Gated Attention Coding for Training High-Performance and Efficient Spiking Neural Networks

Xuerui Qiu[1,3,4], **Rui-Jie Zhu**[5], **Yuhong Chou**[6], **Zhaorui Wang**[4] **Liang-Jian Deng**[4*], **Guoqi Li**[1,2*]

[1]Institute of Automation, Chinese Academy of Sciences, China
[2]Peng Cheng Laboratory, China
[3]School of Future Technology, University of Chinese Academy of Sciences
[4] University of Electronic Science and Technology of China, China
[5]University of California, Santa Cruz, USA
[6]Xi'an Jiaotong University
{qiuxuerui2024, guoqi.li}@ia.ac.cn, rzhu48@ucsc.edu, zhaorui_wang@std.uestc.edu.cn, liangjian.deng@uestc.edu.cn

Abstract

Spiking neural networks (SNNs) are emerging as an energy-efficient alternative to traditional artificial neural networks (ANNs) due to their unique spike-based event-driven nature. Coding is crucial in SNNs as it converts external input stimuli into spatio-temporal feature sequences. However, most existing deep SNNs rely on direct coding that generates powerless spike representation and lacks the temporal dynamics inherent in human vision. Hence, we introduce Gated Attention Coding (GAC), a plug-and-play module that leverages the multi-dimensional gated attention unit to efficiently encode inputs into powerful representations before feeding them into the SNN architecture. GAC functions as a preprocessing layer that does not disrupt the spike-driven nature of the SNN, making it amenable to efficient neuromorphic hardware implementation with minimal modifications. Through an observer model theoretical analysis, we demonstrate GAC's attention mechanism improves temporal dynamics and coding efficiency. Experiments on CIFAR10/100 and ImageNet datasets demonstrate that GAC achieves state-of-the-art accuracy with remarkable efficiency. Notably, we improve top-1 accuracy by 3.10% on CIFAR100 with only 6-time steps and 1.07% on ImageNet while reducing energy usage to 66.9% of the previous works. To our best knowledge, it is the first time to explore the attention-based dynamic coding scheme in deep SNNs, with exceptional effectiveness and efficiency on large-scale datasets. Code is available at https://github.com/bollossom/GAC.

Introduction

Artificial neural networks (ANNs) have garnered remarkable acclaim for their potent representation and astounding triumphs in a plethora of artificial intelligence domains such as computer vision (Krizhevsky et al. 2017), natural language processing (Hirschberg 2015) and big data applications (Niu et al. 2020). Nonetheless, this comes at a significant cost in terms of energy consumption. In contrast, spiking neural networks (SNNs) exhibits heightened biological plausibility (Maass 1997), spike-driven nature, and low power consumption on neuromorphic hardware, e.g., TrueNorth (Merolla et al. 2014), Loihi (Davies et al. 2018), Tianjic (Pei et al. 2019).

Moreover, the versatility of SNNs extends to various tasks,

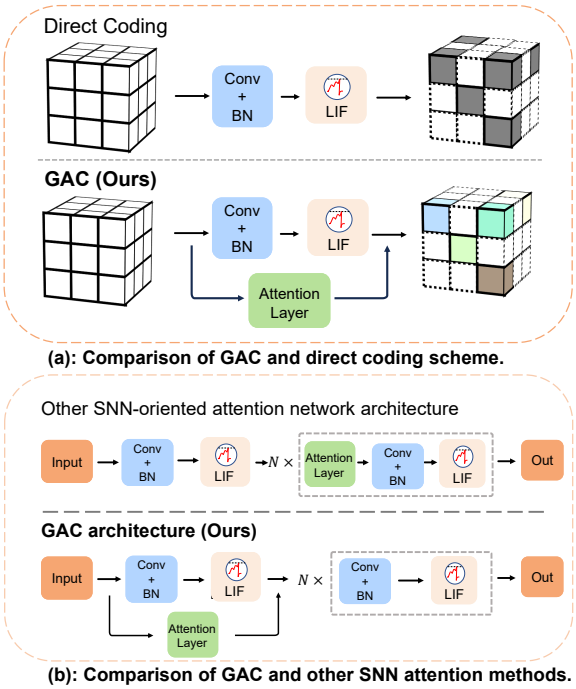

Figure 1: How our Gated Attention Coding (GAC) differs from existing SNNs' coding (Wu et al. 2019) and attention methods (Yao et al. 2021, 2023c). In (a), the solid-colored cube represents the float values, the gray cube denotes the binary spike values, and the cube with the dotted line represents the sparse values. In comparison with direct coding, GAC generates spatio-temporal dynamics output with powerful representations. In (b), compared to other attention methods, GAC only adds the attention module to the encoder without requiring N Multiply-Accumulation (MAC) blocks for dynamically calculating attention scores in subsequent layers.

*Corresponding author.

including image classification (Hu et al. 2021; Wei et al. 2023; Xu et al. 2023), image reconstruction (Qiu et al. 2023b), and language generation (Zhu et al. 2023), although the majority of their applications currently lie within the field of computer vision.

To integrate SNNs into the realm of computer vision, the initial challenge lies in transforming static images into spatio-temporal feature sequences. Various coding schemes have emerged to address this issue, such as rate coding (Van Rullen and Thorpe 2001), temporal coding (Comsa et al. 2020), and phase coding (Kim et al. 2018). Among these, direct coding (Wu et al. 2019) as shown in Fig. 1-(a), excel in training SNNs on large-scale datasets with minimal simulation time steps. Moreover, by adopting direct coding, recent SNN models (Li et al. 2021b; Shen et al. 2023; Zhou et al. 2023) achieve state-of-the-art performance across various datasets, showcasing the immense potential of coding techniques. However, this approach utilizes a trainable layer to generate float values repetitively at each time step. The repetitive nature of direct coding leads to periodic identical outputs at every time step, generating powerless spike representations and limiting spatio-temporal dynamics. In addition, the repetitive nature of direct coding fails to generate the temporal dynamics inherent in human vision, which serves as the fundamental inspiration for SNN models. Human vision is characterized by its ability to process and perceive dynamic visual stimuli over time. The repetitive nature of direct coding falls short in replicating this crucial aspect, emphasizing the need for alternative coding schemes that can better emulate the temporal dynamics observed in human vision.

Humans can naturally and effectively find salient regions in complex scenes (Itti, Koch, and Niebur 1998). Motivated by this observation, attention mechanisms have been introduced into deep learning and achieved remarkable success in a wide spectrum of application domains, which is also worth to be explored in SNNs as shown in Fig. 1-(b) (Yao et al. 2021, 2023c). However, it has been observed that implementing attention mechanisms to directly modify membrane potential and dynamically compute attention scores for each layer, rather than using static weights, disrupts the fundamental asynchronous spike-driven communication in these methods. Consequently, this approach falls short of providing full support for neuromorphic hardware.

In this paper, we investigate the shortcomings of traditional direct coding and introduce an innovative approach termed Gated Attention Coding (GAC) as depicted in Fig. 1. Instead of producing periodic and powerless results, GAC leverages a multi-dimensional attention mechanism for gating to elegantly generate powerful temporal dynamic encodings from static datasets. As a preprocessing layer, GAC doesn't disrupt the SNNs' spike-driven, enabling efficient neuromorphic hardware implementation with minimal modifications. Experimental results demonstrate that our GAC not only significantly enhances the performance of SNNs, but also notably reduces latency and energy consumption. Moreover, our main contributions can be summarized as follows:

- We propose an observer model to theoretically analyze direct coding limitations and introduce the GAC scheme, a plug-and-play preprocessing layer decoupled from the SNN architecture, preserving its spike-driven nature.
- We evaluate the feasibility of GAC and depict the encoding result under both direct coding and GAC setups to demonstrate the powerful representations of GAC and its advantage in generating spatio-temporal dynamics.
- We demonstrate the effectiveness and efficiency of the proposed method on the CIFAR10/100 and ImageNet datasets. Our method outperforms the previous state-of-the-art works and shows significant improvements across all test datasets with lower energy consumption.

Related Works

Bio-inspired Spiking Neural Networks

Spiking Neural Networks (SNNs) offer a promising approach to achieving energy-efficient intelligence. These networks aim to replicate the behavior of biological neurons by employing binary spiking signals, where a value of 0 indicates no activity and a value of 1 represents a spiking event. The spike-driven communication paradigm in SNNs is inspired by the functionality of biological neurons and holds the potential for enabling energy-efficient computational systems (Roy, Jaiswal, and Panda 2019; Zhu et al. 2022; Shan et al. 2023; Deng et al. 2023). By incorporating advanced deep learning and neuroscience knowledge, SNNs can offer significant benefits for a wide range of applications (Jin et al. 2022; Qiu et al. 2023a,b; Kundu et al. 2023). Recently, there exist two primary methods of training high-performance SNNs. One way is to discretize ANN into spike form through neuron equivalence (Li et al. 2021a), i.e., ANN-to-SNN conversion, but this requires a long simulation time step and boosts the energy consumption. We employ the direct training method (Wu et al. 2018) and apply surrogate gradient training.

SNN Coding Schemes

Numerous coding schemes are proposed for image classification tasks. Phase coding (Kim et al. 2018) used a weighted spike to encode each pixel and temporal coding (Park et al. 2020; Comsa et al. 2020; Zhou et al. 2021) represents information with the firing time of the first neuron spike. These methods have been successfully applied to simple datasets with shallow networks, but achieving high performance becomes more challenging as datasets and networks become larger and more complex. To address this issue, rate coding (Van Rullen and Thorpe 2001), which encodes each pixel using spike firing frequency, has been suggested. However, it suffers from long time steps to remain high performance, while small time steps result in lower representation resolution. To overcome these limitations, Wu et al. (2019) proposed the direct coding, in which input is given straight to the network without conversion to spikes and image-spike encoding is done by the first $\{Conv\text{-}BN\}$ layer. Then repeat this procedure at each time step and feed the results to spiking neurons. Finally, these encoded spikes will be sent to the SNN architecture for feature extraction. However, the limited powerless spike representations in SNNs using direct coding leads to parameter sensitivity and subpar performance. The repetition operation fails to generate dynamic output and neglects redundant data, thus underutilizing the spatio-temporal extraction

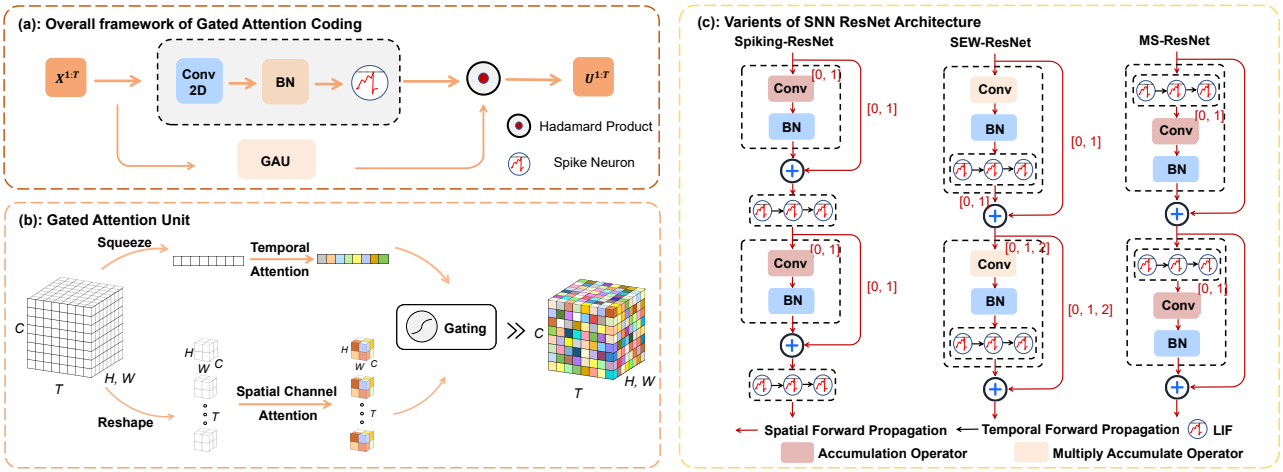

Figure 2: The GAC-SNN framework consists of two main components: an encoder and an architecture. In (a), we introduce the encoder, i.e., the GAC module. (b) focuses on the GAU, which acts as the fundamental building block of the GAC module. It comprises Temporal Attention, Spatial Channel Attention, and Gating sub-modules. (c) Common SNN ResNet architectures. The Conv layer in SEW-ResNet uses a multiply-accumulate operator, not spike computations. Spiking-ResNet retains its spike-driven nature via direct coding, while GAC disrupts it. More details can be seen in discussions. MS-ResNet avoids floating-point multiplications, preserving its spike-driven nature. Hence, we use the MS-ResNet to benefit from neuromorphic implementations.

ability of subsequent SNN architectures and increasing energy consumption in neuromorphic hardware.

Attention Mechanism

Initially introduced to enhance the performance of sequence-to-sequence tasks (Bahdanau, Cho, and Bengio 2014), the attention mechanism is a powerful technique that enables improved processing of pertinent information (Ioffe and Szegedy 2015). By effectively filtering out distracting noise, the attention mechanism facilitates more focused and efficient data processing, leading to enhanced performance in various applications. Yao et al. (2021) attach the squeeze-and-excitation (Hu, Shen, and Sun 2018) attention block to the SNNs' temporal-wise input, assessing the significance over different frames during training and discarding irrelevant frames during inferencing. However, this method only gets better performance on small datasets with shallow networks. Yao et al. (2023c) switch CBAM attention (Woo et al. 2018) to multi-dimension attention and inject it in SNN architecture, revealing deep SNNs' potential as a general architecture to support various applications.

Currently, integrating attention blocks into SNN architectures is difficult because it necessitates designing separate multiplicative modules in subsequent layers to dynamically calculate attention scores. This impedes the spike-driven nature of inherently additive SNNs. Hence, the seamless integration of SNNs with neuromorphic hardware is hindered, primarily due to the hardware's limited support for static weights, which is a key requirement for conventional attention implementations. A potential solution to address this issue involves confining the application of attention mechanisms solely to the encoder, i.e., the first layer of the SNNs. By limiting the attention modifications to the initial stage, the subsequent layers can still maintain the essential spike-driven communication. This approach holds promise in enabling a more feasible implementation of SNNs on neuromorphic hardware, as it mitigates the incompatibility arising from dynamic attention mechanisms throughout the architecture.

Method

In this section, we first introduce the iterative spiking neuron model. Then we proposed the Gated Attention Coding (GAC) and Gated Attention Unit (GAU) as the basic block of it. Next, we provide the overall framework for training GAC-SNN. Moreover, we conduct a comprehensive analysis of the direct coding scheme and explain why our GAC outperforms in generating spatio-temporal dynamics encoding results.

Iterative Spiking Neuron Model

We adopt the Leaky Integrate-and-Fire (LIF) spiking neuron model and translate it to an iterative expression with the Euler method (Wu et al. 2018; Yao et al. 2023b). Mathematically, the LIF-SNN layer can be described as an explicitly iterable version for better computational traceability:

$$\begin{cases} \boldsymbol{U}^{t,n} = \boldsymbol{H}^{t-1,n} + f(\boldsymbol{W}^{\boldsymbol{n}}, \boldsymbol{X}^{t,n-1}) \\ \boldsymbol{S}^{t,n} = \Theta(\boldsymbol{U}^{t,n} - \boldsymbol{V}_{th}) \\ \boldsymbol{H}^{t,n} = \tau \boldsymbol{U}^{t,n} \cdot (1 - \boldsymbol{S}^{t,n}) + \boldsymbol{V}_{reset} \boldsymbol{S}^{t,n}, \end{cases} \tag{1}$$

where τ is the time constant, t and n respectively represent the indices of the time step and the n-th layer, $\boldsymbol{W}$ denotes synaptic weight matrix between two adjacent layers, $f(\cdot)$ is the function operation stands for convolution (Conv) or fully connected (FC), $\boldsymbol{X}$ is the input, and $\boldsymbol{\Theta}(\cdot)$ denotes the Heaviside step function. When the membrane potential $\boldsymbol{U}$ exceeds the firing threshold $\boldsymbol{V}_{th}$, the LIF neuron will trigger a spike $\boldsymbol{S}$. Moreover, $\boldsymbol{H}$ represents the membrane potential after the trigger event which equals to $\tau \boldsymbol{U}$ if no spike is generated and otherwise equals to the reset potential $\boldsymbol{V}_{reset}$.

Gated Attention Unit (GAU)

Temporal Attention. To establish temporal-wise relationships between SNNs' input, we first perform the squeezing step on the spatial-channel feature map of the repeated input $\boldsymbol{X} \in \mathbb{R}^{T\times C\times H\times W}$, where T is the simulation time step and C is the channel size. Then we use Avgpool and Maxpool to calculate the maximum and average of the input $\boldsymbol{X}$. Additionally, we use a shared MLP network to turn both average-pooled and max-pooled features into a temporal weight vector $\boldsymbol{M} \in \mathbb{R}^{T}$, i.e.,

$$\begin{aligned}\mathcal{F}_T(\boldsymbol{X}) = &(\boldsymbol{W}_m(\mathrm{ReLU}(\boldsymbol{W}_n(\mathrm{AvgPool}(\boldsymbol{X})))) \\ &+ \boldsymbol{W}_m(\mathrm{ReLU}(\boldsymbol{W}_n(\mathrm{MaxPool}(\boldsymbol{X})))),\end{aligned} \tag{2}$$

where $\mathcal{F}_T(\cdot)$ is the functional operation of temporal attention. And $\boldsymbol{W}_m \in \mathbb{R}^{T\times \frac{T}{r}}$, $\boldsymbol{W}_n \in \mathbb{R}^{\frac{T}{r}\times T}$ are the weights of two shared dense layers. Moreover, r is the temporal dimension reduction factor used to manage its computing overhead.

Spatial Channel Attention. To generate the spatial channel dynamics for encoding result, we use a shared 2-D convolution operation at each time step to get the spatial channel matrix $\boldsymbol{N} = [\boldsymbol{N}^1, \boldsymbol{N}^2, \cdots, \boldsymbol{N}^t] \in \mathbb{R}^{T\times C\times H\times W}$, i.e.,

$$\mathcal{F}_{SC}(\boldsymbol{X}) = \sum_{i=0}^{K-1}\sum_{j=0}^{K-1} \boldsymbol{W}_{i,j} \cdot \boldsymbol{X}^t_{i,j}, \tag{3}$$

where $\mathcal{F}_{SC}(\cdot)$ is the functional operation of spatial channel attention, $\boldsymbol{W}_{i,j}$ is the learnable parameter and K represents the size of the 2-D convolution kernel size.

Gating. After the above two operations, we get the temporal vector $\boldsymbol{M}$ and spatial channel matrix $\boldsymbol{N}$. Then to extract SNNs' input $\boldsymbol{X}$ temporal-spatial-channel fused dynamics features, we first broadcast the temporal vector to $\mathbb{R}^{T\times 1\times 1\times 1}$ and gating the above result by :

$$\mathcal{F}_G(\boldsymbol{X}) = \sigma(\boldsymbol{M} \odot \boldsymbol{N}) = \sigma(\mathcal{F}_T(\boldsymbol{X}) \odot \mathcal{F}_{SC}(\boldsymbol{X})), \tag{4}$$

where $\sigma(\cdot)$ and $\odot$ are the Sigmoid function and Hadamard Product. By the above three sub-modules, we can get the functional operation $\mathcal{F}_G(\cdot)$ of GAU, which is the basic unit of the next novel coding.

Gated Attention Coding (GAC)

Compared with the previous direct coding (Wu et al. 2019; Hu et al. 2021), we introduce a novel encoding called Gated Attention Coding (GAC). And Fig. 2 describes the specific process. Given that the input $\boldsymbol{X} \in \mathbb{R}^{C\times H\times W}$ of static datasets, we assign the first layer as an encoding layer. We first use the Conv layer to generate features, then repeat this procedure after each time step and feed the results to the LIF neuron and GAU module respectively. Finally, gating the output of the above two modules. Hence, the whole GAC process can be described as:

$$\boldsymbol{O} = \mathcal{F}_G(f^{k\times k}(\boldsymbol{X})) \odot \mathcal{SN}(f^{k\times k}(\boldsymbol{X})), \tag{5}$$

where $\boldsymbol{X}$ and $\boldsymbol{O}$ is the GAC-SNN's input and output, $f^{k\times k}(\cdot)$ is a shared 2-D convolution operation with the filter size of $k \times k$, and $\mathcal{SN}(\cdot)$ is the spiking neuron model. Moreover, $\odot$ is the Hadamard Product, and $\mathcal{F}_G(\cdot)$ is the functional operation of GAU, which can fuse temporal-spatial-channel information for better encoding feature representations.

Overall Training Framework

We give the overall training algorithm of GAC for training deep SNNs from scratch with our GAC and spatio-temporal backpropagation (STBP) (Wu et al. 2018). In the error backpropagation, we suppose the last layer as the decoding layer, and the final output $\boldsymbol{K}$ can be determined by: $\boldsymbol{K} = \frac{1}{T}\sum_{t=1}^{T} \boldsymbol{O}^t$, where $\boldsymbol{O}^t$ is the SNNs' output of the last layer and T is the time steps. Then we calculate the cross-entropy loss function (Rathi and Roy 2021) between output and label, which can be described as:

$$q_i = \frac{e^{k_i}}{\sum_{j=1}^{n} e^{k_j}}, \tag{6}$$

$$\mathcal{L} = -\sum_{i=1}^{n} y_i log(q_i), \tag{7}$$

where $\boldsymbol{K} = (k_1, k_2, \cdots, k_n)$ and $\boldsymbol{Y} = (y_1, y_2, \cdots, y_n)$ are the output vector and label vector.

Theoretical Analysis

To understand the highlights of our proposed method and the role of SNN encoders, we introduce the observer model for measuring direct coding and our GAC. Encoders are used to convert static images into feature sequences, incorporating temporal information into SNNs at each time step. Some encoders are embedded within the SNN as part of it (e.g., the first Conv-based spiking neuron layer for direct coding), while others are not included in the SNN models, e.g., rate coding. The embedded encoders can be easily distinguished from the rest of the network since they use actual values for linear transformations, unlike spikes or quantized data. Functionally, encoders convert static data into the temporal dimension. This definition helps us understand what SNN encoders are.

Definition 1. ***Encoder.*** *An encoder in SNNs for image classification tasks is used to translate static input $\boldsymbol{X} \in \mathbb{R}^{C_{in}\times H\times W}$ into dynamics feature sequences $\boldsymbol{A} \in \mathbb{R}^{T\times C_{out}\times H\times W}$.*

Moreover, two points should be noted in **Definition** 1. Firstly, $\boldsymbol{A}$ is used to indicate the encoders' output no matter spikes or real values. And Membrane Shortcut (MS) ResNet architecture (Hu et al. 2021) is considered to use sequential real values as input after the encoder. Secondly, although two dimensions (C_{out}, T) are changed, the spatial one C_{out} is similar to the operation in ANNs, which means T is unique for SNN encoders. In other words, the time step T is the secret of the time information, and the SNN encoder is designed to generate this added dimension. The discussion above implies that to understand and metric an encoder, we should focus on its temporal dimension and find a proper granularity.

Definition 2. ***Neuron Granularity.*** *Considering $\boldsymbol{A} \in \mathbb{R}^{T\times C_{out}\times H\times W}$ is output feature sequences of the encoder, given a fixed position c, h, w for the output $\boldsymbol{A}$, so that we get a vector along temporal axis $a = \boldsymbol{A}_{:,c,h,w} = \left[a^1, a^2, \cdots, a^t\right]$.*

Here the encoding feature vector a is not subscripted in **Definition** 2 because the encoder is usually symmetric, and the choice of position for analysis does not affect its generality.

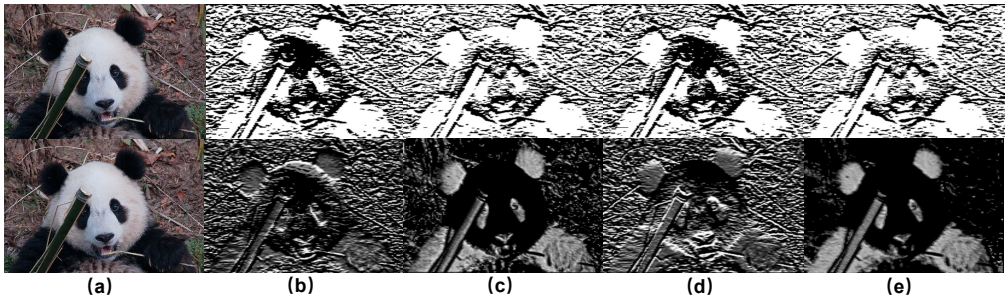

Figure 3: Visualization results. (a) Original image. (b)(c)(d)(e) Encoding results of the direct coding (top) and GAC (bottom) at different time steps. Compared to direct coding, GAC enhances dynamics by introducing variations at each time step.

To measure the vector a, we introduce the observer model and information entropy (Jaynes 1957). Assuming an observer is positioned right behind the encoder, recording and predicting the elements of vector a in a time-ordered manner. Hence, the observer model can be formally established as follows:

- The observer notices the elements of encoded feature sequences $a \in \mathbb{R}^T$ or $\{0,1\}^T$ in time step order.
- At any time step t, the observer remembers all elements from time 1 to time $t-1$, and it is guessing the element a^t of encoded feature sequences $a \in \mathbb{R}^T$.
- The observer is aware of the mechanism or the encoder structure, but not its specific parameters.

Moreover, at time step t, guessing a^t, the observer should answer it with probability. The probability can be described as:

$$p^t(a^t) = p(a^t|a^1, \cdots, a^{t-1}), \tag{8}$$

And we use the information entropy to measure the quantity of information gotten by the observer, i.e.,

$$\mathcal{H}(\boldsymbol{V}^t) = \sum_t p^t(a^t)\log(p^t(a^t)), \tag{9}$$

where $\boldsymbol{V}^t$ is used to indicate the random variable version of a^t. Specifically, when $p^t(a^t) = 0$ or 1, $p^t(a^t)\log(p^t(a^t)) = 0$ it means that a deterministic event contribute 0 to information entropy. Moreover, for the observer model, when the element a^t is deterministic, there is no additional information that deserves observing at time step t.

To better understand the concept of information entropy in this context, it is crucial to consider the role of an encoder whose task is to convert information into tensors that generate temporal dynamics. Ideally, the encoder should utilize as numerous time steps as possible to code information, resulting in a positive information entropy along the time axis. The positive entropy indicates the presence of information, which is crucial for spiking neural networks. While precisely quantifying entropy value is difficult, it is possible to measure the duration of positive entropy. In this way, longer-lasting positive entropy can be considered a more effective use of the temporal dimension.

Definition 3. ***Dynamics Duration & Dynamics Loss.*** *Considering a specific position with encoded feature vector $a = [a^1, a^2, \cdots, a^t]$ alone temporal dimension, if $\exists t_e$ such that $\forall t > t_e$, the observer model's entropy $\mathcal{H}(\boldsymbol{V}^t)$ about possible encoding results is 0. Let $\boldsymbol{T}_e$ be the greatest lower bound of t_e. And $\boldsymbol{T}_e = \inf(t_e)$ is defined as* ***Dynamics Duration****. Then we call the time steps after $\boldsymbol{T}_e$ is* ***Dynamics Loss****.*

According to **Definition** 3, we can delineate the encoder's effective encoding range. Dynamics duration indicates when encoding results entropy $\mathcal{H}(\boldsymbol{V}^t) \geq 0$. And at dynamics loss time steps, the entropy $\mathcal{H}(\boldsymbol{V}^t)$ remains 0, rendering encoding unnecessary. Moreover, to metric the encoder's dynamics, the key is to compare the dynamics duration time step $\boldsymbol{T}_e$.

Proposition 1. *Assuming that the dynamic duration of GAC and direct coding be $\boldsymbol{T}_g$ and $\boldsymbol{T}_d$ respectively, we have $\boldsymbol{T}_g \geq \boldsymbol{T}_d$ when giving same $\{$Conv-BN$\}$ parameters.*

Proof. Denoted that $\boldsymbol{X} \in \mathbb{R}^{T\times C_{out}\times H\times W}$ is the repetitive output of direct coding and GAC after same $\{$*Conv-BN*$\}$ module, $a \in \mathbb{R}^T$ is the encoded feature vector.

For direct coding, it sends the repetitive output $\boldsymbol{X}$ to the spiking neuron for coding, resulting in the encoded feature vector a being powerless and periodic with 0 or 1. Moreover, the period $\boldsymbol{T}_p$ is:

$$\boldsymbol{T}_p = \left\lceil \log_\tau(1 - \frac{\boldsymbol{V}_{th}(1-\tau)}{x_{i,j}}) \right\rceil, \tag{10}$$

where τ is the time constant, $\boldsymbol{V}_{th}$ is the firing threshold and $x_{i,j}$ is the pixel of the input $\boldsymbol{X}$ after $\{$*Conv-BN*$\}$ module. Hence, the subsequent output is predictable when the observer has found the first spike. Thus, the direct coding's dynamic duration $\boldsymbol{T}_d = \boldsymbol{T}_p$. Moreover, the derivation of $\boldsymbol{T}_p$ and analysis of other coding schemes' dynamics can be found in **Appendix**.

For GAC, we multiplied the output of direct coding and GAU to expand the dynamic duration of the encoding results. Thus, the GAC's dynamic duration $\boldsymbol{T}_g = \lfloor \frac{\boldsymbol{T}}{\boldsymbol{T}_d} \rfloor \boldsymbol{T}_d$. It can be seen that $\boldsymbol{T}_g \geq \boldsymbol{T}_d$. □

According to **Proposition** 1, GAC lasts its dynamics longer than direct coding. Moreover, this reflects the superiority of GAC in generating dynamic encoding results. As depicted in Fig. 3, GAC's encoding results on static datasets vary significantly at each time step, i.e., temporal dynamics.

Methods	Architecture	Params (M)	Time Steps	CIFAR10 Acc.(%)	CIFAR100 Acc.(%)
ANN2SNN (Hao et al. 2023)[AAAI]	VGG-16	33.60/33.64	32	95.42	76.45
tdBN (Zheng et al. 2021)[AAAI]	Spiking-ResNet-19	12.63/12.67	6	93.16	71.12
TET (Deng et al. 2022)[ICLR]	Spiking-ResNet-19	12.63/12.67	6	94.50	74.72
GLIF (Yao et al. 2022)[NeurIPS]	Spiking-ResNet-19	12.63/12.67	6	95.03	77.35
MS-ResNet‡ (Hu et al. 2021)	MS-ResNet-18	12.50/12.54	6	94.92	76.41
GAC-SNN	MS-ResNet-18	12.63/12.67	6	**96.46**±0.06	**80.45**±0.27
	MS-ResNet-18	12.63/12.67	4	96.24±0.08	79.83±0.15
	MS-ResNet-18	12.63/12.67	2	96.18±0.03	78.92±0.10
ANN‡ (Hu et al. 2021)	MS-ResNet-18	12.50/12.54	N/A	96.75	80.67

Table 1: Comparison between the proposed methods and previous works on CIFAR datasets. ‡ denote self-implementation results with open-source code (Hu et al. 2021). The "Params" column indicates network parameter size on CIFAR10/100 datasets.

Theoretical Energy Consumption Calculation

The GAC-SNN architecture can transform matrix multiplication into sparse addition, which can be implemented as an addressable addition on neuromorphic chips. In the encoding layer, convolution operations serve as MAC operations that convert analog inputs into spikes, similar to direct coding-based SNNs (Wu et al. 2019). Conversely, in SNN's architecture, the Conv or FC layer transmits spikes and performs AC operations to accumulate weights for postsynaptic neurons. Additionally, the inference energy cost of GAC-SNN can be expressed as follows:

$$E_{total} = E_{MAC} \cdot FL^1_{conv} + E_{AC} \cdot T \cdot (\sum_{n=2}^{N} FL^n_{conv} \cdot fr^n + \sum_{m=1}^{M} FL^m_{fc} \cdot fr^m), \tag{11}$$

where N and M are the total number of Conv and FC layers, E_{MAC} and E_{AC} are the energy costs of MAC and AC operations, and fr^m, fr^n, FL^n_{conv} and FL^m_{fc} are the firing rate and FLOPs of the n-th Conv and m-th FC layer. Previous SNN works ((Horowitz 2014; Rathi and Roy 2021; Yao et al. 2023a)) assume 32-bit floating-point implementation in 45nm technology, where E_{MAC} = 4.6pJ and E_{AC} = 0.9pJ for various operations.

Experiments

In this section, we evaluate the performance of GAC-SNN on static datasets, e.g., CIFAR10, CIFAR100, ImageNet (Li et al. 2017; Krizhevsky et al. 2017). To verify the effectiveness and efficiency of the proposed coding, we integrate the GAC module into the Membrane Shortcut (MS) ResNet (Hu et al. 2021), to see if the integrated architecture can generate significant improvement when compared with previous state-of-the-art works. Specifically, the details of the architecture are shown in Fig. 2-(c), and why we use it is illustrated in the discussions. And Experiments are also verified using Mind-Spore. The code implemented by MindSpore will be open-sourced to this link (https://github.com/mindspore-lab/models/tree/master/research/cas/GAC_AAAI).

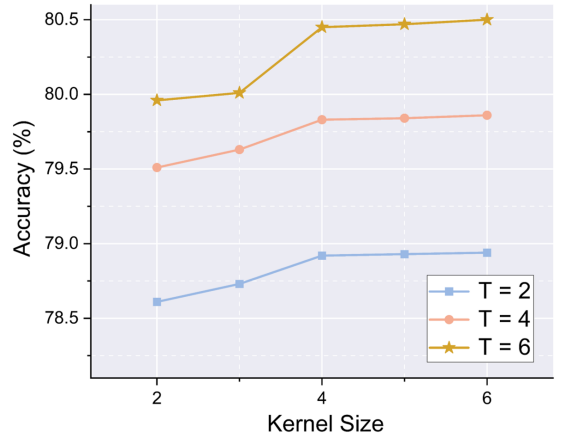

Figure 4: Ablation study on CIFAR100.

GAC Can Produce Powerful and Dynamics Results

We evaluated GAC's effectiveness in reducing redundant temporal information and improving encoding results for static datasets. By training MS-ResNet-34 on ImageNet with and without GAC, we generated the encoding output shown in Fig. 3. And it can be seen that our GAC can help SNNs to capture more texture information. Hence, our approach enhances SNNs' representation ability and temporal dynamics by introducing significant variations in GAC results at each time step, compared to the periodic output of direct coding.

GAC Can Get Effective and Efficient SNNs

Effectiveness. The GAC-SNN demonstrate remarkable performance enhancement compared to existing state-of-the-art works (Table. 1-2). On CIFAR10 and CIFAR100, GAC achieves higher accuracy than previous advanced works using only 2-time steps. With the same time steps, GAC improves 1.43% and 3.10% on CIFAR10 and CIFAR100 over GLIF (Yao et al. 2022). And compared to the baseline MS-ResNet (Hu et al. 2021), our method outperforms it on CIFAR10 and CIFAR100 by 1.54% and 4.04% with 6-time steps. For the

Methods	Architecture	Spike -driven	Params (M)	Time Steps	Power (mJ)	Top-1 Acc.(%)
ANN2SNN (Hao et al. 2023)^{AAAI}	ResNet-34	✓	21.79	64	-	68.61
TET (Deng et al. 2022)^{ICLR}	Spiking-ResNet-34	✓	21.79	6	-	64.79
tdBN (Zheng et al. 2021)^{AAAI}	Spiking-ResNet-34	✓	21.79	6	6.39	63.72
SEW-ResNet (Fang et al. 2021)^{NeurIPS}	SEW-ResNet-34	✗	21.79	4	4.04	67.04
MS-ResNet (Hu et al. 2021)	MS-ResNet-18	✓	11.69	6	4.29	63.10
	MS-ResNet-34	✓	21.80	6	5.11	69.43
Att-MS-ResNet (Yao et al. 2023c)^{TPAMI}	Att-MS-ResNet-18	✗	11.96(+0.27)	6	4.11	64.15*
	Att-MS-ResNet-34	✗	22.30(+0.50)	6	5.05	69.35*
GAC-SNN	MS-ResNet-18	✓	11.82(**+0.13**)	6/4	2.34/**1.49**	**65.14**/64.05
	MS-ResNet-34	✓	21.93(**+0.13**)	6/4	3.38/**2.20**	**70.42**/69.77
ANN (Hu et al. 2021)	MS-ResNet-18	✗	11.69	N/A	14.26	69.76
	MS-ResNet-34	✗	21.80	N/A	16.87	73.30

Table 2: Comparison between the proposed method and previous works on the ImageNet dataset. Power is the average theoretical energy consumption when predicting a batch of images from the test set, details of which are shown in Eq.11. The "Spike-driven" column indicates if an independent design of the multiplication module is required in the SNN architecture. And the mark ✗ in the "Spike-driven" column denotes hindering neuromorphic hardware implementation. * needs a large training time (1000 epochs and 600 batch size) compared to other methods.

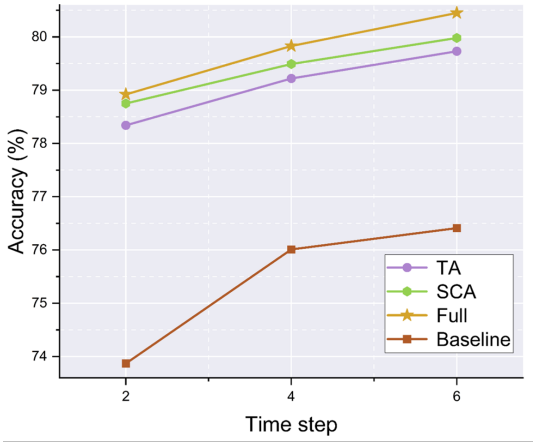

Figure 5: Ablation study on CIFAR100.

larger and more challenging ImageNet dataset, compared with the baseline MS-ResNet (Hu et al. 2021), we apply our GAC to MS-ResNet-18 and can significantly increase the accuracy (65.14% v.s. 63.10%). Compared with other advanced works (Fang et al. 2021; Yao et al. 2023c), GAC-based MS-ResNet-34 achieves 70.42% top-1 accuracy and surpasses all previous directly-trained SNNs with the same depth.

Efficiency. Compared with prior works, the GAC-SNN shines in energy consumption (Table. 2). We first make an intuitive comparison of energy consumption in the SNN field. Specifically, GAC-SNN (This work) vs. SEW-ResNet-34 at 4-time steps: Power, 2.20mJ vs. 4.04mJ. That is, our model has +3.22% higher accuracy than SEW-ResNet with only the previous 54.5% power. And GAC-SNN (This work) vs. MS-ResNet-34 vs. Att-MS-ResNet-34 at 6-time steps: Power, 2.34mJ vs. 4.29mJ vs. 4.11mJ. That is, our model has the lowest power under the same structure and time steps. For instance, as the layers increase from 18 to 34, MS ResNet (baseline) has 1.83×(4.29mJ/2.34mJ) and 1.51×(5.11mJ/3.38mJ) higher energy consumption than our GAC-SNN. At the same time, our task performance on the above same depth network structures has improved by +2.04% and +0.99%, respectively.

Architecture	Schemes	T	Acc.(%)
ResNet19	Phase Coding	8	91.40
VGG16	Temporal Coding	100	92.68
ResNet19	Rate Coding	6	93.16
MS-ResNet18	Direct Coding	6	94.92
MS-ResNet18	GAC	6	**96.46**

Table 3: Comparisons with different coding schemes.

Ablation Study

Comparison between Different SNN Coding Schemes. To future demonstrate the advantage of GAC, we evaluate the performance of our GAC and other coding schemes e.g., Phase coding (Kim et al. 2018), Temporal coding (Zhou et al. 2021), Rate coding (Wu et al. 2019), Direct coding (Wu et al. 2019). Table. 3 displays CIFAR10 test accuracy, where GAC achieves 96.46% top-1 with MS-ResNet-18 in 6-time steps.

The Effect of Parameter Kernel Size K. We investigate the impact of the 2D convolution kernel size K in the Spatial Channel Attention module of our GAC. Specifically, there is a trade-off between performance and latency as kernel size increases. It is almost probable that when kernel size increases, the receptive region of the local attention mechanism also

does so, improving SNN performance. These benefits do, however, come at a cost of high parameters and significant latency. To this end, we trained the GAC-based MS-ResNet-18 on CIFAR100 with various K values. As shown in Fig. 4, accuracy rises with increasing K, plateauing after K exceeds 4. This indicates our GAC maintains strong generalization despite large K variations. To maintain excellent performance and efficiency, we consider employing $K = 4$ in our work.

Comparison of Different Attention. We conducted ablation studies on the Temporal Attention (TA) and Spatial Channel Attention (SCA) modules to assess their effects. Fig. 5 indicates that the SCA module contributes more to performance improvement due to the most SNNs designs that channels outnumber time steps. The SCA module extracts additional significant features compared to the TA module. Notably, regardless of the module we ablate, performance will be affected, which may help you understand our design.

Discussions

Analysis of Different GAC-SNN's ResNet Architecture. Residual connection is a crucial basic operation in deep SNNs' ResNet architecture. And there are three shortcut techniques in existing advanced deep SNNs. Spiking-ResNet (Hu, Tang, and Pan 2021) performs a shortcut between membrane potential and spike. Spike-Element-Wise (SEW) ResNet (Fang et al. 2021) employs a shortcut to connect the output spikes in different layers. Membrane Shortcut (MS) ResNet (Hu et al. 2021), creating a shortcut between membrane potential of spiking neurons in various layers. Specifically, we leverage the membrane shortcut in the proposed GAC for this reason:

Spike-driven describes the capacity of the SNN architecture to convert matrix multiplication (i.e., high-power Multiply-Accumulation) between weight and spike tensors into sparse addition (i.e., low-power Accumulation). The spike-driven operations can only be supported by binary spikes. However, as the SEW shortcut creates the addition between binary spikes, the values in the spike tensors are multi-bit (integer) spikes. Additionally, GAC-based Spiking-ResNet is not entirely spike-driven. Because GAC will change the second layer convolution's inputs to floating-point numbers on the Spiking-ResNet. By contrast, as shown in Fig. 2-(c), spiking neurons are followed by the MS shortcut. Hence, both Conv and FC layers act as sparse addition operations in GAC-based MS-ResNet and always get binary spikes as their input.

Impact of GAC and Other SNNs' Attention Methods on the Spike-driven Nature. As shown in Fig. 1-(b), other SNN-oriented attention works (Yao et al. 2021, 2023c) adding an attention mechanism to the SNN architecture need to design numerous multiplication blocks and prevent all matrix multiplications related to the spike matrix from being converted into sparse additions, which hinders the implementation of neuromorphic hardware. However, adding an attention mechanism in the encoder doesn't hinder it. As the encoder and architecture are decoupled in SNN hardware design (Li et al. 2023), our GAC, like direct coding (Wu et al. 2019), only incorporates a multiplication block for analog-to-spike conversion in the encoder without impacting the spike-driven nature of the sparse addition SNN architecture.

Conclusion

This paper focuses on the SNNs' coding problem, which is described as the inability of direct coding to produce powerful and temporal dynamic outputs. We have observed that this issue manifests as periodic powerless spike representations due to repetitive operations in direct coding. To tackle this issue, we propose Gated Attention Coding (GAC), a spike-driven and neuromorphic hardware-friendly solution that seamlessly integrates with existing Conv-based SNNs. GAC incorporates a multi-dimensional attention mechanism inspired by attention mechanism and human dynamics vision in neuroscience. By effectively establishing sptiao-temporal relationships at each moment, GAC acts as a preprocessing layer and efficiently encodes static images into powerful representations with temporal dynamics while minimizing redundancy. Our method has been extensively evaluated through experiments, demonstrating its effectiveness with state-of-the-art results: CIFAR10 (96.46%), CIFAR100 (80.45%), and ImageNet (70.42%). We hope our investigations pave the way for more advanced coding schemes and inspire the design of high-performance and efficient spike-driven SNNs.

Appendix

Derivation of the direct coding period $\boldsymbol{T}_p$. $\boldsymbol{T}_p$ is the first time step of firing and the period of firing for the corresponding LIF model. It is trivial that T increases monotonically with x and the resolution of the encoding depends on the value of x. The most important thing is that after the LIF model, the direct coding is period coding, making the $0, 1$ output periodical. Moreover, the derivation of the direct coding's period is as follows: Suppose when $t = 0$, the membrane potential $\boldsymbol{U}^t = 0$, and the neuron fires at time step $t = \boldsymbol{T}_p$. According to the direct coding, the input of any specific neuron is $x_{i,j}$ (a constant). Since the threshold and the attenuation factor are separately denoted as $\boldsymbol{V}_{th}$ and τ.According to the iterative formula of the LIF model before the spike fire time, we have:

$$\boldsymbol{U}^t = \tau \boldsymbol{U}^{t-1} + x_{i,j} = \sum_{k=0}^{t-1} \tau^k x_{i,j} = x_{i,j} \sum_{k=0}^{t-1} \tau^k, \quad (12)$$

Also, we have: $\boldsymbol{U}^{T_p-1} \leq \boldsymbol{V}_{th} \leq \boldsymbol{U}^{T_p}$, Bringing Eq.12 to above, we have:

$$x_{i,j} \sum_{k=0}^{T_p-2} \tau^k \leq \boldsymbol{V}_{th} \leq x_{i,j} \sum_{k=0}^{T_p-1} \tau^k, \quad (13)$$

$$T_p = \lceil \log_\tau (1 - \frac{\boldsymbol{V}_{th}(1-\tau)}{x_{i,j}}) \rceil, \quad (14)$$

Acknowledgments

This work is supported by National Science Foundation for Distinguished Young Scholars (62325603), National Natural Science Foundation of China (Grants 62236009, U22A20103), and CAAI-MindSpore Open Fund, developed on OpenI Community.

References

Bahdanau, D.; Cho, K.; and Bengio, Y. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Comsa, I. M.; Potempa, K.; Versari, L.; Fischbacher, T.; Gesmundo, A.; and Alakuijala, J. 2020. Temporal coding in spiking neural networks with alpha synaptic function. In *2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 8529–8533. IEEE.

Davies, M.; Srinivasa, N.; Lin, T.-H.; Chinya, G.; Cao, Y.; Choday, S. H.; Dimou, G.; Joshi, P.; Imam, N.; Jain, S.; et al. 2018. Loihi: A neuromorphic manycore processor with on-chip learning. *IEEE Micro*, 38(1): 82–99.

Deng, H.; Zhu, R.; Qiu, X.; Duan, Y.; Zhang, M.; and Deng, L. 2023. Tensor Decomposition Based Attention Module for Spiking Neural Networks. *arXiv preprint arXiv:2310.14576*.

Deng, S.; Li, Y.; Zhang, S.; and Gu, S. 2022. Temporal Efficient Training of Spiking Neural Network via Gradient Re-weighting. In *International Conference on Learning Representations (ICLR)*.

Fang, W.; Yu, Z.; Chen, Y.; Huang, T.; Masquelier, T.; and Tian, Y. 2021. Deep residual learning in spiking neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 34, 21056–21069.

Hao, Z.; Bu, T.; Ding, J.; Huang, T.; and Yu, Z. 2023. Reducing ANN-SNN Conversion Error Through Residual Membrane Potential. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, volume 37, 11–21.

Hirschberg. 2015. Advances in natural language processing. *Science*, 349(6245): 261–266.

Horowitz, M. 2014. 1.1 computing's energy problem (and what we can do about it). In *2014 IEEE International Solid-State Circuits Conference Digest of Technical Papers (ISSCC)*, 10–14. IEEE.

Hu, J.; Shen, L.; and Sun, G. 2018. Squeeze-and-excitation networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 7132–7141.

Hu, Y.; Deng, L.; Wu, Y.; Yao, M.; and Li, G. 2021. Advancing Spiking Neural Networks Towards Deep Residual Learning. *arXiv preprint arXiv:2112.08954*.

Hu, Y.; Tang, H.; and Pan, G. 2021. Spiking deep residual networks. *IEEE Transactions on Neural Networks and Learning Systems*, 1–6.

Ioffe, S.; and Szegedy, C. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning (ICML)*, 448–456. pmlr.

Itti, L.; Koch, C.; and Niebur, E. 1998. A model of saliency-based visual attention for rapid scene analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(11): 1254–1259.

Jaynes, E. T. 1957. Information theory and statistical mechanics. *Physical review*, 106(4): 620.

Jin, C.; Zhu, R.-J.; Wu, X.; and Deng, L.-J. 2022. Sit: a bionic and non-linear neuron for spiking neural network. *arXiv preprint arXiv:2203.16117*.

Kim, J.; Kim, H.; Huh, S.; Lee, J.; and Choi, K. 2018. Deep neural networks with weighted spikes. *Neurocomputing*, 311: 373–386.

Krizhevsky, A.; Sutskever, I.; Hinton, G. E.; et al. 2017. ImageNet Classification with Deep Convolutional Neural Networks. *Commun. ACM*, 60(6): 84–90.

Kundu, S.; Zhu, R.-J.; Jaiswal, A.; and Beerel, P. A. 2023. Recent Advances in Scalable Energy-Efficient and Trustworthy Spiking Neural networks: from Algorithms to Technology. *arXiv preprint arXiv:2312.01213*.

Li, H.; Liu, H.; Ji, X.; Li, G.; and Shi, L. 2017. Cifar10-dvs: an event-stream dataset for object classification. *Frontiers in Neuroscience*, 22: 244131.

Li, J.; Shen, G.; Zhao, D.; Zhang, Q.; and Zeng, Y. 2023. FireFly: A High-Throughput Hardware Accelerator for Spiking Neural Networks With Efficient DSP and Memory Optimization. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 31(8): 1178–1191.

Li, Y.; Deng, S.; Dong, X.; Gong, R.; and Gu, S. 2021a. A free lunch from ANN: Towards efficient, accurate spiking neural networks calibration. In *International conference on machine learning (ICML)*, 6316–6325. PMLR.

Li, Y.; Guo, Y.; Zhang, S.; Deng, S.; Hai, Y.; and Gu, S. 2021b. Differentiable Spike: Rethinking Gradient-Descent for Training Spiking Neural Networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 34, 23426–23439.

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

Merolla, P. A.; Arthur, J. V.; Alvarez-Icaza, R.; Cassidy, A. S.; Sawada, J.; Akopyan, F.; Jackson, B. L.; Imam, N.; Guo, C.; Nakamura, Y.; et al. 2014. A million spiking-neuron integrated circuit with a scalable communication network and interface. *Science*, 345(6197): 668–673.

Niu, X.; Li, B.; Li, C.; Xiao, R.; Sun, H.; Deng, H.; and Chen, Z. 2020. A dual heterogeneous graph attention network to improve long-tail performance for shop search in e-commerce. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD)*, 3405–3415.

Park, S.; Kim, S.; Na, B.; and Yoon, S. 2020. T2FSNN: deep spiking neural networks with time-to-first-spike coding. In *2020 57th ACM/IEEE Design Automation Conference (DAC)*, 1–6. IEEE.

Pei, J.; Deng, L.; Song, S.; Zhao, M.; Zhang, Y.; Wu, S.; Wang, G.; Zou, Z.; Wu, Z.; He, W.; et al. 2019. Towards artificial general intelligence with hybrid Tianjic chip architecture. *Nature*, 572(7767): 106–111.

Qiu, X.; Luan, Z.; Wang, Z.; and Zhu, R.-J. 2023a. When Spiking Neural Networks Meet Temporal Attention Image Decoding and Adaptive Spiking Neuron. In *International Conference on Learning Representations Tiny Paper*.

Qiu, X.-R.; Wang, Z.-R.; Luan, Z.; Zhu, R.-J.; Wu, X.; Zhang, M.-L.; and Deng, L.-J. 2023b. VTSNN: a virtual temporal spiking neural network. *Frontiers in Neuroscience*, 17: 1091097.

Rathi, N.; and Roy, K. 2021. Diet-snn: A low-latency spiking neural network with direct input encoding and leakage and threshold optimization. *IEEE Transactions on Neural Networks and Learning Systems*, 34(6): 3174–3182.

Roy, K.; Jaiswal, A.; and Panda, P. 2019. Towards spike-based machine intelligence with neuromorphic computing. *Nature*, 575(7784): 607–617.

Shan, Y.; Qiu, X.; Zhu, R.-j.; Li, R.; Wang, M.; and Qu, H. 2023. OR Residual Connection Achieving Comparable Accuracy to ADD Residual Connection in Deep Residual Spiking Neural Networks. *arXiv preprint arXiv:2311.06570*.

Shen, J.; Xu, Q.; Liu, J. K.; Wang, Y.; Pan, G.; and Tang, H. 2023. ESL-SNNs: An Evolutionary Structure Learning Strategy for Spiking Neural Networks. In *Proceedings of the AAAI conference on artificial intelligence (AAAI)*, volume 37, 86–93.

Van Rullen, R.; and Thorpe, S. J. 2001. Rate coding versus temporal order coding: what the retinal ganglion cells tell the visual cortex. *Neural computation*, 13(6): 1255–1283.

Wei, W.; Zhang, M.; Qu, H.; Belatreche, A.; Zhang, J.; and Chen, H. 2023. Temporal-Coded Spiking Neural Networks with Dynamic Firing Threshold: Learning with Event-Driven Backpropagation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 10552–10562.

Woo, S.; Park, J.; Lee, J.-Y.; and Kweon, I. S. 2018. Cbam: Convolutional block attention module. In *Proceedings of the European conference on computer vision (ECCV)*, 3–19.

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

Wu, Y.; Deng, L.; Li, G.; Zhu, J.; Xie, Y.; and Shi, L. 2019. Direct training for spiking neural networks: Faster, larger, better. In *Proceedings of the AAAI conference on artificial intelligence (AAAI)*, volume 33, 1311–1318.

Xu, Q.; Li, Y.; Shen, J.; Liu, J. K.; Tang, H.; and Pan, G. 2023. Constructing deep spiking neural networks from artificial neural networks with knowledge distillation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 7886–7895.

Yao, M.; Gao, H.; Zhao, G.; Wang, D.; Lin, Y.; Yang, Z.; and Li, G. 2021. Temporal-wise attention spiking neural networks for event streams classification. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 10221–10230.

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

Yao, M.; Zhang, H.; Zhao, G.; Zhang, X.; Wang, D.; Cao, G.; and Li, G. 2023b. Sparser spiking activity can be better: Feature Refine-and-Mask spiking neural network for event-based visual recognition. *Neural Networks*, 166: 410–423.

Yao, M.; Zhao, G.; Zhang, H.; Hu, Y.; Deng, L.; Tian, Y.; Xu, B.; and Li, G. 2023c. Attention Spiking Neural Networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(8): 9393–9410.

Yao, X.; Li, F.; Mo, Z.; and Cheng, J. 2022. GLIF: A Unified Gated Leaky Integrate-and-Fire Neuron for Spiking Neural Networks. In *Advances in Neural Information Processing Systems (Neurips)*, volume 35, 32160–32171.

Zheng, H.; Wu, Y.; Deng, L.; Hu, Y.; and Li, G. 2021. Going Deeper with Directly-Trained Larger Spiking Neural Networks. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, volume 35, 11062–11070.

Zhou, S.; Li, X.; Chen, Y.; Chandrasekaran, S. T.; and Sanyal, A. 2021. Temporal-coded deep spiking neural network with easy training and robust performance. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, volume 35, 11143–11151.

Zhou, Z.; Zhu, Y.; He, C.; Wang, Y.; YAN, S.; Tian, Y.; and Yuan, L. 2023. Spikformer: When Spiking Neural Network Meets Transformer. In *The Eleventh International Conference on Learning Representations (ICLR)*.

Zhu, R.-J.; Zhao, Q.; Eshraghian, J. K.; and Li, G. 2023. Spikegpt: Generative pre-trained language model with spiking neural networks. *arXiv preprint arXiv:2302.13939*.

Zhu, R.-J.; Zhao, Q.; Zhang, T.; Deng, H.; Duan, Y.; Zhang, M.; and Deng, L.-J. 2022. TCJA-SNN: Temporal-Channel Joint Attention for Spiking Neural Networks. *arXiv preprint arXiv:2206.10177*.