Bishop: Sparsified Bundling Spiking Transformers on Heterogeneous Cores with Error-Constrained Pruning

Boxun Xu
Department of Electrical and Computer Engineering,
University of California, Santa Barbara
Santa Barbara, CA, USA

Yuxuan Yin
Department of Electrical and Computer Engineering,
University of California, Santa Barbara
Santa Barbara, CA, USA

Vikram Iyer
Department of Electrical and Computer Engineering,
University of California, Santa Barbara
Santa Barbara, CA, USA

Peng Li
Department of Electrical and Computer Engineering,
University of California, Santa Barbara
Santa Barbara, CA, USA

Abstract

Spiking neural networks(SNNs) have emerged as a promising solution for deployment on resource-constrained edge devices and neuromorphic hardware due to their low power consumption. Spiking transformers, which integrate attention mechanisms similar to those found in artificial neural networks (ANNs), have recently exhibited impressive performance. However, these models are large in size and involve high-volume computation in both time and space, posing significant challenges for efficient hardware acceleration. We present Bishop, the first dedicated hardware accelerator architecture and HW/SW co-design framework for spiking transformers that optimally represents, manages, and processes spike-based workloads while exploring spatiotemporal sparsity and data reuse. Specifically, we introduce the concept of Token-Time Bundle (TTB), a container that bundles spiking data of a set of tokens over multiple time points. Our heterogeneous accelerator architecture Bishop concurrently processes workload packed in TTBs and explores intra- and inter-bundle multiple-bit weight reuse to significantly reduce memory access. Bishop utilizes a stratifier, a dense core array, and a sparse core array to process MLP blocks and projection layers. The stratifier routes high-density spiking activation workload to the dense core and low-density counterpart to the sparse core, ensuring optimized processing tailored to the given spatiotemporal sparsity level. To further reduce data access and computation, we introduce a novel Bundle Sparsity-Aware (BSA) training pipeline that enhances not only the overall but also structured TTB-level firing sparsity. Moreover, the processing efficiency of self-attention layers is boosted by the proposed Error-Constrained TTB Pruning (ECP), which trims activities in spiking queries, keys, and values both before and after the computation of spiking attention maps with a well-defined error bound. Finally, we design a reconfigurable TTB spiking attention core to efficiently compute spiking attention maps by executing highly simplified "AND" and "Accumulate" operations. On average, Bishop achieves a 5.91× speedup and 6.11× improvement in energy efficiency over previous SNN accelerators, while delivering higher accuracy across multiple datasets.

ISCA '25, June 21–25, 2025, Tokyo, Japan

ACM ISBN 979-8-4007-1261-6/2025/06
https://doi.org/10.1145/3695053.3731063

Keywords

Spiking Neural Networks, Neuromorphic Accelerators, Transformers, HW/SW Co-Design

1 Introduction

As the third generation of neural networks [34], spiking neural networks (SNNs) exhibit a closer resemblance to biological neural circuits than their non-spiking artificial neural network (ANN) counterparts. SNNs hold promise owing to their biological plausibility, event-driven characteristics, and low power consumption [43]. The recent spiking-based transformer models have demonstrated superior performance compared to conventional spiking network architectures [2, 54, 60, 64, 65], mirroring the trend observed in ANNs where vision transformers outperform VGGs or ResNets. A number of hardware accelerators for ANN-based transformers have been proposed. For example, [7, 39] focus on more efficient processing of attention-based computations; [17, 59] propose techniques dealing with transformers exhibiting a targeted level of sparsity; [8] utilizes Taylor approximation to eliminate the complex non-local softmax functions in attention computation.

However, hardware acceleration of spiking transformers remains largely unexplored, giving rise to significant challenges related to computational overhead, latency, and energy consumption, especially when dealing with large spiking transformers. Continuing the use of hardware architectures designed for spiking fully connected (FC) or convolutional neural networks (CNNs) or adapting existing ANN-based transformer accelerators [33, 48, 62] falls short in harnessing the unique characteristics of spiking transformers.

This work presents the first dedicated hardware accelerator architecture and HW/SW co-design framework tailored for spiking transformers. We contend that the most promising approach to accelerate spiking transformer models is to optimally represent, manage, and process spike-based workloads while exploring spatiotemporal sparsity and data reuse. As illustrated in Fig. 1, this work makes the following contributions.

In the context of spiking vision transformers, we introduce the concept of spiking Token-Time Bundles (TTBs). A TTB is a container that bundles spiking data for a set of spatial tokens over a number of time points, capturing the temporal variations of these tokens. The TTB serves as the fundamental unit of work to be processed on our proposed Bishop architecture. The utilization of

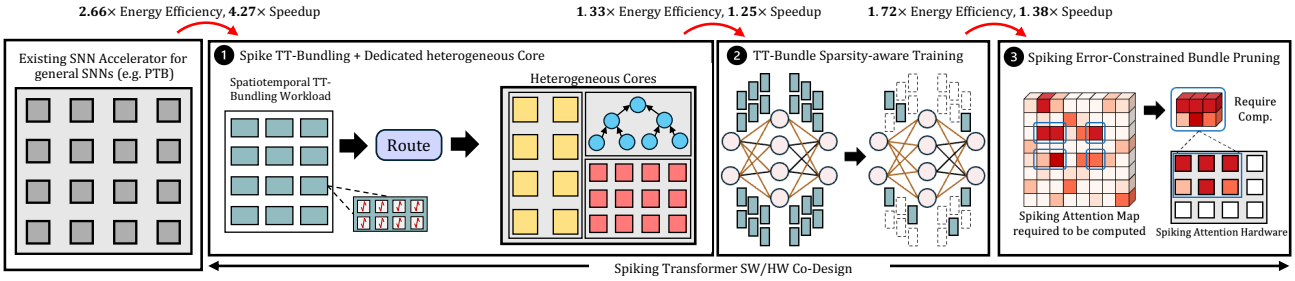

Figure 1: Bishop: the first accelerator architecture and SW/HW co-design framework dedicated to spiking transformers.

TTBs empowers us to harness structured data reuse and firing sparsity inherent in key computations within the transformer across both time and space.

Our Bishop architecture ❶ is heterogeneously comprised of a TTB stratifier, a TTB dense core, a TTB sparse core, and a TTB attention core. The TTB stratifier efficiently routes high-density spiking-TTB workloads to the TTB dense core and directs low-density workloads to the TTB sparse core, ensuring optimized processing tailored to the present spatiotemporal sparsity level of the workload. Taking advantage of the binary nature of spiking TT-bundled queries and keys, we design a reconfigurable multiplier-less And-ACcumulate (AAC) array. This array efficiently processes spiking attention layers by executing highly simplified "AND" and "Accumulate" operations and optimized dataflow, achieving a 2.66× improvement in energy efficiency and a 4.27× speedup over the Parallel Time Batching (PTB) architecture of [27].

From the algorithmic perspective, we explore firing sparsity tailored for hardware as a key opportunity for efficient acceleration. To this end, we introduce ❷, a TTB sparsity-aware training pipeline that not only enhances the overall sparsity of spiking activations but does so in a structured manner by organizing them into TTBs for efficient processing on Bishop. Furthermore, we propose ❸, an Error-Constrained TT-Bundle Pruning (ECP) algorithm to eliminate redundant operations across tokens of spiking queries, keys, scores, and values with a well-defined pruning threshold, facilitating efficient attention computation in conjunction with ❶ while maintaining high attainable performance.

2 Background

2.1 Spiking Neural Networks

Leaky Integrate-and-Fire (LIF) Model. We adopt the widely used leaky integrate-and-fire neuronal model in SNNs [20], which has the following discretized dynamics over time:

$$V_m[t_k] = V_m[t_{k-1}] + I[t_k] - V_{leak} \tag{1}$$

$$S[t_k] = \begin{cases} 1 & \text{if } V_m[t_k] > V_{th} \rightarrow V_m[t_k] = 0 \\ 0 & \text{else } \rightarrow V_m[t_k] = V_m[t_k] \end{cases} \tag{2}$$

where $V_m[t_k]$ is the membrane potential of a spiking neuron at time point t_k; $I[t_k]$ is the total synaptic input current at t_k; V_{leak} is the leakage; V_{th} is the firing threshold, and $S[t_k]$ is the spiking output. A spiking output is generated if the membrane potential $V_m[t_k]$ exceeds V_{th}, setting $S[t_k]$ to 1 and resetting $V_m[t_k]$ to 0.

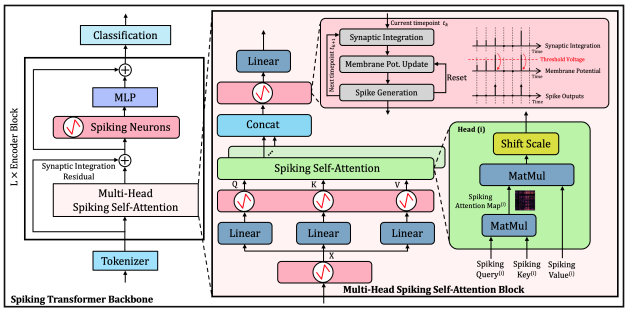

Figure 2: Spiking transformer model architecture with multi-head spiking self-attention [53, 64].

Spiking Transformers. The recent emergence of spiking transformers presents a major development in spiking neural networks [49, 54, 60, 64, 65], showing large performance improvements over other SNNs [44, 45] such as spiking CNNs [19, 24, 29, 42]. Specifically, the state-of-the-art spiking transformers incorporate a multi-head Spiking Self-Attention (SSA) mechanism [53, 54], capturing global correlations among input tokens in a spike form. This approach demonstrates impressive performance with promising energy efficiency in vision applications compared with ANNs under the same model size. Tab. 1 compares the accuracy of spiking transformers with other state-of-the-art ANNs and SNNs across various image[12], gesture[1] and speech command[50] recognition datasets.

As depicted in Fig. 2, spiking transformers incorporate a spiking tokenizer, which transforms a static image or a dynamic input, e.g, a dynamic vision sensor (DVS) stream, represented as an input sequence $I \in \mathbb{R}^{T\times C\times H\times W}$ with spatial size $H \times W$ in C channels across T time points, into $I' \in \mathbb{R}^{T\times N\times D}$, i.e., N D-dimensional tokens across T time points. The core model architecture comprises L sequentially connected residual encoder blocks with each consisting of a multiple-head Spiking Self-Attention (SSA) block and a spiking MLP block. The output from the final encoder block is

Table 1: Comparison of ANN and SNN Accuracies.

Workload	ANN Accuracy	SNN Accuracy
CIFAR10	ResNet-19[15]: 94.97% Transformer[16]: 96.73%	VGG-11[42]: 92.22% CIFARNet[61]: 91.41% ResNet-19[63]: 92.92% ResNet-20[41]: 92.54% Spiking Transformer: **95.19%**
CIFAR100	ResNet-19[15]: 75.35% Transformer[16]: 81.02%	VGG-11[42] 67.87% ResNet-19[63]: 70.86% ResNet-20[41]: 64.07% Spiking Transformer: **77.86%**
DVS-Gesture	12-layer CNN: 94.59%[1]	CIFARNet [61]: 91.32% ResNet-19[63]): 96.9% Spiking Transformer: **98.3%**
ImageNet	ResNet-34[45]: 70.69% Transformer[16]: 80.8%	Spiking ResNet-34[15]: 64.79% SEW-ResNet-50[19]: 67.78% Spiking Transformer: **73.38%**
Google SC	ConvNet[21]: 87.0% AttentionRNN[10]: 93.9%	Spiking ResNet[52]: 92.9% Spiking Transformer: **95.11%**

processed with a global average pooling across all tokens and time points, and is subsequently employed to make the final prediction through a linear layer[16].

Spiking Self-Attention. Unlike conventional spiking CNNs, spiking transformers feature the use of SSA blocks for extracting global token dependencies. The SSA blocks in our proposed Bishop architecture operate as follows:

$$
\begin{aligned}
Q^{(i)} &= \mathcal{LIF}(X^{(i)} \cdot W_Q^{(i)}) \quad (\\
K^{(i)} &= \mathcal{LIF}(X^{(i)} \cdot W_K^{(i)}) \quad (\\
V^{(i)} &= \mathcal{LIF}(X^{(i)} \cdot W_V^{(i)}) \quad (\\
O^{(i)} &= (Q^{(i)} \cdot K^{(i)T} \cdot s) \cdot V^{(i)} \quad (\\
O_{temp} &= \mathcal{LIF}(Concat\{O^{(1)}, ..., O^{(H)}\}) \quad (\\
O_{attn} &= (O_{temp} \cdot W_O), \quad (
\end{aligned}
$$

where X represents the binary inputs produced by the precedin LIF neurons, and H denotes the number of heads. For each i-th hea $Q^{(i)}, K^{(i)}, V^{(i)}, O^{(i)}$ represent the query, key, value, and outpu respectively; $W_Q^{(i)}, W_K^{(i)}, W_V^{(i)}$ are the weights of the correspond- ing linear projection layers for computing $Q^{(i)}$, $K^{(i)}$, and $V^{(i)}$, respectively. $\mathcal{LIF}$ refers to an LIF neuron layer. s is a power-of- two scaling parameter, allowing for efficient scaling via bit shifting. W_O represents the weights of the final linear projection layer, and O_{attn} is the output from the H-head SSA block. In contrast to the method described in [64] which extensively employs multipliers due to spike residuals, we reposition the final LIF neuron layer to precede the last linear layer of each SSA block[53] as in (7). This adjustment allows for efficient multiplication-free computation of attention output O_{attn} based on the product of the weights W_o with the binary LIF activations O_{temp} in (8).

2.2 Complexity/Workload Profiling of Spiking Transformers

Computational Complexity. We analyze the computational complexity of a spiking transformer. As shown in Fig. 2, the MLP block, the four (Q, K, V, O) linear projection layers, and attention layers are mostly dominant sources of complexity. In the MLP block and linear projection layers of a multi-head SSA block, the spike inputs are multiplied with the corresponding weights to produce the synaptic inputs to the LIF neuron units, which generate firing outputs. The space and time complexity of these layers is $O(TND^2)$. The attention layers within an attention block compute the dot-product of spiking queries and keys with a computational complexity of $O(TN^2D)$. With $N \gg D$, the complexity of the attention layers dominates that of the MLP/projection layers. The opposite is true with $D \gg N$. The LIF neuron layers have a non-dominant complexity of $O(TND)$. The tokenizer's computational complexity is $O(THWC^2K^2)$, where K is the size of the employed spiking convolutional (CONV) filters. It is typically not the dominant complexity, and there has been much work targeting hardware acceleration of spiking CNNs [27, 37].

Workload Profiling. To shed more light on the inference complexity of spiking transformers, we perform profiling and provide a FLOP breakdown for a spiking transformer based on the architecture of [64] and trained on the ImageNet dataset. Fig. 3 reveals that the primary computational burdens reside within the spiking attention (Attn) and MLP blocks, especially in deeper spiking transformers. In addition, the dominance of attention blocks over MLP blocks intensifies as N increases. The cumulative FLOPs attributed to these blocks range from 66.5% to 91.0% of the total workload. Hence, attention and MLP blocks are the primary target of the

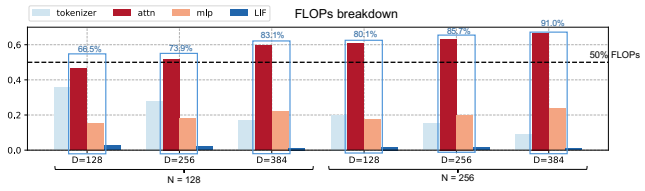

Figure 3: The FLOPs breakdown of a spiking transformer with different token and feature sizes trained on ImageNet.

2.3 Existing SNN accelerators

There exists a body of SNN accelerators focusing on enhancing inference energy efficiency or latency through the employment of novel devices [46], circuits [32], processing architectures [26, 27, 35, 37, 55] and on-chip communication networks [9, 11, 23, 25, 31], or on improving training efficiency [30, 47, 57]. These efforts predominantly target spiking CNNs such as spiking-based AlexNet/VGG/ResNets [19, 28, 63]. [22] extends traditional ANN transformer models by enabling them to operate under both integer and rate-encoded spiking modes. However, these architectures are not optimized for large spiking transformers.

We identify four major challenges and opportunities with respect to hardware acceleration of spiking transformers. Firstly,

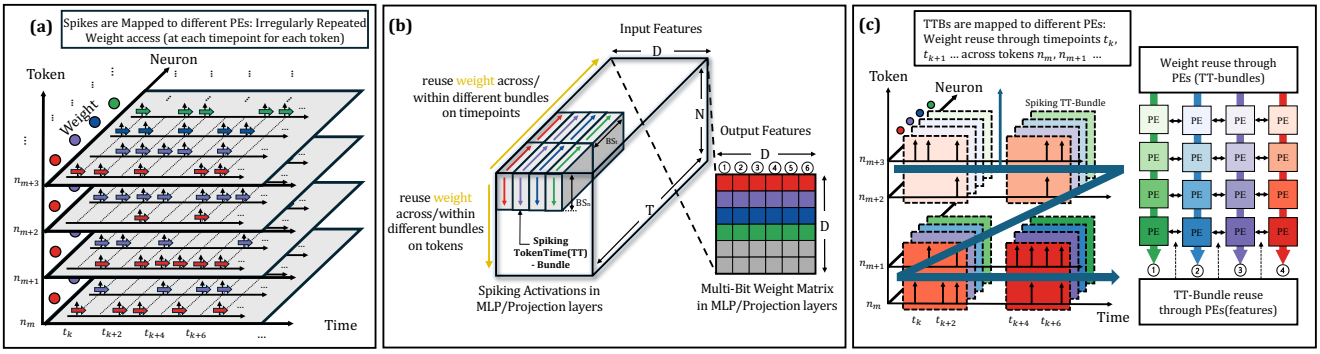

Figure 4: (a) The conventional approach lacks parallel processing in time and space, processing each token at each timepoint in a time-serial manner, causing irregularly repeated weight accesses. (b) The spiking Token-Time (TT)-Bundle allows for multi-bit weight data reuse across multiple tokens and time points. (c) Weight reuse in both an intra- and inter-bundle manner. Multiple TT bundles of neurons corresponding to different output features are mapped onto different PE columns. Multiple TT bundles across different input features marked as different color depths are mapped onto different PE rows.

given the excessive use of multi-bit weight data in spiking transformers and the fact that spiking activations are single-bit, it is crucial to maximize weight reuse both temporally and spatially. While spiking CNN accelerators [26, 27] attempt to reduce weight data access, they are limited to exploiting weight reuse only in the temporal dimension within a systolic array. Instead, a proper packing of the spatiotemporal spiking workloads optimized for the computational structure within spiking transformers can lead to structured sparsity and improve weight reuse across both time and space. Secondly, existing SNN accelerators [27, 37, 46, 47] do not explore a heterogeneous architecture, leading to large efficiency loss when encountering workloads with a varying degree of structured or unstructured sparsity. Thirdly, the high performance of recent spiking transformers can be largely attributed to the new spiking attention mechanisms adopted [2, 54, 60, 64, 65]. Nonetheless, dedicated architectural support for these mechanisms remains largely absent. Finally, while algorithm/hardware co-optimization such as weight pruning [4, 40], temporal pruning [5], and quantization [56] have demonstrated effectiveness for spiking CNNs, a systematic approach to accelerator architecture design and algorithm/hardware co-optimization is currently lacking for spiking transformers. Focusing on 3D physical design, [51] proposes a simple spiking transformer architecture that does not pack spatiotemporal tokens into bundles, thereby missing opportunities for effective weight data reuse and for supporting temporal and token-level parallel processing within each processing unit, nor does it exploit any form of spiking sparsity or adequately consider HW/SW co-design.

To this end, we propose `Bishop`, which to the best of our knowledge is the first accelerator architecture and HW/SW co-design framework dedicated to spiking transformers. It's designed to fully exploit irregular sparse firing patterns and maximize data reuse on a heterogeneous array architecture. Additionally, it incorporates a sparsity-aware training pipeline and error-constrained pruning tailored for spiking transformers. These approaches significantly reduce processing latency and enhance energy efficiency, effectively addressing critical computational and data access bottlenecks in state-of-the-art spiking transformers.

3 Spiking Token-Time (TT) Bundle

3.1 Challenges in Managing Spatiotemporal Workloads

Spatiotemporal workloads should be appropriately packed to facilitate efficient processing. In [26] a spiking systolic array is proposed to spatially process different postsynaptic neurons in parallel and temporally process the input integration of each neuron over several sequential time steps. However, the lack of time-parallel processing exacerbates the overhead of accessing expensive multiple-bit, e.g., 8 bits, weight data. The systolic-array-based Parallel Time Batching (PTB) architecture in [27] addresses this issue by packing spiking activities across multiple time points within a time window and processing several such windows concurrently on the systolic array, thereby improving weight data reuse.

However, both works only support the computations of convolution neural networks (CNNs) and fully-connected layers (FCs) without addressing the unique computation structures of spiking transformers. For instance, FC layers perform matrix-vector multiplications on spiking activtations with a vector length equal to the number of neurons. On the other hand, the main computations in spiking transformers have a very different spatiotemporal computation structure. MLP layers in transformers perform matrix-matrix multiplications on spiking activations in the dimension of N (tokens) $\times$ D (features) across T time points based on weights whose dimension is $D \times D$. Furthermore, SNNs may operate over a wide range of timesteps. Packing workloads solely in time as in the PTB architecture [27] is only effective when the number of timesteps is

large, e.g., 100-300. The number of operational timesteps of spiking transformers may vary from 300 [49] down to 4 [60, 64]. Operating with a reduced number of time steps significantly reduces both inference latency and training costs, marking a trend driven by advances in SNN training algorithms. To maintain flexibility in hardware architectures, there is a need for supporting spatiotemporal computation within spiking transformers across a wide range of time horizons, with a specific emphasis on facilitating computation over shorter time spans.

3.2 Proposed Spiking Token-Time (TT) Bundle

The most promising approach to accelerate spiking transformer models is to optimally represent, manage, and process spike-based workloads while exploring spatiotemporal sparsity and opportunities for data reuse. We introduce the concept of spiking Token-Time Bundle (TTB), a container that bundles spiking data for a set of tokens over a set of time points, specifically for spiking transformers. As shown in Fig. 4, we split the spiking activities of N tokens, D features, across T time points into multiple token-time bundles. Each TTB packs BS_n tokens across BS_t timepoints for a given output feature. Accordingly, we have $\lceil T/BS_t \rceil \times \lceil N/BS_n \rceil$ token-time bundles. The utilization of TTBs provides two major benefits in data reuse and processing.

3.2.1 Multi-bit Weight Data Reuse. Without data packing in TTBs, an accelerator irregularly and repeatedly loads weight data to process individual input spikes of a token at a time point, discarding much of the opportunities in data reuse, as shown in Fig. 4(a), where processing of each spike is mapped onto a PE. In contrast, the proposed Bishop architecture maps the processing of a TTB onto a PE, and exploits weight data reuse at multiple levels as shown in Fig. 4(b). Because the same multi-bit weights are used for different time points and different tokens for each feature, the $1 \times D$ weight data in one row, for example, the highlighted red row, is shared within each TTB for processing BS_n tokens across BS_t time points. In addition, the same weight data is reused across $\lceil T/BS_t \rceil \times \lceil N/BS_n \rceil$ TTBs. As such, Bishop simultaneously exploits both intra-TTB and inter-TTB weight data reuse. In addition, the same input activiations are shared within each PE row.

3.2.2 Explore structured TTB-Level Sparsity. Furthermore, we tag all TTBs and categorize them into two groups. A TTB is called *active* if there exists at least one active spike in the bundle. Otherwise, we call it *inactive*. Inherent spatiotemporal sparsity of spiking activities can be leveraged to skip processing inactive TTBs, which constitute of a bulk of all bundles. Since Bishop packs and dispatches workloads in terms of TTBs, skipping inactive bundles can be efficiently accommodated in the dataflow and avoids large overheads resulting from skipping computations at a much smaller granularity, e.g., at the spike or token level. The effectiveness of this approach is further enhanced by the Bundle-Sparsity Aware Training (BSA) algorithm discussed in Section 4.1.

4 Proposed Spiking Transformer Algorithms

We present two algorithms, namely spiking TT Bundle-Sparsity Aware Training (BSA) and Error-Constrained TT Bundle Pruning (ECP) to optimize bundle-level sparsity and prune spiking activities in queries and keys, respectively. ECP is also integrated into the training pipeline, leading to ECP-aware training to maintain high accuracy. BSA and ECP provide an end-to-end algorithm and hardware co-optimization approach for the proposed Bishop architecture.

4.1 Bundle-Sparsity Aware Training (BSA)

We visualize the distribution of active token-time bundles (TTBs) in spiking transformers in Fig. 5(a). Without applying BSA, spiking transformers exhibit moderate TTB-level sparsity. For instance, in Model 1 of Table 2, 29% of the bundles are active across all layers, offering some restricted opportunities for computation skipping.

Unlike suppressing activities at the individual spike level as in [13], the proposed Bundle-Sparsity Aware Training (BSA) algorithm sparsifies spiking activities at the TTB level, i.e., by reducing the number of active TTBs, thus providing a structured way of exploiting sparsity.

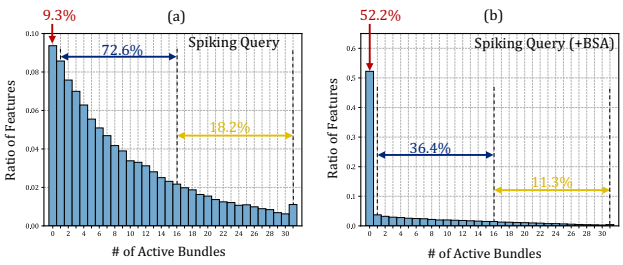

Figure 5: Active bundle distribution of spiking queries (Q) across each input feature in the 4^{th} encoder block of a spiking ViT (Model 1) trained on CIFAR10: (a) without BSA, and (b) with BSA.

In BSA, we denote the spiking activations at time step t in layer l, token n, and input feature dimension d by $X^{(l)}_{n,t,d}$, and a bundle for layer l by $TTB^{(l)}_{bn,bt,d}$, where bn, bt, and d are the bundle-token, bundle-time, and input feature index, respectively. While $TTB^{(l)}_{bn,bt,d}$ packs multiple tokens and time steps, its activity tag $Z_{bn,bt,d}$ is defined by the L_0 norm of the spiking activations that fall within it:

$$Z_{bn,bt,d} = ||X_{bn\cdot BS_n:(bn+1)\cdot BS_n-1,bt\cdot BS_t:(bt+1)\cdot BS_t-1,d}||_0 \quad (9)$$

$Z_{bn,bt,d}$ is zero if there is no active spike in the bundle. We sum up all bundle tags across all D features and L layers of the transformer to define a bundle-level sparsity loss L_{bsp}:

$$L_{bsp} = \sum_{l=0}^{L-1} \sum_{bn=0}^{\lceil N/BS_n \rceil -1} \sum_{bt=0}^{\lceil T/BS_t \rceil -1} \sum_{d=0}^{D-1} Z_{bn,bt,d} \quad (10)$$

When forming the above L_{bsp} loss, we consider activations from all MLP and linear projection layers as well as the bundles associated with the queries (Q) and keys(K) in the attention layers. BSA optimizes the model weight parameters $\boldsymbol{\theta}$ by minimizing a total loss: $L_{tot} = L_{CE} + \lambda L_{bsp}$, which jointly considers a cross-entropy model accuracy loss L_{CE} and L_{bsp} with a hyperparameter λ, controlling the tradeoff between the accuracy and TTB-level sparsity.

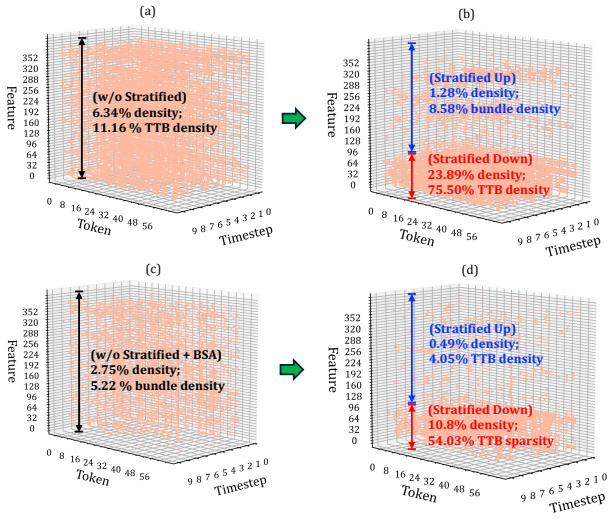

Figure 6: Spiking activities at the output projection layer in the 3^{rd} encoder block of a spiking ViT (Model 1) trained on CIFAR10: (a) the original workload without BSA training, (b) stratified TTBs without BSA , (c) the sparsified workload with BSA, and (d) stratified TTBs with BSA.

Fig. 5 visualizes the original and BSA-altered distributions of the active bundles of Model 1, defined in Table 2. Specifically, BSA significantly improves TTB sparsity and reshapes their distribution across input features, leading to most features having only a small number of active bundles. In addition, BSA leads to a substantial increase in the percentage of input features (from 9.3% to 52.2% in Model 1, as shown in Fig. 5) that have no active TTBs, further facilitating structured pruning of their associated weights. Fig. 6 further shows the joint effects of stratification and BSA.

5 Proposed Bishop Architecture

5.1 Error-Constrained TT Bundle Pruning (ECP)

The performance of spiking transformers critically relies on the computation of self-attention maps, whose complexity is quadratic in the number tokens N. As N increases, the computational overhead of using spiking queries, keys and values to compute increasingly larger spiking attention maps and subsequent computation based on (6) becomes a dominant bottleneck. Models trained over ImageNet, for example, may have $N = 192$ tokens and a fewer number of features, i.e., $D = 128$, making the complexity of the spiking self attention (SSA) layers dominate that of the MLP/projection layers. The number of tokens can be even much greater in many long-sequence tasks [6]. We have identified that over 30% of computational overhead comes from SSA layers in typical spiking transformers. The proposed Error-Constrained TT Bundle Pruning (ECP) algorithm prunes away binary activations in the queries (Q) and keys (K), which further triggers a large amount of structured computational reduction in the resulting attention map (S), the values (V), and the outputs (Y) as illustrated in Fig. 7.

Notably, ECP explores a key property of the spiking self-attention mechanism that is not present in its ANN counterpart for effective pruning while ensuring a well-controlled error bound. In ANNs, bounding the scores (S) given the queries (Q) and keys (K) is difficult because Q and K are continuous-valued floating point numbers. Differently, we explore the binary nature of the spiking Q and K for a more straightforward bounding of S without computing them. The total number of active bundles n_{ab} in a particular bundle row across all features of the Q tensor can be easily obtained from the active bundle tags. If n_{ab} is less than a threshold $\theta_{p,Q}$, due to the binary nature of K, it is certain that any activation in a bundle that is in the corresponding row of the scores (S) tensor would be less than $\theta_{p,Q}$ per $S = QK^T$. We prune out this row from the Q tensor entirely while limiting the error to be no greater than $\theta_{p,Q}$. In practice, the actual pruning error can be lower than $\theta_{p,Q}$ because the K tensor is sparse and its sparsity pattern does not necessarily coincide with that of Q. Moreover, we adopt the same process to prune the K tensor with a user-specified error bound $\theta_{p,K}$.

As illustrated in Fig. 7, pruning Q and K tensors in the above manner has a compounding effect. If 20% and 10% of the Q and K bundle rows, respectively, remain after the application of ECP, 80% of rows and 90% of columns in the attention map will be pruned away, reducing the overhead of computing S down to 2%. This further reduces data access to the V tensor and ultimately decreases the writeback overhead of the Y tensor. Incorporating ECP into training does not necessarily degrade model accuracy. In many cases, ECP can slightly improve model accuracy by acting as a denoising mechanism. Fig. 8 visualizes the attention maps from the final transformer block of a spiking transformer model trained on ImageNet-100, illustrating how ECP enhances focus on important regions of an input image.

Fig. 9 shows the proposed heterogeneous Bishop architecture, which consists of a hierarchical memory system and three main processing cores, targeting the three computational bottlenecks identified in Section 2.2. The TT-Bundle Sparse Core and TT-Bundle Dense Core are dedicated to computing synaptic input integration in the MLP and linear projection layers. The outputs from the two cores are combined by the sparse-dense addition module within the spike generator to compute the final membrane potential V_{mem} for each spiking neuron, which is then used to conditionally generate output spikes. The TT-Bundle Attention Core is an efficient engine for accelerating spiking self-attention layers. The output from the TT-Bundle Attention Core is fed into the Spike Generator to produce spike-form attention outputs per (8).

5.2 Motivation of the Proposed Architecture

Bishop is designed to address the following key issues.

Stratified workloads for heterogeneous processing. Bishop stratifies sparse and dense workloads for processing on the heterogeneous cores. Fig. 10(a) shows that typical spiking spatiotemporal workload is a mix of dense and sparse spiking activities with different sparsity levels on each feature dimension, prohibiting efficient processing. Thanks to our bundle sparsification, we can stratify the workload into dense and sparse parts, as visualized in Fig. 6, to be dispatched to the dense and sparse core, respectively, as in Fig. 10(b). This significantly improves hardware utilization and avoids scheduling difficulties and inefficiencies that arise from using sparse cores for dense computations, and vice versa.

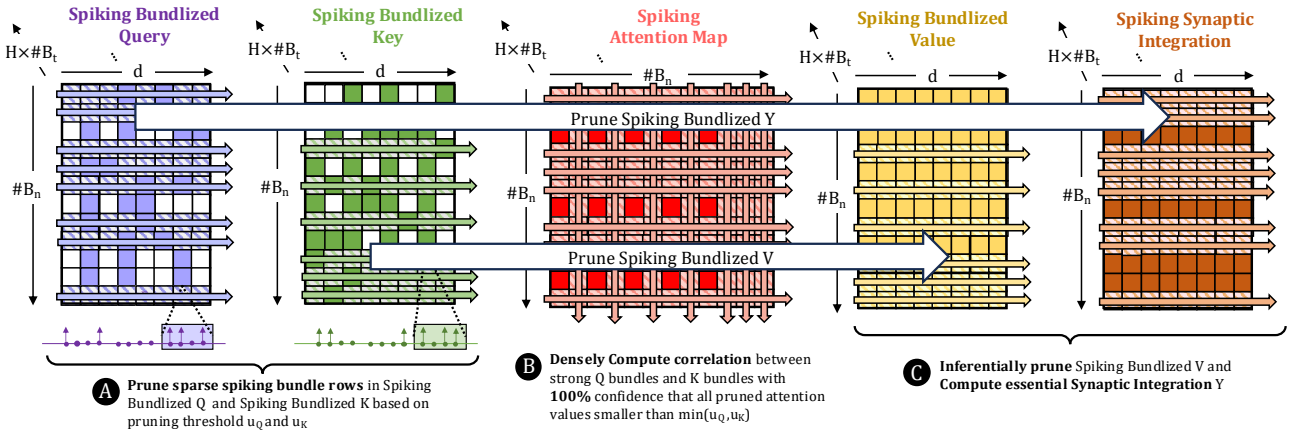

Figure 7: Error-constrained TT bundle pruning (ECP) for queries (Q), keys (K), attention scores (S), values (V), and outputs (Y).

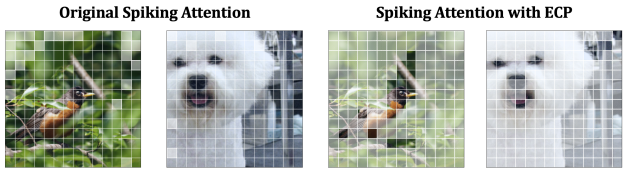

Figure 8: **The impact of ECP on the computed attention.**

Structured weight reuse based on TTBs. Packing the workload into TTBs supports multi-bit weight reuse and avoids repeated weight accesses when dealing with irregular spiking patterns on the dense and sparse cores, and allows for key/query data reuse on the spiking attention core.

Tailored core for spiking attention computation. The multi-bit self-attention computation in ANN-based transformers is expensive and the hardware accelerators bulit on them do not exploit the unique properties of spiking self attention. To efficiently compute spiking self-attention maps and outputs from binary spike inputs in (6), we propose a reconfigurable systolic array, comprising AND gates, multiplexers, and accumulators as a customized attention core that eliminates the need for costly multi-bit multipliers and significantly reduces area and power overhead. Furthermore, we employ the feature-first tiling to realize an optimized score-stationary dataflow to reduce data movements of multi-bit attention scores while flowing the binary query/key/value data onto the array.

5.3 Stratify Sparse and Dense Workload

Fig. 10 illustrates how the spiking workload is stratified into the dense and sparse parts, with the feature indices recorded to align rows in the weight matrix so that the Dense TT-Bundle Core processes $X_D W_D$ and the Sparse TT-Bundle core computes $X_s W_s$. As detailed in Alg. 1, the stratifier compares the number of active TTBs for each input feature against a stratification threshold to classify the feature to be either “dense" or “sparse". A feature index buffer is employed to track the indices of both sparse and dense features. Subsequently, the coordinated weights and input bundles are fed onto the dense/spare core for processing.

Algorithm 1: Stratifying spiking bundles across different input features.

Input: Spiking bundled workload: $X \in \mathbb{R}^{B\times D}$, Weight: $W \in \mathbb{R}^{D\times D}$, total number of bundles: B, total number of input features: D, stratifying column-sparsity threshold: θ_s
Output: Floating low-density spiking bundles X_S and the associated weight W_S, sinking high-density spiking bundles X_D and associated weight W_D
parallel for $i = 0; i < D; i$++ **do**
  if $\sum_{j=0}^{B-1} ||S_{j,i}||_0 > \theta_s$ **then**
    append i **to** R_D; //store dense feature indexes to access permuted weights and spiking bundles
  end
  else
    append i **to** R_S; //store sparse feature indexes to access permuted weights and spiking bundles
  end
end
$X_D, W_D \leftarrow X_{:,\in R_D}, W_{\in R_D,:}$; //routed to dense TTB core;
$X_S, W_S \leftarrow X_{:,\in R_S}, W_{\in R_S,:}$; //routed to sparse TTB core;
return X_D, X_S, W_D *and* W_S

5.4 Dedicated Dense and Sparse Cores

TT-Bundle Dense Core. The stratified dense workload in the MLP and projection layers is directed to the TT-Bundle Dense Core, which employs an output-stationary architecture, reminiscent of a TPU-like systolic array, as depicted in Fig. 9. Each Token-Time bundle is assigned to a distinct processing element (PE). These PEs process and pass spiking bundles in a top-to-bottom sequence.

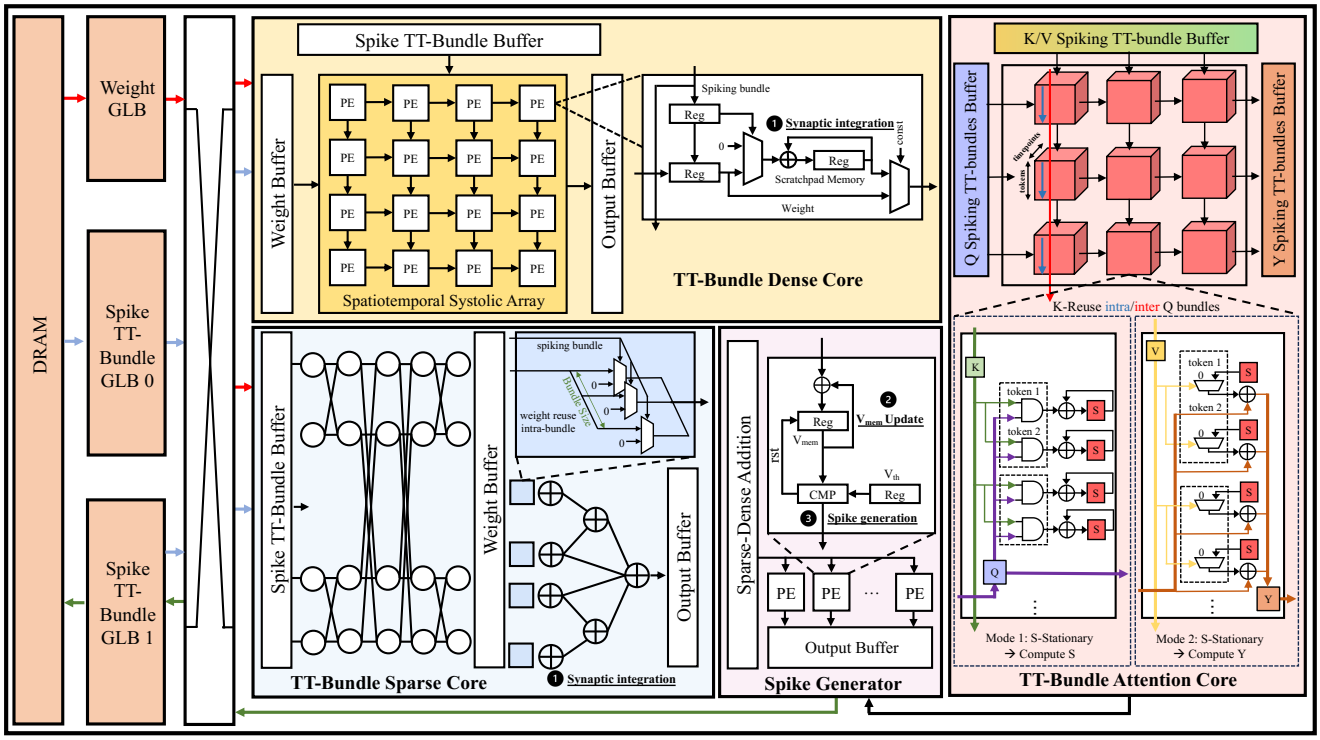

Figure 9: The overall heterogeneous Bishop Architecture.

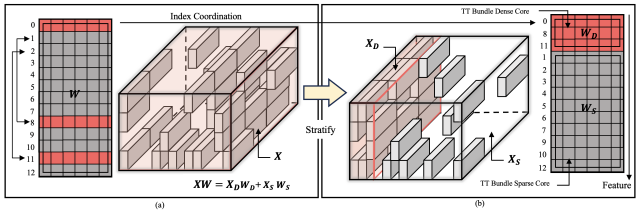

Figure 10: (a) The original spiking workload X. (b) The stratified dense workload X_D and sparse workload X_S; weight indices are coordinated to dispatch W_D and W_S onto the TT-Bundle dense/TT-Bundle core, respectively.

Concurrently, the coordinated weights are passed from left to right across the array, where each PE column computes a specific output feature. In this process, weight data is repeatedly reused within a bundle in an intra-bundle reuse manner, and reused by all PEs in the same row in an inter-bundle reuse manner. Leveraging the binary nature of spiking inputs, each PE executes "Select ACcumulation" (SAC) operations to effectively multiply synaptic weights with spiking inputs. A "SAC" operation is efficiently implemented by one MUX and one accumulator. The partial sums are stored in PEs' local registers. Upon completion of the computation assigned to the PE array, the partial sums of synaptic integration are buffered into the output buffer, awaiting merging with the results from the TT-bundle sparse core.

TT-Bundle Sparse Core. Working in parallel with the TT-Bundle Dense Core, the TT-Bundle Sparse Core processes the sparse part ($X_S W_S$) of spiking synaptic integration. We design our TT-bundle sparse core by adopting a SIGMA-like architecture [38] to efficiently handle irregular sparsity patterns, thanks to its flexible and configurable distribution and reduction network. The core utilization is enhanced considerably by the proposed BSA training algorithm presented in Section 4.1, which improves the network's structured bundle-level sparsity. In addition, this core facilitates multi-bit weight data reuse when processing different tokens and time points within a bundle as clustered firing activities may take place in the bundle. Finally, we merge and accumulate the partial sums of the synaptic integration streaming out of the sparse and dense cores by performing sparse-dense addition in the Spike Generator. The Spike Generator maps neurons to be processed onto different PEs, where an output spike is conditionally generated if its V_{mem} exceeds the threshold voltage V_{th} at certain time point.

5.5 Dedicated TT-Bundle Attention Core

Attention computation in ANN transformers is costly. It involves a sequence of multi-bit multiplications between queries (Q) and keys (K), non-local softmax operations, and multi-bit multiplications of the resulting scores (S) and values (V). As described in Section 2.1, the spiking attention mechanism employs binary tensors Q, K, and V. This, coupled with the elimination of computationally expensive softmax operations, provides a distinct advantage. The proposed attention core efficiently computes spiking attentions by leveraging

the binary nature of the Q/K/V data through two-step operations. The first step computes an accumulated attention map by multiplying Q with K^T to produce attention score S; the second step is to compute the output Y by multiplying S with V. Thanks to our ECP algorithm, there only exist a limited number of $Q/K/V$ bundles that need to be loaded and processed. This selective processing allows the attention array to be dynamically allocated, solely executing the essential computations.

The attention core utilizes a systolic array while employing optimized dataflows and data-reuse schemes. As shown in Fig. 9, each PE has two operation modes. Mode 1 utilizes the flowing Q and K data to compute S with an S-stationary dataflow. Mode 2 utilizes the stationary S and flowing V to compute Y. In each PE, the workloads at different time points are mapped onto different groups of logic gates circled by the dotted line. Multiple groups simultaneously process the same set of tokens for different time points.

Mode 1. We configure the core into multiple **A**nd-**AC**cumulate (AAC) units to efficiently compute the accumulated attention on different tokens at each time point, and store the resulting partial sums in the S registers. S-stationary dataflow: Because the attention score S has a higher bit width, ranging from 6 to 10 model-dependent bits, than the binary spikes in Q and K, we adopt an S-stationary dataflow to minimize data movement. This approach involves the directional flow of Q bundles from left to right, coupled with the streaming of K bundles from top to bottom across the array. This setup executes "AND" operations between binary queries (Q) and keys (K), followed by the accumulation of partial sums in the S register within each PE. The process continues until the computation of S is completed for all features. K-reuse with intra/inter-Q bundles: K bundles tend to have a higher token sparsity than Q, especially after our error-constrained bundle pruning. We reuse one set of K-tokens that correspond to the same spatial location across multiple time points to operate with B_n tokens from one Q bundle in each PE. Specifically, at a given processing time, each PE contains one Q bundle, and the set of K-tokens is reused for the computation with all tokens in that Q bundle in an intra-Q-bundle manner. Across various Q bundles mapped onto the PEs in the same column, the K-token set is reused again in an inter-Q-bundle manner.

Mode 2. The attention core is configured into **S**elect-**AC**cumulation (SAC) units based on the S-stationary dataflow when computing Y. V-reuse with intra/inter-S bundles: As shown in Fig. 9, we maintain S in the local registers and load V from top to bottom, and flow the computed partial sums Y from left to right. In each "SAC"operation, the binary V input selects the right S data to be accumulated into the partial sum, which is stored in the Y register. In a fashion that is similar to the K-reuse described above, V is reused in both an intra- and inter-S bundle manner. In each PE, V is reused for processing different S tokens for multiple time points to facilitate the intra-S bundle reuse. The same V data is reused again in an inter-S bundle manner across the PEs in the same column. When Y is read out, it is aggregated into the partial sum maintained within the Y TT-bundle buffers. Upon completion of Y's computation, with Y represented in an integer format, a shifter is employed to adjust the scale of Y per (6) based on a power-of-two scaling factor s. The processed Y is then fed into the spiking generator, and the final binary attention output is stored back into the TTB GLBs.

6 Evaluation

6.1 Evaluation Setup

Model Training and Datasets. We develop a training flow based on Pytorch, integrating the proposed Bundle-Sparsity Aware Training (BSA) and Error-Constrained TTB Pruning (ECP) algorithms. Several spiking transformer models are trained on widely adopted image recognition datasets including CIFAR10, CIFAR100, ImageNet-100[12], the neuromorphic dynamic vision sensor (DVS) dataset DVS-Gesture-128 [1], and speech command recognition dataset Google Speech Command V2 [50].

To evaluate the scalability of our proposed Bishop architecture, we train multiple spiking transformers with different model architectures, following the settings in Tab. 2. In the case of CIFAR10 and CIFAR100, each image is sized at 32×32, which is segmented into 64 (N) tokens, with each token representing a 4×4 pixel area. The feature size (D) is set to 384. The fact that $D >> N$ renders the MLPs and linear projection layers as the dominant contributors to computational complexity. An ImageNet-100 image has a resolution of 224×224 pixels per channel. We split each image into 196 tokens, each representing a 16×16 pixel area. The feature size is set to 128, making the attention layers the most dominant source of computational complexity because $N > D$. For the neuromorphic dataset DVS-Gesture-128, the visual input at a time step comprises 128×128 pixels, which is divided into 64 tokens, each sized at 16×16. We use a batch size of 256 for 300 training epochs on the CIFAR and ImageNet-100 datasets, and a batch size of 64 for 100 epochs on the DVS-Gesture-128 dataset. Advances in training algorithms have resulted in high-accuracy SNNs operating on a reduced number of time steps. Different from the prior work [27, 37, 47] that employ 100 to 300 time steps, we limit the number of time steps to be from 4 to 20. This reduction contributes to a lower training cost and aligns with the trend in developing high-performance spiking neural models [61, 64] for low-latency processing.

In implementing BSA, we configure the parameter λ differently for various datasets: 1 for CIFAR10, 0.5 for CIFAR100, 0.3 for ImageNet, and 1.0 for DVS-Gesture-128. This choice allows us to strike a balance between bundle sparsity and accuracy tailored for each dataset. For error-constrained pruning of queries and keys in attention layers, we set the bundle pruning thresholds to 10 for models trained on DVS-Gesture-128 and 6 for other models without compromising accuracy.

Baselines and Evaluation Metrics. Baselines: To benchmark the proposed Bishop architecture, we compare it with an edge GPU (NVIDIA Jetson Nano) and Parallel Time Batching (PTB), a recent competitive SNN accelerator architecture[27]. For a fair comparison, PTB and the proposed Bishop are configured to have the same number of PEs with each PE possessing the same amount of register and compute resources, resulting in nearly identical area and operational power when synthesized using a commercial 28nm process design kit (PDK). Evaluation Metrics: We consider chip area, power, latency, and energy dissipation.

Table 2: Spiking transformer architectures for three static datasets and one dynamic dataset

Type	Model	Dataset	Blocks (B)	Timesteps (T)	Tokens (N)	Features (D)
Static	Model 1	CIFAR10	4	10	64	384
	Model 2	CIFAR100	4	8	64	384
	Model 3	ImageNet100	8	4	196	128
Dynamic	Model 4	DVS-Gesture	2	20	64	128
Language	Model 5	Google SC	4	8	256	384

Modeling of Architecture, Latency, Energy Dissipation, and Synthesis. We build an analytic cycle-accurate heterogeneous-core architecture simulation environment to support all unique features in spiking transformers. It traces data movement between the heterogeneous cores and hierarchical memories for assessing latency and energy dissipation. We follow the standard practice to employ a three-level memory hierarchy for memory-intensive transformer computations [18, 58]. Similar to many other analytic models, each level of memory is double-buffered to hide latency and equally partitioned to separately store different types of data.

We use CACTI 7.0 [3] to estimate energy dissipation of the weight global buffer (GLB) and spiking TT-bundle GLBs. The weight GLB is a 144KB SRAM with 512-bit read/write ports. The ping-pong spiking TT-bundle GLB0/GLB1 with each being 12KB can be equally partitioned for storage of binary spiking Q/K/V/Y. The employed DRAM has a DDR4-2400 memory bandwidth of 76.8GB/s and power consumption of 323.9mW at a core frequency of 500MHz. In Bishop, the TT-bundle sparse core consists of up to 128 parallel TT-bundle processing units. Both the TTB dense core and TTB attention core consist of 512 PE units, processing up to 32 output features and 16 TT-bundles in parallel. Each TTB unit can reconfigurablely process up to 10 spikes in one cycle. The spike generator may process up to 512 neurons in parallel. We use the open-source SIGMA simulator STONNE [36] for cycle-accurate simulation of the TTB sparse core.

The RTL implementation of the Bishop accelerator is synthesized using a commercial 28nm CMOS technology, resulting to a die area of $2.96mm^2$, a peak power dissipation of $627mW$, and a clock rate of 500 MHz. In comparison, the synthesized baseline PTB accelerator has a chip area of $2.80mm^2$ and a peak power dissipation of $606.9mW$.

6.2 Overall Latency and Energy Evaluation

Fig. 12 and Fig. 13 show the end-to-end performance of our Bishop accelerator and the two baselines. Fig. 11 shows more detailed layer-wise speedup and energy saving of Bishop in comparison with PTB [27]. Across four pre-trained models, Bishop on average achieves 299× and 5.91× speedups compared with the edge GPU and PTB [27], respectively, while reducing energy by 6.11× over PTB. Notably, these significant latency and energy reductions are achieved without comprising model accuracy.

On CIFAR10, Bishop speeds up the edge GPU and PTB by 173.9× and 4.68×; Bishop+BSA has a 238.6× and 6.37× speedup over the edge GPU and PTB; Bishop+BSA+ECP has a speedup of 249.4× and 6.71× over the edge GPU and PTB, respectively. On CIFAR100,

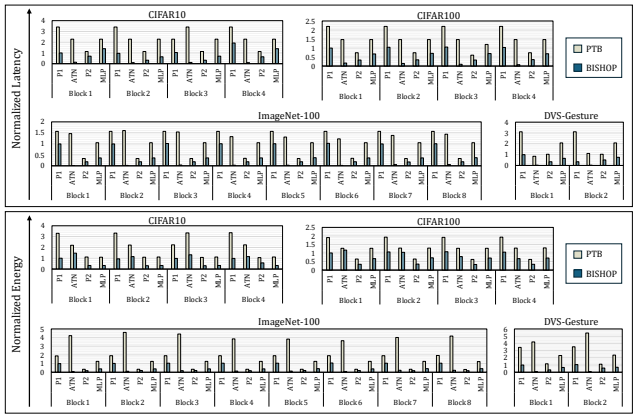

Figure 11: Normalized latency and energy comparison of Bishop with PTB [27] when accelerating the same spiking transformers trained on CIFAR10, CIFAR100, ImageNet-100 and DVS-Gesture-128. The latency and energy consumption are normalized by those of the first projection layer of the first block in Bishop, respectively. P1, ATN, and P2 indicate the $Q/K/V$ linear projection layer, spiking self-attention layer, and O linear projection layer within the spiking self-attention block, respectively; MLP indicates the spiking MLP layer.

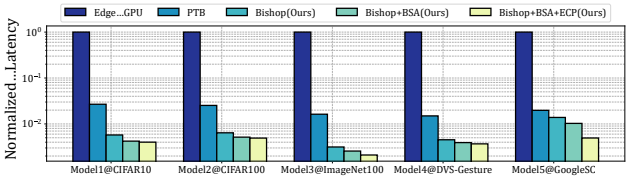

Figure 12: **Evaluation on end-to-end normalized latency reduction.**

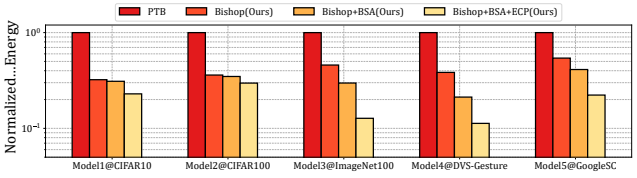

Figure 13: **Evaluation on end-to-end normalized energy consumption reduction.**

Bishop offers a 156× and 3.95× speed up over the edge GPU and PTB; Bishop+BSA provides a 193.9× and 4.90× speedup over the edge GPU and PTB; The speedups are 203.3× and 5.14×, respectively, over the edge GPU and PTB by using Bishop+BSA+ECP. On ImageNet-100, Bishop has a 317.6× and 5.17× speed up over the edge GPU and PTB; Bishop+BSA has a 389× and 6.34× speedup over the edge GPU and PTB; Bishop+BSA+ECP speeds up the edge

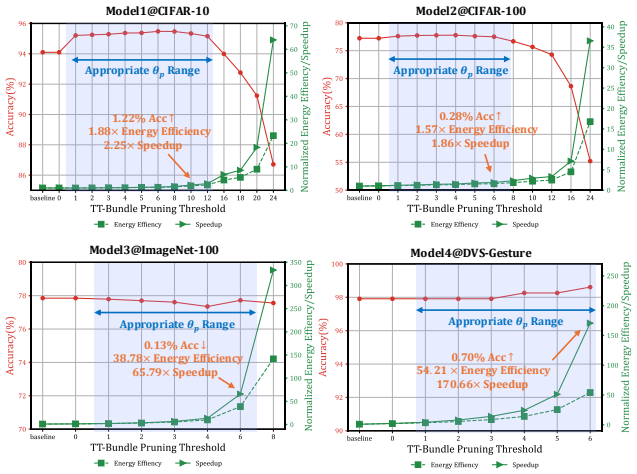

Figure 14: **Accuracy v.s. normalized energy efficiency and speedup of the spiking self-attention layers of four spiking transformers under different ECP pruning thresholds.**

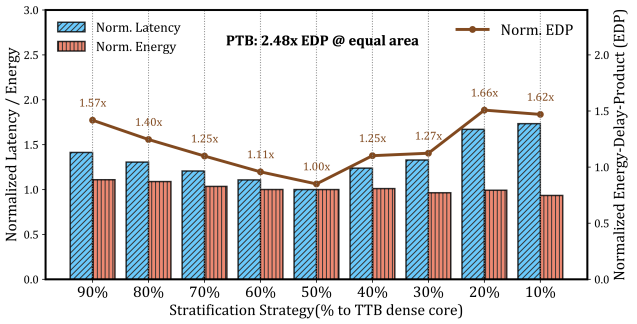

Figure 15: **Impact of stratifacation strategies on the energy, latency and EDP of Model 3 trained on ImageNet-100. Different stratification strategies yield varying stratification thresholds (θ_s) to achieve a targeted dense-to-sparse core token split ratio.**

GPU and PTB by 474.8× and 7.73×, respectively. On DVS-Gesture-128, Bishop has a 221× and 3.30× speed up over the edge GPU and PTB; Bishop+BSA increases the speedups to 254.9× and 3.81×; Bishop+BSA+ECP has 271.76× and 4.06× speedup over the edge GPU and PTB, respectively.

For model 5 evaluated on the language task Google Speech Command [50], Bishop achieves a speedup of 72.2× over the edge GPU and 1.43× over PTB. With BSA, the speedups increase to 97.33× and 1.92×, respectively. Further incorporating ECP boosts the speedups to 202.42× over the edge GPU and 4.0× over PTB.

6.3 Evaluation of Bishop Algorithms

Fig. 14 quantifies the impact of the pruning threshold of ECP on the energy efficiency and speedup of spiking self-attention layers. Across a range of pruning thresholds, a significant proportion of bundles are removed from the Q/K/V tensors without causing any substantial drop in model accuracy. In some cases, pruning even improves accuracy by introducing a denoising effect.

Nevertheless, using a large pruning threshold could degrade model performance, hence it should be properly set within an appropriate range to trade off between hardware performance and accuracy. For example, with the TTB pruning threshold θ_p set to 8 on CIFAR10, the model accuracy is improved by 1.24% while 28.2% of the Q tokens and 48.0% of the K tokens are pruned away, reducing the attention map computation by 62.7% and memory access by 38.1%. The total energy and latency are reduced to 0.52× and 0.44×, respectively. For CIFAR100, the model's accuracy is improved by 0.53% while keeping 93.2% and 55.1% of the tokens in the queries and keys, reducing energy dissipation to 0.65× and latency to 0.42×, respectively. On ImageNet-100, ECP leads to a slight accuracy drop of 0.13% while retaining only 10.7% of the query tokens and 9.7% of the key tokens. This drastic pruning reduces attention map computation overhead down to 1.04%, reducing energy to 0.03× and latency to 0.02× for the self-attention layers. On DVS-Gesture-128, the model's performance is enhanced by 0.7% while pruning the vast majority of tokens. Only 8.0% of the Q tokens and 5.49% of the K tokens remain. On average, the spiking Q tokens can be pruned away by 51.71%, reaching up to 92% in some cases. Similarly, the average pruning of the spiking K tokens can reach 67.77% with a peak reduction of 97.51%. Consequently, on average, only 15.5% of the computation is performed, leading to an average 83.76% decrease in energy consumption and 43.92% decrease in latency for the computations of the self-attention layers.

6.4 Evaluation of Bishop Hardware Accelerator

We compare Bishop and PTB [27] purely from a hardware architectural perspective by removing the proposed BSA and ECP algorithms and using the ImageNet-100 dataset.

We first demonstrate the effect of the heterogeneity of Bishop. The spiking transformer trained on the ImageNet-100 dataset has an average of 20% sparsity level across all layers. The stratifier in Bishop directs 50% of the workload to the TT-bundle dense core, and dispatches the remaining sparser workload to the TT-bundle sparse core. For the inference of a single image, on average the dense core takes 1.16ms and consumes 0.29mJ of energy. The sparse core takes 0.53ms while burning 0.038mJ of energy. In contrast, if all workload is processed by the dense core as in the case of PTB, the latency and energy increases to 1.83ms and 0.45mJ, respectively. Thus, the heterogeneity of Bishop offers a 1.39× speed-up and a 1.57× energy saving.

As for spiking attention computation, the dedicated attention core of Bishop reduces latency by 10.7-23.3× while achieving 1.39-1.96× energy saving over PTB.

6.5 Design Space Explorations

There are two important architectural hyperparameters, namely, the stratification threshold and TTB bundle volume, which have a large impact on Bishop's performance.

6.5.1 Impact of Stratification Threshold. Choosing a higher value for the stratification threshold θ_s shifts more workload from the

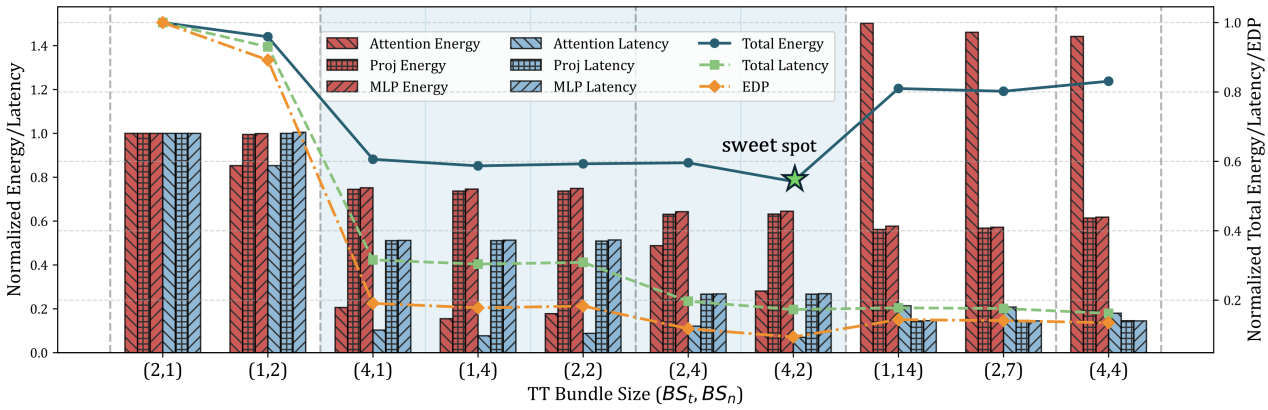

Figure 16: **Sensitivity analysis of TTB bundle volume** (BS_t, BS_n) **for Model 3 trained on ImageNet-100.**

TTB dense core to the TTB sparse core. Nevertheless, a large imbalance in workload between the two cores can degrade Bishop's performance. Using the dense core to process an excessive amount of workloads, which can constitute a large portion of the sparse workloads that cannot be efficiently processed by the dense core, may reduce the utilization of the sparse core and degrade the overall latency of Bishop. Conversely, overloading the sparse core reduces its efficiency in processing assigned dense workloads, lowers the utilization of the dense core, and degrades the overall latency. In contrast, the stratification threshold θ_s has a relatively minor effect on the energy dissipation of Bishop, since data movement continues to dominate overall energy costs. Nonetheless, shifting more workload to the dense core can result in a slight increase in energy consumption. As a result, the Energy-Delay Product (EDP) initially decreases and then increases as θ_s increases.

Fig. 15 details the impact of different workload distribution strategies on energy, latency, and EDP. In practice, a near-optimal EDP can be achieved when the stratification threshold θ_s of each layer is chosen to approximately balance the workload between the two cores. This results in an EDP improvement of 2.49× compared to the PTB architecture, when realized with an equal chip area in the

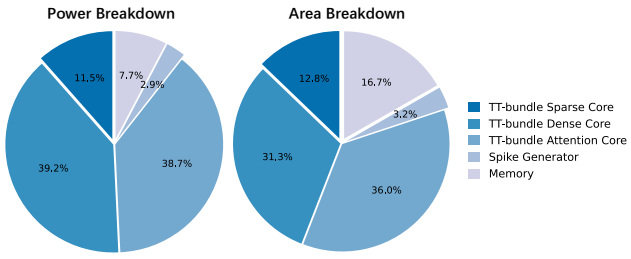

Figure 17: Power/area breakdown of the proposed Bishop accelerator.

6.5.2 Impact of TTB Bundle Volume. As shown in Fig. 4, the TT bundle volume specifies the number of spatiotemporal tokens packed in a bundle and is defined by $BS_t \times BS_n$, where BS_t and BS_n are the temporal and spatial bundle sizes, respectively. As shown in Fig. 2, the computation workload can be divided into self-attention layers, projection layers and MLP layers. Fig. 16 evaluates the impact of the TTB volume on the Bishop accelerator's energy and latency.

For the attention layers, increasing the bundle volume reduces the effectiveness of the proposed ECP pruning, leading to more bundles with only a small number of active spikes. However, these bundles still need to be transferred from the DRAM to the GLBs and eventually to the attention cores, which can degrade both energy efficiency and latency. On the other hand, when the bundle volume (BS_t, BS_n) is very small, such as $(1, 2)$ or $(2, 1)$, the intra-bundle and inter-bundle key/value data reuse drops, leading to increased data movement and higher latency.

For the projection and MLP layers, initially increasing the bundle volume boosts multi-bit weight data reuse. However, when the bundle volume goes beyond a certain value, such as 14 in Fig. 16, many idle tokens get bundled into a TTB. This leads to increased memory traffic and unnecessary processing of idle tokens, resulting in higher energy dissipation, as spiking activation memory access becomes more dominant than multi-bit weight data access. For example, when the bundle volume increases from (2,4) to (4,14), spiking activation memory access rises from 13% to 21.4% while weight memory access drops from 36.9% to 16.9%. In practice, selecting a bundle volume between 4 and 8, as indicated by the light blue region in Fig. 16, achieves near-optimal total energy and latency.

6.6 Area and Power breakdown

Fig. 17 shows the area and energy breakdown of the Bishop accelerator synthesized using a commercial 28nm technology. Nearly 90% of the total power and 80% of the chip area are consumed by the three major cores. The TTB sparse core, TTB dense core, and TTB attention core consume $72.2mW$(11.5%), $246.1mW$(39.2%) and $242.51mW$(38.7%), and occupy $0.38mm^2$(12.8%), $0.92mm^2$(31.3%), $1.06mm^2$(36.0%) chip area, respectively. In contrast, the spiking generator array consumes $0.09mm^2$(3.2%) area and $18.1mW$(2.9%), and the GLBs consume $0.495mm^2$(16.7%) area and $48.3mW$(7.7%),

respectively. The total die area and peak power of the Bishop accelerator are $2.96mm^2$ and $627mW$, respectively, similar to the PTB accelerator ($2.80mm^2$, $606.9mW$).

7 Related Work

Among the prior SNN accelerators discussed in Section 2.3, PTB [27] batches processing of multiple time steps for a given neuron within a time window, allowing multi-bit weight sharing within the window. Several other works have also explored temporally parallel processing of spiking CNN workloads. While not designed specifically for spiking transformers, LoAS [55] explores both the input activation and weight data sparsity for further efficiency improvements, particularly for SNNs with a small number of time steps. Stellar [35] accelerates SNNs based on few-spike (FS) neurons instead of LIF neurons. Importantly, all these works focus on spiking CNNs with homogeneous accelerator architectures while Bishop introduces agile spatiotemporal processing and a heterogeneous architecture tailored for complex spiking transformer workloads.

Like Bishop, sparsity has been a key focus in prior work. [14] proposes a training method to enhance spike-level sparsity, whereas Bishop maximizes structural TT-bundle-level sparsity, which is more critical for hardware acceleration. While sparse attention mechanisms have been explored in ANN-based transformer accelerators [33, 48], these methods rely on post-attention computation pruning unlike the proposed Error-Constrained TTB Pruning (ECP). In addition, they do not target binary spiking attention mechanisms.

8 Conclusions

We present Bishop, the first dedicated hardware accelerator architecture and HW/SW co-design framework for spiking transformers. Bishop operates on spiking time-token bundles (TTBs) to minimize weight data access and explore structured bundle-level sparsity. Bishop utilizes a bundle sparsity-aware training pipeline to improve structured TTB sparsity, and error-constrained pruning to aggressively trim spiking queries and keys, thereby significantly reducing the overhead of computing large spiking attentions. Bishop incorporates a dedicated TT-bundle dense/sparse core, a dense/sparse workload stratifier, and a dedicated spiking attention core to reduce data movement and boost acceleration efficiency. Extensive experimental studies have demonstrated significant advantages of Bishop over the prior SNN accelerators.

ACKNOWLEDGMENTS

This material is based upon work supported by the National Science Foundation under Grants No. 1948201 and No. 2310170.

References

[1] A. Amir, B. Taba, D. Berg, T. Melano, J. McKinstry, C. Di Nolfo, T. Nayak, A. Andreopoulos, G. Garreau, M. Mendoza *et al.*, "A low power, fully event-based gesture recognition system," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 7243–7252.

[2] M. Bal and A. Sengupta, "Spikingbert: Distilling bert to train spiking language models using implicit differentiation," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 38, no. 10, 2024, pp. 10 998–11 006.

[3] R. Balasubramonian, A. B. Kahng, N. Muralimanohar, A. Shafiee, and V. Srinivas, "Cacti 7: New tools for interconnect exploration in innovative off-chip memories," *ACM Trans. Archit. Code Optim.*, vol. 14, no. 2, jun 2017. [Online]. Available: https://doi.org/10.1145/3085572

[4] Y. Chen, Z. Yu, W. Fang, T. Huang, and Y. Tian, "Pruning of deep spiking neural networks through gradient rewiring," *arXiv preprint arXiv:2105.04916*, 2021.

[5] S. S. Chowdhury, N. Rathi, and K. Roy, "Towards ultra low latency spiking neural networks for vision and sequential tasks using temporal pruning," in *European Conference on Computer Vision*. Springer, 2022, pp. 709–726.

[6] Z. Dai, Z. Yang, Y. Yang, J. G. Carbonell, Q. V. Le, and R. Salakhutdinov, "Transformer-xl: Attentive language models beyond a fixed-length context," in *Annual Meeting of the Association for Computational Linguistics*, 2019. [Online]. Available: https://api.semanticscholar.org/CorpusID:57759363

[7] T. Dao, D. Fu, S. Ermon, A. Rudra, and C. Ré, "Flashattention: Fast and memory-efficient exact attention with io-awareness," *Advances in Neural Information Processing Systems*, vol. 35, pp. 16 344–16 359, 2022.

[8] J. Dass, S. Wu, H. Shi, C. Li, Z. Ye, Z. Wang, and Y. Lin, "ViTALiTy: Unifying low-rank and sparse approximation for vision transformer acceleration with a linear taylor attention," in *2023 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*. IEEE, pp. 415–428.

[9] M. Davies, N. Srinivasa, T.-H. Lin, G. Chinya, Y. Cao, S. H. Choday, G. Dimou, P. Joshi, N. Imam, S. Jain *et al.*, "Loihi: A neuromorphic manycore processor with on-chip learning," *Ieee Micro*, vol. 38, no. 1, pp. 82–99, 2018.

[10] D. C. De Andrade, S. Leo, M. L. D. S. Viana, and C. Bernkopf, "A neural attention model for speech command recognition," *arXiv preprint arXiv:1808.08929*, 2018.

[11] M. V. DeBole, B. Taba, A. Amir, F. Akopyan, A. Andreopoulos, W. P. Risk, J. Kusnitz, C. O. Otero, T. K. Nayak, R. Appuswamy *et al.*, "Truenorth: Accelerating from zero to 64 million neurons in 10 years," *Computer*, vol. 52, no. 5, pp. 20–29, 2019.

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

[13] L. Deng, Y. Wu, Y. Hu, L. Liang, G. Li, X. Hu, Y. Ding, P. Li, and Y. Xie, "Comprehensive snn compression using admm optimization and activity regularization," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 34, no. 6, pp. 2791–2805, 2023.

[14] ——, "Comprehensive snn compression using admm optimization and activity regularization," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 34, no. 6, pp. 2791–2805, 2023.

[15] S. Deng, Y. Li, S. Zhang, and S. Gu, "Temporal efficient training of spiking neural network via gradient re-weighting," *arXiv preprint arXiv:2202.11946*, 2022.

[16] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," *arXiv preprint arXiv:2010.11929*, 2020.

[17] H. Fan, T. Chau, S. I. Venieris, R. Lee, A. Kouris, W. Luk, N. D. Lane, and M. S. Abdelfattah, "Adaptable butterfly accelerator for attention-based NNs via hardware and algorithm co-design," in *2022 55th IEEE/ACM International Symposium on Microarchitecture (MICRO)*. IEEE, pp. 599–615. [Online]. Available: https://ieeexplore.ieee.org/document/9923888/

[18] Z. Fan, Q. Zhang, P. Abillama, S. Shoouri, C. Lee, D. T. Blaauw, H. Kim, and D. Sylvester, "Taskfusion: An efficient transfer learning architecture with dual delta sparsity for multi-task natural language processing," in *Proceedings of the 50th Annual International Symposium on Computer Architecture, ISCA 2023, Orlando, FL, USA, June 17-21, 2023*, Y. Solihin and M. A. Heinrich, Eds. ACM, 2023, pp. 5:1–5:14. [Online]. Available: https://doi.org/10.1145/3579371.3589040

[19] W. Fang, Z. Yu, Y. Chen, T. Huang, T. Masquelier, and Y. Tian, "Deep residual learning in spiking neural networks," *Advances in Neural Information Processing Systems*, vol. 34, pp. 21 056–21 069, 2021.

[20] W. Gerstner and W. M. Kistler, *Spiking neuron models: Single neurons, populations, plasticity*. Cambridge university press, 2002.

[21] P. Jansson, "Single-word speech recognition with convolutional neural networks on raw waveforms," 2018.

[22] S. Kim, S. Kim, W. Jo, S. Kim, S. Hong, and H.-J. Yoo, "20.5 c-transformer: A 2.6-18.1j/token homogeneous dnn-transformer/spiking-transformer processor with big-little network and implicit weight generation for large language models," in *2024 IEEE International Solid-State Circuits Conference (ISSCC)*, vol. 67, 2024, pp. 368–370.

[23] S. Krithivasan, S. Sen, S. Venkataramani, and A. Raghunathan, "Dynamic spike bundling for energy-efficient spiking neural networks," in *2019 IEEE/ACM International Symposium on Low Power Electronics and Design (ISLPED)*, 2019, pp. 1–6.

[24] C. Lee, S. S. Sarwar, P. Panda, G. Srinivasan, and K. Roy, "Enabling spike-based backpropagation for training deep neural network architectures," *Frontiers in neuroscience*, vol. 14, p. 497482, 2020.

[25] D. Lee, G. Lee, D. Kwon, S. Lee, Y. Kim, and J. Kim, "Flexon: A flexible digital neuron for efficient spiking neural network simulations," in *2018 ACM/IEEE 45th Annual International Symposium on Computer Architecture (ISCA)*. IEEE, 2018, pp. 275–288.

[26] J.-J. Lee and P. Li, "Reconfigurable dataflow optimization for spatiotemporal spiking neural computation on systolic array accelerators," in *2020 IEEE 38th International Conference on Computer Design (ICCD)*, 2020, pp. 57–64.

[27] J.-J. Lee, W. Zhang, and P. Li, "Parallel time batching: Systolic-array acceleration of sparse spiking neural computation," in *2022 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, 2022, pp. 317–330.

[28] Y. Li, S. Deng, X. Dong, and S. Gu, "Converting artificial neural networks to spiking neural networks via parameter calibration," *arXiv preprint arXiv:2205.10121*, 2022.

[29] Y. Li, Y. Guo, S. Zhang, S. Deng, Y. Hai, and S. Gu, "Differentiable spike: Rethinking gradient-descent for training spiking neural networks," *Advances in Neural Information Processing Systems*, vol. 34, pp. 23 426–23 439, 2021.

[30] L. Liang, Z. Qu, Z. Chen, F. Tu, Y. Wu, L. Deng, G. Li, P. Li, and Y. Xie, "H2learn: High-efficiency learning accelerator for high-accuracy spiking neural networks," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 41, no. 11, pp. 4782–4796, 2021.

[31] X. Liu, Z. Pu, P. Qu, W. Zheng, and Y. Zhang, "Activen: A scalable and flexibly-programmable event-driven neuromorphic processor," in *2024 57th IEEE/ACM International Symposium on Microarchitecture (MICRO)*. IEEE, 2024, pp. 1122–1137.

[32] Y. Liu, Y. Ma, N. Shang, T. Zhao, P. Chen, M. Wu, J. Ru, T. Jia, L. Ye, Z. Wang *et al.*, "30.2 a 22nm 0.26 nw/synapse spike-driven spiking neural network processing unit using time-step-first dataflow and sparsity-adaptive in-memory computing," in *2024 IEEE International Solid-State Circuits Conference (ISSCC)*, vol. 67. IEEE, 2024, pp. 484–486.

[33] L. Lu, Y. Jin, H. Bi, Z. Luo, P. Li, T. Wang, and Y. Liang, "Sanger: A co-design framework for enabling sparse attention using reconfigurable architecture," in *MICRO-54: 54th Annual IEEE/ACM International Symposium on Microarchitecture*, 2021, pp. 977–991.

[34] W. Maass, "Networks of spiking neurons: The third generation of neural network models," *Neural Networks*, vol. 10, no. 9, pp. 1659–1671, 1997.

[35] R. Mao, L. Tang, X. Yuan, Y. Liu, and J. Zhou, "Stellar: Energy-efficient and low-latency snn algorithm and hardware co-design with spatiotemporal computation," in *2024 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*. IEEE, 2024, pp. 172–185.

[36] F. Muñoz-Matrínez, J. L. Abellán, M. E. Acacio, and T. Krishna, "Stonne: Enabling cycle-level microarchitectural simulation for dnn inference accelerators," in *2021 IEEE International Symposium on Workload Characterization (IISWC)*, 2021.

[37] S. Narayanan, K. Taht, R. Balasubramonian, E. Giacomin, and P.-E. Gaillardon, "Spinalflow: An architecture and dataflow tailored for spiking neural networks," in *Proceedings of the ACM/IEEE 47th Annual International Symposium on Computer Architecture*, ser. ISCA '20. IEEE Press, 2020, p. 349–362. [Online]. Available: https://doi.org/10.1109/ISCA45697.2020.00038

[38] E. Qin, A. Samajdar, H. Kwon, V. Nadella, S. Srinivasan, D. Das, B. Kaul, and T. Krishna, "SIGMA: A sparse and irregular GEMM accelerator with flexible interconnects for DNN training," in *IEEE International Symposium on High Performance Computer Architecture, HPCA 2020, San Diego, CA, USA, February 22-26, 2020*. IEEE, 2020, pp. 58–70. [Online]. Available: https://doi.org/10.1109/HPCA47549.2020.00015

[39] Z. Qu, L. Liu, F. Tu, Z. Chen, Y. Ding, and Y. Xie, "DOTA: detect and omit weak attentions for scalable transformer acceleration," in *Proceedings of the 27th ACM International Conference on Architectural Support for Programming Languages and Operating Systems*, ser. ASPLOS '22. Association for Computing Machinery, pp. 14–26. [Online]. Available: https://dl.acm.org/doi/10.1145/3503222.3507738

[40] N. Rathi, P. Panda, and K. Roy, "Stdp-based pruning of connections and weight quantization in spiking neural networks for energy-efficient recognition," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 38, no. 4, pp. 668–677, 2019.

[41] N. Rathi and K. Roy, "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*, vol. 34, no. 6, pp. 3174–3182, 2021.

[42] N. Rathi, G. Srinivasan, P. Panda, and K. Roy, "Enabling deep spiking neural networks with hybrid conversion and spike timing dependent backpropagation," *arXiv preprint arXiv:2005.01807*, 2020.

[43] K. Roy, A. Jaiswal, and P. Panda, "Towards spike-based machine intelligence with neuromorphic computing," *Nature*, vol. 575, p. 607–617, 2019.

[44] ——, "Towards spike-based machine intelligence with neuromorphic computing," *Nature*, vol. 575, no. 7784, pp. 607–617, 2019.

[45] A. Sengupta, Y. Ye, R. Wang, C. Liu, and K. Roy, "Going deeper in spiking neural networks: Vgg and residual architectures," *Frontiers in neuroscience*, vol. 13, p. 95, 2019.

[46] S. Singh, A. Sarma, N. Jao, A. Pattnaik, S. Lu, K. Yang, A. Sengupta, V. Narayanan, and C. R. Das, "Nebula: A neuromorphic spin-based ultra-low power architecture for snns and anns," in *2020 ACM/IEEE 47th Annual International Symposium on Computer Architecture (ISCA)*. IEEE, 2020, pp. 363–376.

[47] S. Singh, A. Sarma, S. Lu, A. Sengupta, M. T. Kandemir, E. Neftci, V. Narayanan, and C. R. Das, "Skipper: Enabling efficient snn training through activation-checkpointing and time-skipping," in *2022 55th IEEE/ACM International Symposium on Microarchitecture (MICRO)*. IEEE, 2022, pp. 565–581.

[48] H. Wang, Z. Zhang, and S. Han, "Spatten: Efficient sparse attention architecture with cascade token and head pruning," in *2021 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*. IEEE, 2021, pp. 97–110.

[49] Z. Wang, Y. Fang, J. Cao, Q. Zhang, Z. Wang, and R. Xu, "Masked spiking transformer," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023, pp. 1761–1771.

[50] P. Warden, "Speech commands: A dataset for limited-vocabulary speech recognition," *arXiv preprint arXiv:1804.03209*, 2018.

[51] B. Xu, J. Hwang, P. Vanna-iampikul, S. K. Lim, and P. Li, "Spiking transformer hardware accelerators in 3d integration," in *Proceedings of the 43rd IEEE/ACM International Conference on Computer-Aided Design*, 2024, pp. 1–9.

[52] Q. Yang, Q. Liu, and H. Li, "Deep residual spiking neural network for keyword spotting in low-resource settings." in *Interspeech*, 2022, pp. 3023–3027.

[53] M. Yao, J. Hu, T. Hu, Y. Xu, Z. Zhou, Y. Tian, B. Xu, and G. Li, "Spike-driven transformer v2: Meta spiking neural network architecture inspiring the design of next-generation neuromorphic chips," *arXiv preprint arXiv:2404.03663*, 2024.

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

[55] R. Yin, Y. Kim, D. Wu, and P. Panda, "Loas: Fully temporal-parallel dataflow for dual-sparse spiking neural networks," in *2024 57th IEEE/ACM International Symposium on Microarchitecture (MICRO)*. IEEE, 2024, pp. 1107–1121.

[56] R. Yin, Y. Li, A. Moitra, and P. Panda, "Mint: Multiplier-less integer quantization for energy efficient spiking neural networks," in *2024 29th Asia and South Pacific Design Automation Conference (ASP-DAC)*. IEEE, 2024, pp. 830–835.

[57] R. Yin, A. Moitra, A. Bhattacharjee, Y. Kim, and P. Panda, "Sata: Sparsity-aware training accelerator for spiking neural networks," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 42, no. 6, pp. 1926–1938, 2023.

[58] H. You, Z. Sun, H. Shi, Z. Yu, Y. Zhao, Y. Zhang, C. Li, B. Li, and Y. Lin, "Vitcod: Vision transformer acceleration via dedicated algorithm and accelerator co-design," in *IEEE International Symposium on High-Performance Computer Architecture, HPCA 2023, Montreal, QC, Canada, February 25 - March 1, 2023*. IEEE, 2023, pp. 273–286. [Online]. Available: https://doi.org/10.1109/HPCA56546.2023.10071027

[59] H. You, Y. Xiong, X. Dai, B. Wu, P. Zhang, H. Fan, P. Vajda, and Y. C. Lin, "Castling-ViT: Compressing self-attention via switching towards linear-angular attention at vision transformer inference," in *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, pp. 14 431–14 442. [Online]. Available: https://ieeexplore.ieee.org/document/10203309/

[60] J. Zhang, B. Dong, H. Zhang, J. Ding, F. Heide, B. Yin, and X. Yang, "Spiking transformers for event-based single object tracking," in *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022, pp. 8791–8800.

[61] W. Zhang and P. Li, "Temporal spike sequence learning via backpropagation for deep spiking neural networks," in *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, virtual*, 2020.

[62] Y. Zhao, D. Wu, and J. Wang, "Alisa: Accelerating large language model inference via sparsity-aware kv caching," *arXiv preprint arXiv:2403.17312*, 2024.

[63] H. Zheng, Y. Wu, L. Deng, Y. Hu, and G. Li, "Going deeper with directly-trained larger spiking neural networks," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 35, no. 12, 2021, pp. 11 062–11 070.

[64] Z. Zhou, Y. Zhu, C. He, Y. Wang, S. YAN, Y. Tian, and L. Yuan, "Spikformer: When spiking neural network meets transformer," in *The Eleventh International Conference on Learning Representations*, 2023.

[65] R.-J. Zhu, Q. Zhao, G. Li, and J. K. Eshraghian, "Spikegpt: Generative pre-trained language model with spiking neural networks," 2023.