He, Yaowei Wang, Shuicheng Yan, Yonghong Tian, and Li Yuan. Spikformer: When spiking neural network meets transformer. *arXiv preprint arXiv:2209.15425*, 2022. 1, 2, 7, 3

[45] Rui-Jie Zhu, Qihang Zhao, and Jason K Eshraghian. Spikegpt: Generative pre-trained language model with spiking neural networks. *arXiv preprint arXiv:2302.13939*, 2023. 2

Are Conventional SNNs Really Efficient? A Perspective from Network Quantization

Supplementary Material

A. Experimental Details

A.1. Static Datasets

In order to verify the efficiency of QSNN, we performed a series of comprehensive evaluations on several datasets. We describe these datasets in more detail below.

ImageNet [5] With over a million categorized images spread across $1,000$ diverse categories, ImageNet stands as a colossal and pivotal dataset. The images in ImageNet, which cover a wide spectrum from tangible items to abstract notions, demonstrate considerable variations in attributes like scale, pose, and illumination. Consequently, it provides a rigorous testing ground for diverse computer vision tasks and serves as an industry-standard benchmark.

CIFAR [14] The CIFAR datasets, particularly CIFAR10 and CIFAR100, are quintessential datasets in the field of machine learning, commonly employed for assessing image classification algorithms. CIFAR10 comprises $60,000$ 32×32 color images, evenly distributed across 10 distinct categories, with each category containing $6,000$ images. In contrast, CIFAR100 maintains the same overall number of images and resolution but is divided into 100 categories, each containing 600 images, offering a finer granularity of classification.

A.2. Neuromorphic Datasets

Neuromorphic datasets provide a benchmark for assessing algorithms tailored to neuromorphic hardware, which often deal with spiking neural networks (SNNs) and event-driven data. Such datasets capture real-world, dynamic visual information in a format suitable for SNNs, emphasizing the importance of time and asynchronous events in information processing.

CIFAR10-DVS [16] The CIFAR10-DVS dataset is a neuromorphic version of the popular CIFAR10 dataset. Instead of static images, CIFAR10-DVS provides sequences of events generated by a Dynamic Vision Sensor (DVS), a type of neuromorphic camera that only captures pixel-level brightness changes, making it more power-efficient and better suited for real-time applications.

DVS-Gesture [1] The DVS-Gesture dataset is a benchmark collection in the field of neuromorphic vision, designed for the task of gesture recognition. The DVS-Gesture dataset includes a variety of hand gestures from multiple subjects, performed under different lighting conditions, and from various angles, challenging the robustness of spiking neural network (SNN) models in recognizing and classifying dynamic patterns.

N-Caltech101 [27] The N-Caltech101 dataset is a neuromorphic version of the well-known Caltech101 dataset, which has been converted using a DVS camera to create event-based vision data. Unlike the original dataset with static images, N-Caltech101 provides a sequence of events as each image is presented to the sensor, capturing the temporal aspect of visual perception. This dataset contains the same categories as the original Caltech101, encompassing a wide range of objects such as animals, vehicles, and everyday items, thus providing a comprehensive challenge for testing the effectiveness of SNNs in processing and classifying neuromorphic vision data.

A.3. Experimental Settings

Following the framework defined by Zhou et al. [44] for consistency and comparability, we set the input dimensions for the ImageNet dataset to 224×224. For other static image datasets, their native image dimensions were retained. The neural morphology datasets underwent a resizing process, adjusting the event streams to 48×48 to streamline computational demands. We used a batch size of 128 and selected the AdamW optimizer [23]. Training involved 310 epochs for the ImageNet dataset, whereas for other datasets, 400 epochs were deemed sufficient. We initialized our models with a learning rate of 0.0005.

Our experiments spanned various neural network architectures, prominently SEW-ResNet [7] and SpikFormer [44]. Although the architectural blueprints of our models mirrored the original specifications, we took the liberty of introducing modifications to accentuate their event-driven nature. This involved imposing bit width restrictions on the output of SEW-ResNet's blocks and SpikFormer's attention matrices. Our thorough evaluations then centered on the performances of S-ACE and NS-ACE for these models across our selected datasets.

A.4. Additional Evaluation

In Tab. 5, various methods and configurations showcase their performance on both CIFAR10 and CIFAR100. Burst[33], although demonstrating impressive performance across different configurations, retains a relatively large number of parameters due to its limited consideration of pulse patterns and its lack of quantization for model weights. Building upon Spikiformer [44], our further experiments reveal that models with multiple pulse patterns can better utilize the bit budget. When combined with

weight quantization, we managed to achieve an accuracy of 96.84% on CIFAR10 and 80.13% on CIFAR100 with just 1/4 of the original model parameters and 1/5 of the S-ACE. These results indicate a performance improvement of 1.33% and 1.92% respectively over the non-optimized methods.

From the neuromorphic results in Tab. 3, it's evident that as we modify the allocation between the spike patterns and simulation steps (S and T), there are significant performance trade-offs. The most notable decline in performance is observed for the DVSC10 dataset when adjusting the S/T ratio from 1/16 to 8/2, indicating a substantial sensitivity to the bit allocation strategy. This is further accentuated in the 1/16/1 configuration, leading to a staggering decrease of 44% in accuracy. However, the impact of weight bit-width on model performance is not particularly pronounced. This might be attributed to the relatively small dataset size of the neural morphology dataset, where weight bit-width may not be the predominant factor influencing model performance.

Building upon our previous discussions and the observed performance on the CIFAR dataset, these results further underscore the intricacy of employing spiking neural networks on the neural morphology dataset. The judicious selection and allocation of bits are of paramount importance when dealing with neural morphology data characterized by temporal dynamics.

B. Hardware Implementation

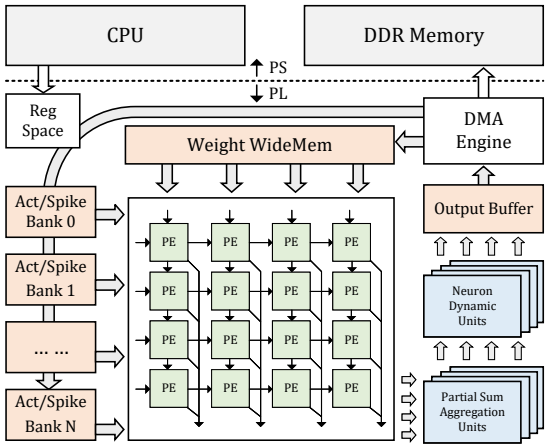


Figure 7. System block design of our hardware accelerator.

To assess the hardware efficacy of various bit allocation strategies, we developed a flexible FPGA accelerator prototype that has the flexibility to support different time steps and activation bit widths. Drawing inspiration from established advancements in bit-serial accelerator, the accelerator first breaks down multi-bit spikes across multiple time steps into several iterations of general matrix multiplication between binary inputs and multi-bit weights. These iterations' resulting partial sum matrices are then aggregated and transmitted to the neurodynamic units to generate the final output. In this way, the hardware efficiency of models with distinct bit allocation strategies can be assessed using the same hardware backend to ensure a fair and accurate comparison. Notably, bit-serial accelerators are prevalent in deep learning accelerator research due to their ability to provide a finer computation granularity, offering a more expansive optimization space. This includes the implementation of mixed bit-width quantization across different layers, enabling more extensive optimization opportunities.

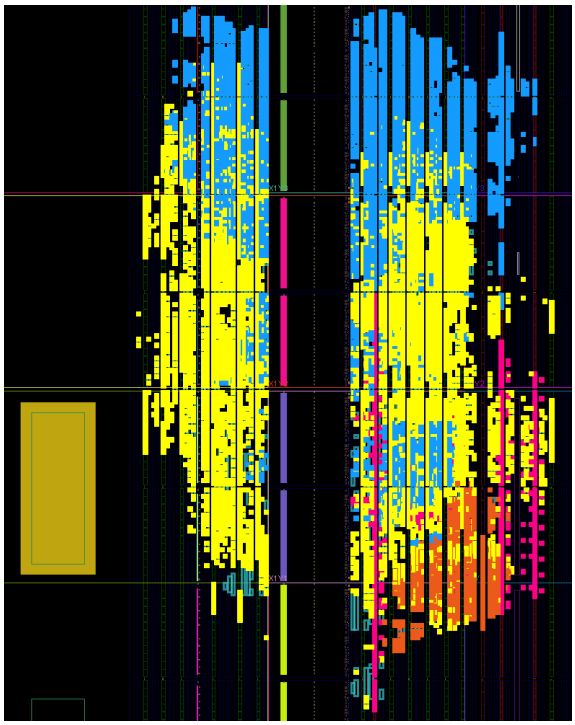


Figure 8. Place and route of the accelerator on xck26 FPGA device.

The system architecture of the accelerator is shown in Fig. 7. This accelerator is implemented on the Xilinx Zynq Ultrascale series FPGA platform, which features embedded host CPUs on the Processing System (PS) side. The PS manages task scheduling, while the Programmable Logic (PL) side handles heavy computational workloads within the FPGA fabric. The system diagram is depicted below. The DMA engine is responsible for input/output activation and weight data transfer between the PL logic and external DDR memory. To optimize data reuse and alleviate DDR bandwidth constraints, on-chip weight and input activation buffers are instantiated. Input buffers are split into several banks for a flexible data layout arrangement. A systolic array, consisting of Processing Elements (PEs) arranged in a

Table 5. Additional evaluation on CIFAR10/100. † represent results reproduced with the same experimental setup.

Methods	Architecture	Assignment b_w / b_s / T	Params (M)	S-ACE (G)	Accuracy CIFAR10	Accuracy CIFAR100
PLIF [8]	CIFARNet-Fang	16 / 1 / 8	0.58	3.57	93.50	74.36†
Diet-SNN [29]	ResNet-20	16 / 1 / 5	0.27	3.32	92.54	64.07†
tdBN [42]	ResNet-19	16 / 1 / 4	12.63	139	92.92	70.86
TET [6]	ResNet-19	16 / 1 / 4	12.63	139	94.44	74.47
Burst [33]	ResNet-19	16 / 2 / 1	12.63	69.6	95.94	77.86
	ResNet-19	16 / 2 / 2	12.63	139	96.01	78.04
	ResNet-19	16 / 2 / 4	12.63	278	96.21	78.12
	ResNet-19	16 / 2 / 6	12.63	417	96.32	78.31
Spikformer [44]	Spikformer-4-256	16 / 1 / 4	4.15	26.28	93.94	75.96
	Spikformer-2-384	16 / 1 / 4	4.15	44.50	94.80	76.95
	Spikformer-4-384	16 / 1 / 4	4.15	59.10	95.51	78.21
Quantized Spikformer	Spikformer-4-384	1 / 1 / 1	0.26	0.77	90.48	70.11
	Spikformer-4-384	1 / 4 / 1	0.26	3.08	95.00	76.90
	Spikformer-4-384	1 / 2 / 2	0.52	3.08	94.43	75.91
	Spikformer-4-384	1 / 1 / 4	0.26	3.69	93.91	74.13
	Spikformer-4-384	2 / 2 / 1	0.52	3.08	95.41	76.67
	Spikformer-4-384	2 / 1 / 2	0.52	3.09	93.56	75.91
	Spikformer-4-384	4 / 1 / 1	1.04	3.09	94.51	74.61
	Spikformer-4-384	4 / 4 / 1	1.04	12.32	96.84	80.13
	Spikformer-4-384	4 / 2 / 2	1.04	12.33	96.50	80.71
	Spikformer-4-384	4 / 1 / 4	1.04	14.77	95.94	78.77
	Spikformer-4-384	8 / 1 / 2	2.08	12.35	95.55	77.72
	Spikformer-4-384	8 / 2 / 1	2.08	12.33	96.29	80.00

two-dimensional grid, executes arithmetic-intensive matrix multiplications using binary inputs and multi-bit weights. Each PE comprises a multiplexer and an accumulator; the binary input controls the multiplexer to determine whether the accumulator adds the current weight. A central controller is designed to fetch the necessary inputs and weight data for the systolic array's operations. Following the matrix multiplications, a post-processing unit aggregates partial sums from each binary slice of inputs and generates output spikes through the neurodynamic units. Subsequently, the output data is organized in the output buffer and transferred back to the external DDR memory to serve as the input for the next layer.

The accelerator decomposes the multi-bit activation (or spike) tensor into binary slices across multiple time steps. The bit-width of the activations (or spikes) and the quantity of time steps determine the number of binary slices. Hence, it is unsurprising that SNNs and QANNs sharing the same bit budget will result in an identical computational workload using such a hardware backend.

The out-of-context Place and Route implementation results are depicted in Fig. 8. In the illustration, the highlighted bright yellow area signifies the systolic array. In contrast, the highlighted blue area encompasses the post-processing units responsible for the partial sum aggregation process and the neurodynamic process. The pink line demarcates the utilized block RAM for the weight buffer, whereas the orange area indicates the split input buffer banks. The dark yellow rectangle on the left represents the CPUs on the PS side of the Zynq Ultrascale device. The comprehensive resource utilization is detailed in Tab. 6.

Table 6. Utilization of different resources on the xck26 FPGA device.

Resource	Used	Total
Look Up Table	23298	117120
Flip Flops	44084	234240
DSP48E2	512	1248
Block Ram	89.5	144
Ultra Ram	8	64