

Efficient Training of Spiking Neural Networks by Spike-aware Data Pruning

Chenxiang Ma, Xinyi Chen, Yujie Wu, Kay Chen Tan, *Fellow, IEEE*, Jibin Wu, *Member, IEEE*

Abstract—Spiking neural networks (SNNs), recognized as an energy-efficient alternative to traditional artificial neural networks (ANNs), have advanced rapidly through the scaling of models and datasets. However, such scaling incurs considerable training overhead, posing challenges for researchers with limited computational resources and hindering the sustained development of SNNs. Data pruning is a promising strategy for accelerating training by retaining the most informative examples and discarding redundant ones, but it remains largely unexplored in SNNs. Directly applying ANN-based data pruning methods to SNNs fails to capture the intrinsic importance of examples and suffers from high gradient variance. To address these challenges, we propose a novel spike-aware data pruning (SADP) method. SADP reduces gradient variance by determining each example’s selection probability to be proportional to its gradient norm, while avoiding the high cost of direct gradient computation through an efficient upper bound, termed spike-aware importance score. This score accounts for the influence of all-or-nothing spikes on the gradient norm and can be computed with negligible overhead. Extensive experiments across diverse datasets and architectures demonstrate that SADP consistently outperforms data pruning baselines and achieves training speedups close to the theoretical maxima at different pruning ratios. Notably, SADP reduces training time by 35% on ImageNet while maintaining accuracy comparable to that of full-data training. This work, therefore, establishes a data-centric paradigm for efficient SNN training and paves the way for scaling SNNs to larger models and datasets. The source code will be released publicly after the review process.

Index Terms—Spiking neural networks, neuromorphic computing, data pruning, efficient training.

I. INTRODUCTION

SPIKING neural networks (SNNs) have gained considerable attention as a promising energy-efficient alternative to traditional artificial neural networks (ANNs) [1]–[6]. Unlike ANNs, which rely on dense, continuous-valued activations that require frequent and computationally intensive updates across the entire network, SNNs encode and transmit information through sparse, binary spikes [7]. This enables only a small subset of neurons to participate in processing afferent information while the majority remain quiescent at any given moment [2]. In addition,

SNNs exhibit abundant spatiotemporal dynamics by temporally integrating incoming spikes into their membrane potentials and generating spikes only upon reaching threshold conditions [3]. The spike-based representation and temporal integration enable inherently sparse, asynchronous, and event-driven computation. When deployed on neuromorphic hardware [8]–[11], these characteristics translate into exceptional energy efficiency, thereby rendering SNNs highly advantageous in resource-constrained applications [12]–[16].

The rapid advancement of SNNs in recent years has been propelled by breakthroughs in scalable training methods [17]–[19] and the expansion of large-scale architectures [20]–[23] and datasets [24]–[27]. In particular, the non-differentiable nature of the spike firing function posed a major obstacle to the development of scalable training methods [28]–[31]. This obstacle was largely overcome by surrogate gradients [32], which enabled the application of gradient-based optimization, such as the back-propagation through time (BPTT) algorithm [17], [33], to SNNs. Building upon this breakthrough, extensive efforts were devoted to enhancing the trainability of large-scale SNNs [34]. In particular, advanced batch normalization methods [35]–[37] were introduced to stabilize the optimization of deep architectures, adaptive surrogate gradients [38], [39] were designed to facilitate gradient flow, and neuron models [40]–[43] incorporating more complex neuronal dynamics were developed to strengthen temporal processing capabilities [44]. Collectively, these advancements have enabled the expansion of SNNs to large-scale architectures with hundreds of layers and millions of parameters, exemplified by spiking variants of ResNets [20], [45] and Transformers [21]–[23]. Alongside this architectural growth, training datasets have also expanded considerably. Performance evaluation has shifted to large-scale datasets like ImageNet [46], which contains over one million training examples. Similarly, event-based datasets, collected by dynamic vision sensors (DVS) [47], have grown from the one thousand examples in DvsGesture [24] to more than one hundred thousand in the recent HAR-DVS dataset [26].

However, the increasing scale of both architectures and datasets leads to substantially longer training time, posing significant challenges in terms of extended development cycles and heavy computational demand. These high training costs are often unsustainable for researchers with limited high-performance computing resources. Consequently, enhancing the training efficiency of SNNs has become urgent for their scalable deployment and sustained advancement.

Preliminary attempts to improve the training efficiency of SNNs have primarily concentrated on learning algorithms [48]–[50] and neuron models [51], [52]. From the algorithmic

Corresponding author: Jibin Wu (e-mail: jibin.wu@polyu.edu.hk).

Chenxiang Ma and Xinyi Chen are with the Department of Data Science and Artificial Intelligence, The Hong Kong Polytechnic University, Hong Kong, SAR.

Yujie Wu is with the Department of Computing, The Hong Kong Polytechnic University, Hong Kong, SAR.

Kay Chen Tan is with the Department of Data Science and Artificial Intelligence and the Research Center of Data Science and Artificial Intelligence, The Hong Kong Polytechnic University, Hong Kong, SAR.

Jibin Wu is with the Department of Data Science and Artificial Intelligence, the Department of Computing, and the Research Center of Data Science and Artificial Intelligence, The Hong Kong Polytechnic University, Hong Kong, SAR.

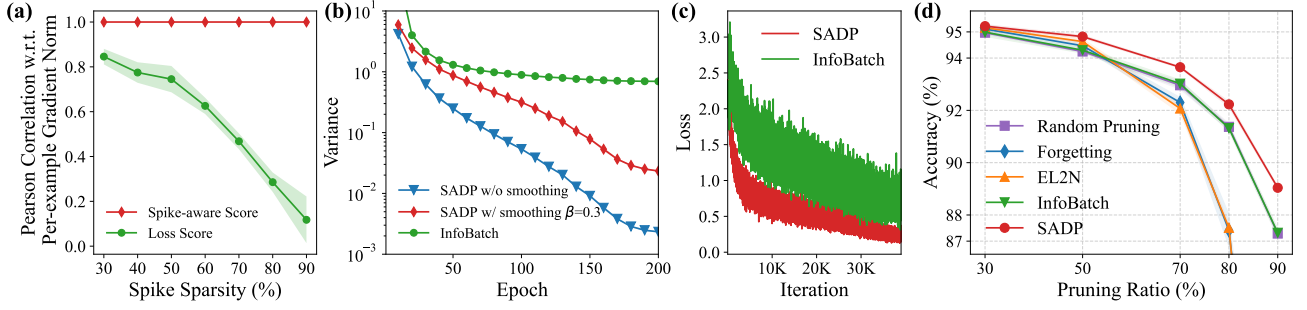


Fig. 1. Comparison of SADP with existing data pruning methods. (a) Pearson correlation between the per-example gradient norm and different importance scores. Our spike-aware score maintains a significantly higher correlation than the loss score, particularly under high spike sparsity. (b) SADP reduces gradient variance throughout training. (c) This variance reduction leads to faster convergence. (d) SADP consistently achieves higher accuracy across different pruning ratios, with accuracy gains becoming more pronounced at higher ratios. Experiments are conducted with ResNet18 on CIFAR-10.

perspective, online learning rules [48]–[50] were developed to mitigate the high memory cost of BPTT. By decoupling the temporal dependencies of gradients, these approaches allow memory consumption to be independent of the total number of time steps. In addition, parallel spiking neuron models [51], [52] were proposed to support simultaneous gradient computation along the temporal dimension, which accelerates training, particularly on long-sequence tasks. In contrast to these prior efforts, this work explores a fundamentally orthogonal perspective by targeting the dataset as the source of training efficiency gains. Given that large-scale datasets typically contain many redundant or uninformative examples, identifying a small and informative subset from the full training dataset, known as data pruning, provides a promising strategy for reducing training time while maintaining competitive performance.

In this article, we systematically investigate data pruning for SNNs. We start by directly applying existing data pruning methods designed for ANNs [53]–[55]. Our analysis reveals two critical reasons why these methods fall short when applied to SNNs. First, existing approaches struggle to efficiently and accurately evaluate data importance in the SNN training. Data pruning methods assign an importance score to each training example and then retain the most informative subset. Approaches such as Forgetting [53] and EL2N [54] compute these scores by tracking training dynamics on the full dataset over tens of epochs, which is computationally expensive. InfoBatch [55] offers a more efficient alternative by using the loss value as the importance score. However, this strategy is relatively limited for SNNs, where the gradient norm, a faithful measure of an example’s contribution to training, relies on sparse spike activity. Due to the all-or-nothing nature of spikes, a spike with a value of zero eliminates the corresponding gradient contribution, regardless of the loss value. As a result, the loss value correlates poorly with the gradient norm. This weak correlation becomes increasingly pronounced as spike sparsity increases (Figure 1(a)), limiting the effectiveness of loss-based importance scores in SNNs.

Second, existing data pruning methods suffer from high gradient variance, hindering the convergence speed of SNN training. Forgetting [53] and EL2N [54] select examples deterministically, leading to both biased and high-variance gradient estimates relative to using the full dataset. InfoBatch [55]

mitigates the bias issue by employing probabilistic sampling and reweighting the gradients of selected examples by the inverse of their probabilities. However, it incurs high gradient variance, which slows convergence and ultimately limits its performance (Figures 1(b) and (c)).

To address these challenges, we propose spike-aware data pruning (SADP) (Figure 2), a novel data pruning method tailored for SNNs. We formalize data pruning as a variance minimization problem and show that, under this objective, the optimal selection probability of a training example is approximately proportional to its gradient norm. Computing exact per-example norms, however, is prohibitively expensive. Therefore, we propose a spike-aware importance score that is an upper bound on the gradient norm, capturing the effect of all-or-nothing spikes while incurring negligible overhead. This score enables a tractable relaxation of the variance minimization objective, thereby closely approximating its optimum. To prevent training instability from low selection probabilities, we further propose a smoothing mechanism that enforces a minimum probability, stabilizing training while preserving low variance. Finally, to facilitate more efficient data usage over the course of training, we propose a dynamic pruning schedule that increases the pruning ratio over epochs while keeping the average ratio constant.

We conduct extensive experiments on both static and event-based vision datasets, including CIFAR-10 [56], CIFAR-100 [56], ImageNet [46], CIFAR10-DVS [25], and HAR-DVS [26], across diverse architectures such as spiking VGG [57], ResNet [20], [58], and Transformer [59]. Experimental results show that SADP consistently outperforms existing data pruning methods across different pruning ratios, with its advantage becoming more pronounced as the pruning ratio increases. Owing to its negligible computational overhead, SADP achieves the theoretical maximum reduction in training time, approximately equal to the given pruning ratio. For instance, SADP reduces training time by 70% on CIFAR10-DVS [25] and 35% on ImageNet [46] without compromising accuracy. Furthermore, SADP demonstrates broad compatibility with various spiking models (such as PSN [51] and T-RevSNN [60]), learning algorithms (including online [48], [49] and local learning [61], [62]), and efficient inference techniques (such as quantization-aware training [63] and network pruning [64]).

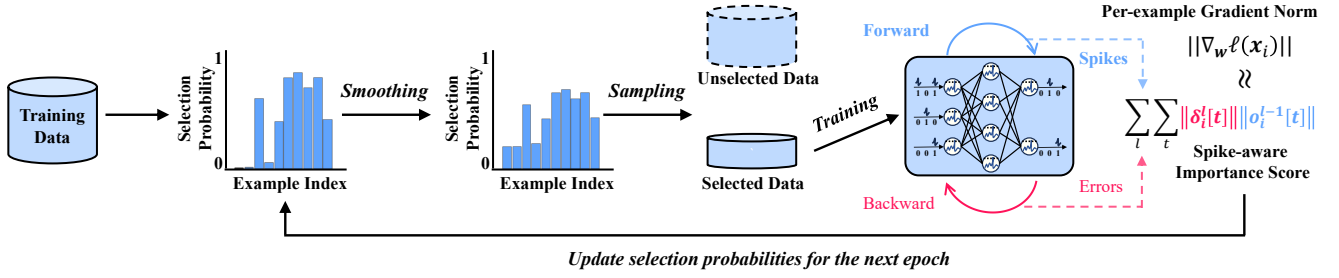


Fig. 2. Illustration of SADP. At the start of each training epoch, selection probabilities are computed for all examples using the proposed spike-aware importance score, which provides an efficient and accurate approximation of the per-example gradient norm for variance minimization. To enhance training stability, the probabilities are smoothed to avoid extremely small values, after which a subset of examples is probabilistically sampled for training the SNN. Note that the spike-aware importance score explicitly captures the effect of sparse binary spikes on the per-example gradient norm, and its computation requires only quantities already available from forward and backward passes, thereby achieving both high efficiency and strong effectiveness.

Our key contributions are summarized as follows:

- We are the first to systematically investigate data pruning for SNNs and identify two challenges that limit the direct adoption of ANN-based methods, namely, ineffective data importance estimation and high gradient variance.
- We propose SADP, the first data pruning framework designed for SNNs. It incorporates a variance minimization formulation, a spike-aware importance score, a probability smoothing mechanism, and a dynamic pruning schedule to jointly overcome the identified challenges, achieving superior accuracy and training efficiency.
- Extensive experiments demonstrate that SADP outperforms prior data pruning methods, reduces training time by over 30% with lossless accuracy, and generalizes well across diverse SNN methods.

The remainder of this paper is structured as follows. Section II provides essential preliminaries on SNNs and data pruning. Section III analyzes the challenges of data pruning in SNNs. Section IV details the proposed SADP method, and Section V presents extensive experimental evaluations and analyses. Finally, Section VI concludes the paper.

II. PRELIMINARIES

A. Spiking Neural Networks

SNNs are commonly built with Leaky Integrate-and-Fire (LIF) neuron model [1]. At time step t , LIF neurons in layer l integrate incoming spikes $\mathbf{o}^{l-1}[t]$ into their membrane potential $\mathbf{u}^l[t]$, which decays over time by a factor of λ . When $\mathbf{u}^l[t]$ surpasses a threshold ϑ , the neurons generate binary (0 or 1) spikes $\mathbf{o}^l[t]$, followed by a reset process. This neuronal dynamics can be formulated as:

$$\mathbf{u}^l[t] = \lambda \mathbf{u}^l[t-1] + \mathbf{W}^l \mathbf{o}^{l-1}[t], \quad (1)$$

$$\mathbf{o}_j^l[t] = \begin{cases} 1, & u_j^l[t] \geq \vartheta, \\ 0, & \text{otherwise,} \end{cases} \quad (2)$$

$$\mathbf{u}^l[t] = \mathbf{u}^l[t] - \vartheta \mathbf{o}^l[t]. \quad (3)$$

Large-scale SNNs are typically trained with BPTT [17], which computes the gradient of a loss $\mathcal{L}$ with respect to the l -th layer

weight $\mathbf{W}^l$ as:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{W}^l} = \sum_{t=1}^T \frac{\partial \mathcal{L}}{\partial \mathbf{u}^l[t]} \frac{\partial \mathbf{u}^l[t]}{\partial \mathbf{W}^l} = \sum_{t=1}^T \delta^l[t] \mathbf{o}^{l-1}[t]^\top, \quad (4)$$

$$\delta^l[t] = \begin{cases} \frac{\partial \mathcal{L}}{\partial \mathbf{u}^L[T]}, & l=L \text{ and } t=T, \\ \delta^L[t+1] \frac{\partial \mathbf{u}^L[t+1]}{\partial \mathbf{u}^L[t]} + \frac{\partial \mathcal{L}}{\partial \mathbf{u}^L[t]}, & l=L \text{ and } t < T, \\ \delta^{l+1}[T] \frac{\partial \mathbf{u}^{l+1}[T]}{\partial \mathbf{o}^l[T]} \frac{\partial \mathbf{o}^l[T]}{\partial \mathbf{u}^l[t]}, & l < L \text{ and } t=T, \\ \delta^l[t+1] \frac{\partial \mathbf{u}^l[t+1]}{\partial \mathbf{u}^l[t]} + \delta^{l+1}[t] \frac{\partial \mathbf{u}^{l+1}[t]}{\partial \mathbf{u}^l[t]}, & \text{otherwise,} \end{cases} \quad (5)$$

where $\delta^l[t] := \frac{\partial \mathcal{L}}{\partial \mathbf{u}^l[t]}$, represents the error back-propagated from the last layer and the last time step. T denotes the total number of time steps. The gradient $\frac{\partial \mathbf{o}_j^l[t]}{\partial \mathbf{u}_j^l[t]}$ is zero except at the point where $u_j^l[t] = \vartheta$, where it becomes infinite. To address this issue, a continuous surrogate gradient function [32] is often adopted to replace the gradient during the backward pass.

B. Data Pruning

Data Pruning seeks to select a small subset $\mathcal{S}$ of size S from the full training dataset $\mathcal{D}$ of size N , such that a model trained on $\mathcal{S}$ can achieve test performance comparable to that obtained when trained on $\mathcal{D}$. The subset size S is determined by a given pruning ratio r , defined as $r := 1 - \frac{S}{N}$. If the subset is efficiently identified, data pruning can reduce the training time by a factor of r . Existing data pruning methods assign an importance score to each training example and select those with higher scores to build the subset. For instance, Forgetting [53] and EL2N [54] track training dynamics to estimate the forgetting and error L2-norm scores, respectively. Examples with scores above a threshold are selected before standard training begins and remain fixed throughout the training.

In contrast, the recently proposed InfoBatch [55] assigns each example's loss as its importance score and adaptively selects a subset at the start of every epoch. To ensure unbiased gradient estimates between training on the selected subset and the full dataset, InfoBatch employs probabilistic sampling. Specifically, each example i is selected with a predefined probability $p_i = z$ if its score is less than the mean of all examples' scores, and

with probability $p_i = 1$ otherwise. Note that the probability value z is a hyperparameter that requires manual tuning. The gradients of the selected examples are subsequently upsampled by a factor of $\frac{1}{p_i}$ to correct for the sampling bias.

III. CHALLENGES OF DATA PRUNING IN SNNs

This section details two critical challenges that limit the effectiveness of existing data pruning methods in SNNs: ineffective data importance estimation and high gradient variance.

A. Ineffective Data Importance Estimation

Accurate estimation of data importance is crucial for data pruning, as it determines which examples are selected. However, existing methods fail to efficiently and effectively capture true data importance when applied to SNNs. Forgetting [53] and EL2N [54] estimate importance scores by iteratively tracking forgetting events and L2-norm of errors, respectively, which incur significant computational overhead. InfoBatch [55] mitigates this cost by using the loss value as the importance score. However, this approach is inadequate for SNNs due to the weak correlation between the loss and the gradient norm, as illustrated in Figure 1(a) and further analyzed below.

We analyze the weight gradient norm of an example when training an SNN on a dataset $\mathcal{D} = \{\mathbf{x}_i\}_{i=1}^N$, since it directly reflects the example's contribution to the training process. For clarity, we formulate the square of the gradient norm computed via the BPTT algorithm [17], given by:

$$\|\nabla_{\mathbf{W}} \ell(\mathbf{x}_i)\|^2 = \sum_{l \in \mathbb{L}} \|\nabla_{\mathbf{W}^l} \ell(\mathbf{x}_i)\|^2 \quad (6)$$

$$= \sum_{l \in \mathbb{L}} \left\| \sum_{t=1}^T \frac{\partial \ell(\mathbf{x}_i)}{\partial \mathbf{u}^l[t]} \frac{\partial \mathbf{u}^l[t]}{\partial \mathbf{W}^l} \right\|^2 \quad (7)$$

$$= \sum_{l \in \mathbb{L}} \left\| \sum_{t=1}^T \delta_i^l[t] \mathbf{o}_i^{l-1}[t]^\top \right\|^2, \quad (8)$$

where $\mathbb{L}$ denotes the set of network layers and $\|\cdot\|$ represents the Euclidean norm of a vector or the Frobenius norm of a matrix. $\nabla_{\mathbf{W}^l} \ell(\mathbf{x}_i)$ denotes the gradient of the example i 's loss $\ell(\mathbf{x}_i)$ with respect to the weights of the l -th layer $\mathbf{W}^l$. $\delta_i^l[t] := \frac{\partial \ell(\mathbf{x}_i)}{\partial \mathbf{u}^l[t]}$ is the backpropagated error corresponding to example i . It captures both spatial and temporal dependencies and is computed recursively as in Eq. (5).

As shown in Eq. (8), the gradient norm of example i is governed by the outer products between its errors $\{\delta_i^l[t]\}_{l,t}$ and spikes $\{\mathbf{o}_i^{l-1}[t]\}_{l,t}$. Due to the binary and sparse nature of spikes, any zero-valued $\mathbf{o}_i^{l-1}[t]$ eliminates the corresponding gradient contributions, regardless of the magnitude of the loss or errors. The all-or-nothing characteristic of spikes thus results in a weak correlation between the loss value [55] and the weight gradient norm in SNNs, which becomes more pronounced as spike sparsity increases (Figure 1(a)). Therefore, loss-based data pruning methods, such as InfoBatch [55], fail to identify truly important examples, leading to suboptimal performance in SNNs (Figure 1 (d)).

Notably, computing the per-example gradient norm in Eq. (8) is computationally expensive in popular SNN training

frameworks such as PyTorch [65], SpikingJelly [66], and snnTorch [3]. This is because these frameworks automatically average gradients over the batch dimension, rendering per-example gradients inaccessible, whereas computing gradients one example at a time is prohibitively inefficient.

As a result, efficiently estimating data importance in a way that closely approximates the per-example gradient norm remains a critical challenge for data pruning in SNNs.

B. High Gradient Variance and Slow Convergence Speed

While InfoBatch [55] ensures unbiased gradient estimates between the selected subset and the full dataset, it suffers from high gradient variance, which slows SNN training convergence. This issue is visualized in Figures 1(b) and (c) and further analyzed in the following.

Here, we consider the general weight update rule for training an SNN at iteration $m+1$, given by:

$$\mathbf{W}_{m+1} = \mathbf{W}_m - \eta \nabla_{\mathbf{W}} \mathcal{L}, \quad (9)$$

where η denotes the learning rate, and $\nabla_{\mathbf{W}} \mathcal{L}$ is the gradient of the total loss $\mathcal{L}$ with respect to the weights $\mathbf{W}$. The total loss $\mathcal{L}$ is defined as the average loss over all examples, i.e. $\mathcal{L} := \frac{1}{N} \sum_{i=1}^N \ell(\mathbf{x}_i)$.

Then, we formalize the convergence speed $\mathcal{C}$ as the expected reduction in distance to the optimal weights $\mathbf{W}^*$ between two consecutive iterations. Specifically, using the weight update rule in Eq. (9), the convergence speed is defined as:

$$\begin{aligned} \mathcal{C} &:= -\mathbb{E} \left[\|\mathbf{W}_{m+1} - \mathbf{W}^*\|^2 - \|\mathbf{W}_m - \mathbf{W}^*\|^2 \right] \\ &= -\mathbb{E} \left[(\mathbf{W}_m - \eta \nabla_{\mathbf{W}} \mathcal{L})^\top (\mathbf{W}_m - \eta \nabla_{\mathbf{W}} \mathcal{L}) \right. \\ &\quad \left. + 2\eta \nabla_{\mathbf{W}} \mathcal{L}^\top \mathbf{W}^* - \mathbf{W}_m^\top \mathbf{W}_m \right] \\ &= -\mathbb{E} \left[-2\eta (\mathbf{W}_m - \mathbf{W}^*)^\top \nabla_{\mathbf{W}} \mathcal{L} + \eta^2 \nabla_{\mathbf{W}} \mathcal{L}^\top \nabla_{\mathbf{W}} \mathcal{L} \right] \\ &= 2\eta (\mathbf{W}_m - \mathbf{W}^*)^\top \mathbb{E}[\nabla_{\mathbf{W}} \mathcal{L}] - \eta^2 \mathbb{E}[\nabla_{\mathbf{W}} \mathcal{L}]^\top \mathbb{E}[\nabla_{\mathbf{W}} \mathcal{L}] \\ &\quad - \eta^2 \text{Var}[\nabla_{\mathbf{W}} \mathcal{L}]. \end{aligned} \quad (10)$$

Eq. (10) reveals that the variance term $\text{Var}[\nabla_{\mathbf{W}} \mathcal{L}]$ introduces noise into the weight update and reduces convergence speed.

This insight underscores an important objective for data pruning: minimizing gradient variance to accelerate convergence and improve training efficiency.

IV. SPIKE-AWARE DATA PRUNING (SADP)

In this section, we propose SADP, which jointly addresses the aforementioned two challenges of accurately and efficiently estimating data importance and minimizing gradient variance. SADP is grounded in a variance minimization framework, where each example's optimal selection probability is proportional to its gradient norm (Section IV-A). To approximate this quantity with negligible computational overhead, we propose an efficient upper bound on the gradient norm, referred to as spike-aware importance score (Section IV-B). To enhance stability during training, SADP further incorporates a probability smoothing mechanism that prevents selection probabilities from collapsing to very small values (Section IV-C). Finally, we

Algorithm 1 Spike-Aware Data Pruning (SADP)

```

1: Input: SNN model  $f_{\mathbf{W}}$ , dataset  $\mathcal{D} = \{\mathbf{x}_i\}_{i=1}^N$ , training
   epochs  $K$ , pruning ratio  $r$ , maximum ratio  $r_{\max}$ , smoothing
   constant  $\beta$ , layer index  $l$ , time window  $T$ , subset size  $S$ ,
   batch size  $B$ .
2: Initialize training examples' importance scores  $\{G_i\}_{i=1}^N$ .
3: for epoch  $k = 1$  to  $K$  do
4:   Compute pruning ratio  $r_k$  using Eq. (42).
5:   Compute examples' selection probabilities  $\mathbf{p} =$ 
       $(p_1, \dots, p_N)$  using Eqs. (40) and (41).
6:   Build subset  $\mathcal{S} = \{\mathbf{x}_i \in \mathcal{D} \mid m_i = 1, m_i \sim$ 
      Bernoulli( $p_i$ ) $\}$ .
7:   for iteration  $m = 1$  to  $\frac{S}{B}$  do
8:     Sample batch  $\{\mathbf{x}_i\}_{i=1}^B \subseteq \mathcal{S}$ .
9:     Perform forward pass and record  $\{\mathbf{o}_i^{l-1}[t]\}_{i,l,t}$ .
10:    Compute the scaled loss based on Eq. (11).
11:    Perform backward pass and record  $\{\delta_i^l[t]\}_{i,l,t}$ .
12:    Compute and update  $\{G_i\}_{i=1}^B$  using Eq. (29).
13:    Update model parameters of  $f_{\mathbf{W}}$ .
14:   end for
15: end for

```

propose a dynamic pruning ratio schedule that promotes more efficient and balanced utilization of data over the course of training (Section IV-D). An overview of SADP is presented in Figure 2, with pseudocode provided in Algorithm 1.

A. Low-variance Data Pruning

We begin by formulating the estimated gradient computed from a pruned subset of size S . At the start of each epoch, every example i is independently selected according to a Bernoulli distribution with a selection probability p_i . Let $m_i \in \{0, 1\}$ denote a binary random variable indicating whether example i is selected. The estimated gradient over the dataset $\mathcal{D} = \{\mathbf{x}_i\}_{i=1}^N$ can then be given as:

$$\hat{\nabla}_{\mathbf{W}} \mathcal{L} = \frac{1}{S} \cdot \frac{S}{N} \sum_{i=1}^N \frac{\nabla_{\mathbf{W}} \ell(\mathbf{x}_i) \cdot m_i}{p_i}, \quad (11)$$

where each selected example's gradient is reweighted by the inverse of its selection probability p_i , ensuring that the estimated gradient remains unbiased with respect to the full-data gradient [55].

Next, we elucidate how to reduce the variance of the estimated gradient $\text{Var}[\hat{\nabla}_{\mathbf{W}} \mathcal{L}]$, which can be formulated as the following optimization problem:

$$\begin{aligned} \min_{\mathbf{p}} \quad & \text{Var}[\hat{\nabla}_{\mathbf{W}} \mathcal{L}] = \frac{1}{N^2} \sum_{i=1}^N \frac{(1-p_i) \|\nabla_{\mathbf{W}} \ell(\mathbf{x}_i)\|^2}{p_i} \\ \text{s.t.} \quad & \sum_{i=1}^N p_i = S \quad \text{and} \quad 0 \leq p_i \leq 1, \end{aligned} \quad (12)$$

where the gradient variance is governed by the selection probabilities $\mathbf{p} = (p_1, \dots, p_N)$. The constraint $\sum_{i=1}^N p_i = S$ ensures that the expected number of selected examples equals the given subset size S . Proposition 1 derives the optimal form of $\mathbf{p}$ that minimizes the gradient variance:

Proposition 1. For the Bernoulli sampling gradient estimator in Eq. (11), the optimal selection probabilities $\mathbf{p}^* = (p_1^*, \dots, p_N^*)$ that minimize the gradient variance $\text{Var}[\hat{\nabla}_{\mathbf{W}} \mathcal{L}]$ are

$$p_i^* = \frac{\min(\|\nabla_{\mathbf{W}} \ell(\mathbf{x}_i)\|, \alpha) \cdot S}{\sum_{j=1}^N \min(\|\nabla_{\mathbf{W}} \ell(\mathbf{x}_j)\|, \alpha)}, \quad i = 1, \dots, N, \quad (13)$$

where the clipping threshold α is defined as

$$\alpha = \frac{\sum_{i=1}^{N-M} \|\nabla_{\mathbf{W}} \ell(\mathbf{x})\|_{(i)}}{S - M}, \quad (14)$$

and $\|\nabla_{\mathbf{W}} \ell(\mathbf{x})\|_{(1)} \leq \dots \leq \|\nabla_{\mathbf{W}} \ell(\mathbf{x})\|_{(N)}$ denote the sorted gradient norms. The integer M satisfies:

$$\frac{\|\nabla_{\mathbf{W}} \ell(\mathbf{x})\|_{(N-M)}}{\sum_{i=1}^{N-M} \|\nabla_{\mathbf{W}} \ell(\mathbf{x})\|_{(i)}} < \frac{1}{S - M} \quad \text{and} \quad (15)$$

$$\frac{\|\nabla_{\mathbf{W}} \ell(\mathbf{x})\|_{(N-M+1)}}{\sum_{i=1}^{N-M+1} \|\nabla_{\mathbf{W}} \ell(\mathbf{x})\|_{(i)}} \geq \frac{1}{S - M + 1}. \quad (16)$$

Proof. Let $g_i := \|\nabla_{\mathbf{W}} \ell(\mathbf{x}_i)\|$, and the ordered set $\{g_i\}_{i=1}^N$ is represented by $g_{(1)} \leq g_{(2)} \leq \dots \leq g_{(N)}$. The Lagrangian for this constrained optimization problem is

$$E(\mathbf{p}, \lambda, \boldsymbol{\mu}, \boldsymbol{\nu}) = \sum_{i=1}^N \frac{(1-p_i)g_{(i)}^2}{p_i} \quad (17)$$

$$+ \lambda \left(\sum_{i=1}^N p_i - S \right) + \sum_{i=1}^N \mu_i (p_i - 1 + \nu_i^2), \quad (18)$$

where $\lambda, \boldsymbol{\mu} = (\mu_1, \dots, \mu_N)$, and $\boldsymbol{\nu} = (\nu_1, \dots, \nu_N)$ are the multipliers. By taking partial derivatives of the Lagrangian with respect to $\mathbf{p}$ and the multipliers, we obtain the Karush-Kuhn-Tucker (KKT) conditions [67]:

$$\frac{\partial E}{\partial p_i} = -\frac{g_{(i)}^2}{p_i^2} + \lambda + \mu_i = 0, \quad i = 1, \dots, N, \quad (19)$$

$$\frac{\partial E}{\partial \lambda} = \sum_{i=1}^N p_i - S = 0, \quad (20)$$

$$\frac{\partial E}{\partial \mu_i} = p_i - 1 + \nu_i^2 = 0, \quad i = 1, \dots, N, \quad (21)$$

$$\frac{\partial E}{\partial \nu_i} = 2\mu_i \nu_i = 0, \quad i = 1, \dots, N, \quad (22)$$

$$\mu_i \geq 0, \quad i = 1, \dots, N. \quad (23)$$

Eq. (22) implies that at least one of μ_i and ν_i must be zero. If $\mu_i = 0$, then from Eq. (19), we have $p_i = \frac{g_{(i)}}{\sqrt{\lambda}}$. Since $p_i \leq 1$, we have $g_{(i)} \leq \sqrt{\lambda}$. If $\nu_i = 0$, then from Eq. (21), we have $p_i = 1$ and $g_{(i)} \geq \sqrt{\lambda}$. Let M be an integer such that $g_{(N-M)} < \sqrt{\lambda}$ and $g_{(N-M+1)} \geq \sqrt{\lambda}$. Then, from Eq. (20), we have:

$$\sqrt{\lambda} = \frac{\sum_{i=1}^{N-M} g_{(i)}}{S - M} := \alpha. \quad (24)$$

Next, we can get:

$$\sum_{i=1}^N \min(g_{(i)}, \alpha) = \sum_{i=1}^{N-M} g_{(i)} + M \cdot \alpha = S \cdot \alpha. \quad (25)$$

Therefore, for $i = 1, \dots, N - M$, we have:

$$p_i^* = \frac{g(i)}{\alpha} = \frac{\min(g(i), \alpha) \cdot S}{\sum_{j=1}^N \min(g(j), \alpha)}. \quad (26)$$

For $i = N - M + 1, \dots, N$, we have:

$$p_i^* = 1 = \frac{\alpha}{\alpha} = \frac{\min(g(i), \alpha) \cdot S}{\sum_{j=1}^N \min(g(j), \alpha)}. \quad (27)$$

Thus, we derive the optimal probabilities. The proof is completed. $\square$

Proposition 1 suggests that the optimal selection probability for each example is proportional to its gradient norm, and larger gradient magnitudes lead to higher probabilities. The probabilities are upper bounded by α to ensure that they remain within the valid range.

However, computing α requires sorting the full set of gradient norms, which incurs a time complexity of $\mathcal{O}(N \log N)$ and poses a bottleneck for large-scale datasets. To address this limitation, we propose a computationally efficient alternative that avoids sorting while maintaining the same optimal solution. Specifically, we iteratively identify and remove examples whose selection probabilities reach 1, and redistribute the probabilities of the remaining examples. Formally, let $\mathcal{R} = \{\mathbf{x}_i \in \mathcal{D} \mid 0 \leq p_i < 1\}$ denote the set of examples with selection probabilities less than 1, and let $\mathcal{R}_{\text{nz}} \subseteq \mathcal{R}$ be the subset with non-zero probabilities. We denote their sizes by $|\mathcal{R}|$ and $|\mathcal{R}_{\text{nz}}|$, respectively. We initialize $\mathcal{R} = \mathcal{R}_{\text{nz}} = \mathcal{D}$ and iteratively update the selection probabilities using:

$$p_i^* = \frac{\|\nabla_{\mathbf{W}} \ell(\mathbf{x}_i)\| \cdot (S - N + |\mathcal{R}|)}{\sum_{j \in \mathcal{R}_{\text{nz}}} \|\nabla_{\mathbf{W}} \ell(\mathbf{x}_j)\|}, \quad \forall i \in \mathcal{R}_{\text{nz}}. \quad (28)$$

Any probability value not less than 1 is bounded at 1, and the corresponding example is removed from the set $\mathcal{R}$. This process is repeated iteratively until all probabilities lie within the valid range $[0, 1]$. Although each iteration of Eq. (28) has a computational complexity of $\mathcal{O}(|\mathcal{R}_{\text{nz}}|)$, the required iteration number in practice is very small, resulting in lower overhead compared to the sorting-based approach.

B. Spike-aware Importance Score

According to Eq. (28), the optimal selection probabilities that minimize gradient variance are proportional to the per-example gradient norm, aligning with the intuition that the gradient norm reflects each example's true contribution to the training process. However, as analyzed in Section III-A, performing a backward pass for each example to compute its gradient norm is prohibitively inefficient. A common alternative is to use the per-example loss as a proxy [55], but this approach proves inadequate in SNNs due to the weak correlation between the loss and the gradient norm.

To address this challenge, we notice that the per-example errors $\{\delta_i^l[t]\}_{i,l,t}$ and the corresponding spikes $\{\mathbf{o}_i^{l-1}[t]\}_{i,l,t}$ have been readily available after the backward pass of BPTT [17]. The per-example gradient norm can thus be reconstructed by computing the outer products between these errors and spikes, as described in Eq. (8). However, calculating the outer products still incurs considerable computational cost.

To further eliminate the outer product operations, we observe that the objective of the variance minimization problem in Eq. (12) is a function of the per-example gradient norm. This allows us to relax the minimization objective by introducing an efficient upper bound G_i on the gradient norm, $G_i \geq \|\nabla_{\mathbf{W}} \ell(\mathbf{x}_i)\|$. Under this relaxation, the optimal selection probabilities are guaranteed to be similar to those derived from the original objective. To this end, we propose an outer-product-free upper bound, referred to as spike-aware importance score, which is formally defined in Proposition 2.

Proposition 2. *The variance minimization objective in Eq. (12) can be relaxed by replacing the per-example gradient norm $\|\nabla_{\mathbf{W}} \ell(\mathbf{x}_i)\|$ with an upper bound G_i , defined as:*

$$G_i = \sum_{l \in \mathbb{L}} \sum_{t=1}^T \|\delta_i^l[t]\| \cdot \|\mathbf{o}_i^{l-1}[t]\|, \quad (29)$$

where $\delta_i^l[t]$ and $\mathbf{o}_i^{l-1}[t]$ represent example i 's errors and spikes at time t and layer l , respectively. The optimal selection probability for each example becomes $\hat{p}_i^* = \frac{G_i \cdot (S - N + |\mathcal{R}|)}{\sum_{j \in \mathcal{R}_{\text{nz}}} G_j}$, $\forall i \in \mathcal{R}_{\text{nz}}$.

Proof. We start by relaxing the original objective in Eq. (12), which aims to minimize the variance of the gradient estimator, by introducing an upper bound $G_i \geq \|\nabla_{\mathbf{W}} \ell(\mathbf{x}_i)\|$ for each training example i . This yields the following relaxed objective:

$$\min_{\mathbf{p}} \sum_{i=1}^N \frac{(1 - p_i) \|\nabla_{\mathbf{W}} \ell(\mathbf{x}_i)\|^2}{p_i} \leq \min_{\mathbf{p}} \sum_{i=1}^N \frac{(1 - p_i) G_i^2}{p_i}. \quad (30)$$

To make this surrogate practically useful, we now derive a form of G_i that is significantly cheaper to compute than the exact gradient norm. We consider two common cases: fully-connected and convolutional layers.

For a fully-connected layer l , the gradient with respect to the weights can be expressed as:

$$\nabla_{\mathbf{W}^l} \ell(\mathbf{x}_i) = \sum_{t=1}^T \delta_i^l[t] \cdot \mathbf{o}_i^{l-1}[t]^\top. \quad (31)$$

By the triangle inequality and sub-multiplicativity of the norm, we obtain:

$$\begin{aligned} \|\nabla_{\mathbf{W}^l} \ell(\mathbf{x}_i)\| &= \left\| \sum_{t=1}^T \delta_i^l[t] \cdot \mathbf{o}_i^{l-1}[t]^\top \right\| \\ &\leq \sum_{t=1}^T \|\delta_i^l[t]\| \cdot \|\mathbf{o}_i^{l-1}[t]\| \\ &\leq \sum_{t=1}^T \|\delta_i^l[t]\| \cdot \|\mathbf{o}_i^{l-1}[t]\|. \end{aligned} \quad (32)$$

For convolutional layers, the weight sharing mechanism changes the structure of the gradient. Each weight kernel slides over spatial regions of the input, and input elements are reused across multiple patches. The gradient with respect to the convolutional kernel is given by:

$$\nabla_{\mathbf{W}^l} \ell(\mathbf{x}_i) = \sum_{t=1}^T \delta_i^l[t] \cdot \text{unfold}(\mathbf{o}_i^{l-1}[t])^\top, \quad (33)$$

where $\text{unfold}(\cdot)$ extracts sliding local patches from the spike tensor and flattens them into columns. Applying norm inequalities again yields:

$$\|\nabla_{\mathbf{w}^l} \ell(\mathbf{x}_i)\| \leq \sum_{t=1}^T \|\delta_i^l[t]\| \cdot \|\text{unfold}(\mathbf{o}_i^{l-1}[t])\|. \quad (34)$$

We now bound the unfolded input norm. Since unfolding duplicates entries across overlapping patches, the norm of the unfolded input can be upper bounded by:

$$\|\text{unfold}(\mathbf{o}_i^{l-1}[t])\| \leq \sqrt{N_{\text{patch}}^l} \cdot \|\mathbf{o}_i^{l-1}[t]\|, \quad (35)$$

where N_{patch}^l denotes the number of patches in layer l . By substituting, we obtain:

$$\|\nabla_{\mathbf{w}^l} \ell(\mathbf{x}_i)\| \leq \sqrt{N_{\text{patch}}^l} \cdot \left(\sum_{t=1}^T \|\delta_i^l[t]\| \cdot \|\mathbf{o}_i^{l-1}[t]\| \right). \quad (36)$$

Let $\mathbb{L}$ denote the set of trainable layers. Then the total gradient norm satisfies:

$$\begin{aligned} \|\nabla_{\mathbf{w}} \ell(\mathbf{x}_i)\| &= \sqrt{\sum_{l \in \mathbb{L}} \|\nabla_{\mathbf{w}^l} \ell(\mathbf{x}_i)\|^2} \\ &\leq \sum_{l \in \mathbb{L}} \|\nabla_{\mathbf{w}^l} \ell(\mathbf{x}_i)\| \\ &\leq N_{\text{patch}}^* \cdot \left(\sum_{l \in \mathbb{L}} \sum_{t=1}^T \|\delta_i^l[t]\| \cdot \|\mathbf{o}_i^{l-1}[t]\| \right), \end{aligned} \quad (37)$$

where $N_{\text{patch}}^* = \max_{l \in \mathbb{L}} \sqrt{N_{\text{patch}}^l}$ is a layer-independent constant that can be ignored when computing normalized probabilities.

We thus obtain the following upper bound:

$$G_i = \sum_{l \in \mathbb{L}} \sum_{t=1}^T \|\delta_i^l[t]\| \cdot \|\mathbf{o}_i^{l-1}[t]\|, \quad (38)$$

which uses quantities already available during BPTT and is cheap to compute. The optimal per-example selection probability thus becomes:

$$\hat{p}_i^* = \frac{G_i \cdot (S - N + |\mathcal{R}|)}{\sum_{j \in \mathcal{R}_{\text{nz}}} G_j}, \quad \forall i \in \mathcal{R}_{\text{nz}}. \quad (39)$$

The proof is completed. $\square$

The spike-aware importance score eliminates the need for outer products and captures the joint influence of binary spikes and spatio-temporal dependent errors, resulting in a strong correlation with the per-example gradient norm. In addition, both components required to compute the score are readily available after the backward pass of BPTT, making the method highly efficient and well-suited for large-scale SNN training.

C. Smoothing Selection Probabilities

Another issue is that examples with very low selection probabilities can produce training instability when selected. This instability arises because, to ensure unbiased gradient estimation, the loss of each selected example is scaled by the inverse of its selection probability, as shown in Eq. (11). SADP

computes selection probabilities in proportion to spike-aware importance scores, and consequently, examples with low scores are assigned very small probabilities. When they are occasionally sampled, their scaled gradients become disproportionately large, potentially destabilizing the training process.

To address this issue, we introduce a smoothing mechanism that enforces a minimum probability. Specifically, if the smallest probability falls below a predefined constant β , we add an offset γ to every spike-aware score G_i such that the smallest probability equals β . The offset γ is calculated by:

$$\frac{(G_{\min} + \gamma) \cdot (S - N + |\mathcal{R}|)}{\sum_{j \in \mathcal{R}_{\text{nz}}} (G_j + \gamma)} = \beta, \quad (40)$$

where $G_{\min}$ is the smallest non-zero score among all examples in $\mathcal{R}$. The smoothed probability is then given by:

$$p_i = \frac{(G_i + \gamma) \cdot (S - N + |\mathcal{R}|)}{\sum_{j \in \mathcal{R}_{\text{nz}}} (G_j + \gamma)}, \quad \forall i \in \mathcal{R}_{\text{nz}}. \quad (41)$$

This smoothing strategy prevents excessive gradient scaling from low-probability examples while maintaining low variance, thereby contributing to more stable and efficient SNN training.

D. Dynamic Pruning Schedule

As training progresses, a growing number of examples become well-learned and contribute less to gradient updates. Motivated by this intuition, we propose a dynamic pruning schedule in which the pruning ratio increases over the course of training: fewer examples are pruned in the early epochs to allow the model to learn broadly from diverse data, while more aggressive pruning is applied in later epochs when less informative examples dominate. The average pruning ratio over epochs keeps unchanged.

Specifically, let K denote the number of training epochs. r and $r_{\max}$ represent the average and maximum pruning ratios, respectively. The pruning ratio r_k at the k -th epoch increase linearly from $2r - r_{\max}$ to $r_{\max}$ over the course of training:

$$r_k = 2r - r_{\max} + \frac{k(2r_{\max} - 2r)}{K}, \quad k = 1, \dots, K. \quad (42)$$

V. EXPERIMENTS

This section presents a comprehensive evaluation of SADP across a wide range of datasets and architectures. In Section V-A, we compare SADP with data pruning baselines in terms of accuracy and training efficiency. Section V-B investigates the contribution of individual components through detailed ablation studies. Section V-C further analyzes gradient approximation and variance reduction, and evaluates the proposed smoothing mechanism for addressing gradient scaling. Finally, Section V-D demonstrates the broad applicability and compatibility of SADP across diverse SNN methods.

A. Performance Evaluation

1) *Experimental Setups: Datasets.* We evaluate SADP on image datasets (CIFAR-10 [56], CIFAR-100 [56], and ImageNet [46]) and neuromorphic datasets (CIFAR-10-DVS [25] and HAR-DVS [26]) using varying pruning ratios. CIFAR-10 [56] and CIFAR-100 [56] contain 50,000 training images

TABLE I
TRAINING CONFIGURATIONS AND HYPERPARAMETER SETTINGS.

Dataset	# Epochs	Optimizer	Learning Rate	Learning Rate Schedule	Batch Size	Weight Decay	Neuronal Decay λ	Threshold ϑ	# Time Steps (T)
CIFAR-10	200	SGD	0.2	Cosine Annealing	128	0.00005	0.1	1.0	4
CIFAR-100	200	SGD	0.2	Cosine Annealing	128	0.00005	0.1	1.0	4
ImageNet (Pretrain)	100	SGD	0.25	Cosine Annealing	512	0.00001	0.2	1.0	1
ImageNet (Finetune)	10	SGD	0.001	Cosine Annealing	128	0	0.2	1.0	4
CIFAR10-DVS	100	AdamW	0.001	Cosine annealing	100	0.0005	0.1	1.0	10
HAR-DVS	100	AdamW	0.001	Cosine Annealing	100	0.0005	0.1	1.0	4

TABLE II
HYPERPARAMETER SETTINGS OF SADP, INCLUDING THE SMOOTHING CONSTANT β AND THE MAXIMUM PRUNING RATIO $r_{\max}$.

Pruning Ratio (%)	β	$r_{\max}(\%)$
30	0.35	60
50	0.3	70
70	0.2	90
90	0.05	100

and 10,000 test images, divided into 10 and 100 classes, respectively. Standard data augmentation is applied to the training set [41], [68], including padding each image by 4 pixels on all sides, followed by a 32×32 crop and random horizontal flipping. Additionally, we employ autoaugment and cutout techniques for further data augmentation [38]. ImageNet [46] comprises 1,000 classes, with 1.2 million images for training and 50,000 images for validation. Standard data augmentation techniques are used [20]. CIFAR10-DVS [25], derived from CIFAR-10, is created by scanning each image through repetitive closed-loop movements in front of a DVS camera [47]. It includes 9,000 training samples and 1,000 testing samples, each with a spatial resolution of 128×128 , which is resized to 48×48 . CIFAR10-DVS retains the 10 classes of CIFAR-10, and no data augmentation techniques are applied to this dataset. HAR-DVS [26] is an event-based human activity recognition (HAR) dataset recorded using the DAVIS346 camera at a spatial resolution of 346×260 . As the *largest* event-based HAR dataset, it encompasses 300 activity classes with a total of 107,646 samples.

Network Architectures. Spiking variants of VGG [57], ResNet [20], [58], and Transformer [59] are adopted.

Training Settings. Table I summarizes training settings. The hyperparameter setup for SADP is provided in Table II, and they are selected based on the ablation studies in Section V-B. We use the same hyperparameters for all datasets at the same pruning ratio. For the experiments on ImageNet with the 35% pruning ratio, the smoothing constant is set to 0.25, and the maximum pruning ratio is 55%. We use the final layer of each network to compute importance scores for computational efficiency. Note that we adopt a two-step training process for ImageNet experiments [49], which includes a pre-training phase with a single time step for 100 epochs on SEW-ResNet34 [20] and 200 epochs on Meta-SpikeFormer [59], followed by a fine-tuning phase with 4 time steps for 10 epochs on SEW-ResNet34 and 50 epochs on Meta-SpikeFormer.

Baselines. We compare SADP against existing data pruning

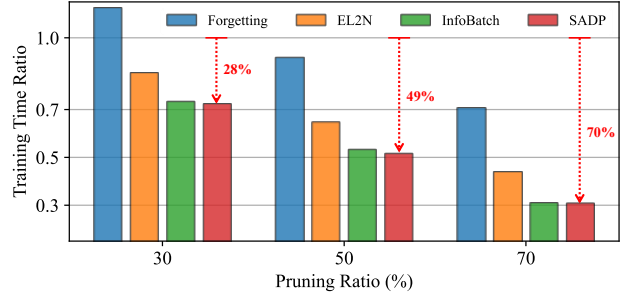


Fig. 3. Comparison of training efficiency on CIFAR-100. The y-axis represents the training time ratio relative to the full-data training. The theoretical maximum training time ratio is equal to the specified pruning ratio.

methods, including Forgetting [53], EL2N [54], and InfoBatch [55], as well as two random baselines (Random Pruning and Reduced Epoch). Random Pruning randomly selects examples at each epoch, while Reduced Epoch uses fewer epochs to match the training iteration number in pruning-based methods. Although these methods were designed for ANNs, they are applicable to SNNs due to their model-agnostic data selection strategies. For fair comparisons, all baseline methods use the same training settings as SADP.

2) *SADP Attains State-of-the-Art Accuracy Across Different Pruning Ratios:* As shown in Table III, SADP consistently outperforms all baseline methods across different pruning ratios, datasets, and architectures. Notably, the accuracy advantage of SADP becomes more pronounced at higher pruning ratios, demonstrating its robustness in retaining informative examples under more aggressive data reduction. Additionally, SADP prunes 30% of the training data on CIFAR-100 and 70% on CIFAR10-DVS without compromising accuracy. These results highlight the strong generalization of SADP and its effectiveness in preserving model accuracy.

3) *SADP Achieves the Theoretical Maximum Training Speedup Owing to Negligible Overhead:* To assess the efficiency of SADP in accelerating SNN training, we measure its wall-clock training time on CIFAR-100 and compare it against baseline methods. Theoretically, a data pruning method with zero overhead can reduce training time by the pruning ratio. However, as shown in Figure 3, both Forgetting [53] and EL2N [54] deviate from this expectation due to the high cost of computing their importance scores. In contrast, SADP consistently achieves training time reductions that closely match the pruning ratios. This indicates that SADP incurs negligible overhead and is highly efficient in practice. Its ability to achieve the theoretical maximum training speedup highlights SADP's

TABLE III

COMPARISON OF SADP, EXISTING DATA PRUNING METHODS, AND RANDOM BASELINES (RANDOM PRUNING AND REDUCED EPOCH) UNDER VARYING PRUNING RATIOS ON BOTH IMAGE AND NEUROMORPHIC DATASETS. REPORTED VALUES ARE THE MEAN ACCURACIES AND STANDARD DEVIATIONS OVER THREE INDEPENDENT RUNS.

Dataset	CIFAR-10			CIFAR-100			CIFAR10-DVS			HAR-DVS		
Net (Time Window)	ResNet18 (T=4)			ResNet19 (T=4)			VGG11 (T=10)			ResNet18 (T=4)		
Full-data Accuracy	95.33 $\pm$ 0.04			80.02 $\pm$ 0.02			77.83 $\pm$ 0.26			49.03		
Pruning Ratio (%)	30	50	70	30	50	70	50	70	90	30	50	70
Reduced Epoch	94.91 $\pm$ 0.08	94.29 $\pm$ 0.11	92.93 $\pm$ 0.05	79.24 $\pm$ 0.07	78.19 $\pm$ 0.11	76.19 $\pm$ 0.11	77.53 $\pm$ 0.29	76.80 $\pm$ 0.29	71.37 $\pm$ 0.29	48.20	46.50	42.23
Forgetting [53]	95.11 $\pm$ 0.17	94.47 $\pm$ 0.09	92.30 $\pm$ 0.19	78.15 $\pm$ 0.17	75.07 $\pm$ 0.31	67.88 $\pm$ 0.30	70.67 $\pm$ 0.83	54.63 $\pm$ 2.74	38.30 $\pm$ 0.94	46.80	43.78	34.37
EL2N [54]	95.16 $\pm$ 0.04	94.63 $\pm$ 0.10	92.07 $\pm$ 0.18	78.10 $\pm$ 0.12	73.51 $\pm$ 0.12	61.48 $\pm$ 0.13	70.90 $\pm$ 0.41	61.60 $\pm$ 0.14	43.97 $\pm$ 0.91	48.08	46.14	35.49
Random Pruning	94.97 $\pm$ 0.06	94.25 $\pm$ 0.03	92.96 $\pm$ 0.15	79.30 $\pm$ 0.06	78.24 $\pm$ 0.05	76.04 $\pm$ 0.08	77.63 $\pm$ 0.05	77.03 $\pm$ 0.37	71.47 $\pm$ 0.12	48.25	46.58	42.87
InfoBatch [55]	94.99 $\pm$ 0.16	94.30 $\pm$ 0.11	93.02 $\pm$ 0.14	79.57 $\pm$ 0.12	78.12 $\pm$ 0.16	76.24 $\pm$ 0.17	77.90 $\pm$ 0.16	77.00 $\pm$ 0.59	72.30 $\pm$ 0.57	48.19	46.84	43.27
SADP	95.22$\pm$0.06	94.82$\pm$0.05	93.65$\pm$0.03	80.01$\pm$0.07	78.63$\pm$0.09	77.10$\pm$0.12	77.97$\pm$0.74	77.80$\pm$0.70	74.80$\pm$0.14	48.51	47.36	44.30

TABLE IV

COMPARISON OF ACCURACY AND WALL-CLOCK TRAINING TIME ON IMAGENET AT A 35% PRUNING RATIO. RESULTS ARE REPORTED WITH SEW-RESNET34 FOR 110 EPOCHS AND META-SPIKEFORMER-31M FOR 250 EPOCHS. THE WALL-CLOCK TIME (IN HOURS) IS CALCULATED AS THE TIME PER GPU MULTIPLIED BY THE NUMBER OF GPUS.

Net	SEW-ResNet34 [20] (T=4)				Meta-SpikeFormer-31M [59] (T=4)		
Method	Full Data	Forgetting [53]	EL2N [54]	SADP	Full Data	InfoBatch [55]	SADP
Accuracy	68.40	65.76	66.97	68.29	77.41	76.46	76.94
Wall-clock Time (Hours)	258	209 ($\downarrow$ 19.0%)	200 ($\downarrow$ 22.5%)	168 ($\downarrow$34.9%)	688	456 ($\downarrow$ 33.7%)	448 ($\downarrow$34.9%)

TABLE V

COMPARISON OF SADP AND BASELINES AT HIGH PRUNING RATIOS ON CIFAR-10.

Method	80%	90%
Random Pruning	91.37 $\pm$ 0.12	87.29 $\pm$ 0.10
Forgetting [53]	87.41 $\pm$ 0.60	71.13 $\pm$ 0.45
EL2N [54]	87.51 $\pm$ 0.15	68.28 $\pm$ 1.41
InfoBatch [55]	91.32 $\pm$ 0.10	87.32 $\pm$ 0.17
SADP	92.23$\pm$0.10	89.04$\pm$0.05

TABLE VI

INFLUENCE OF SADP COMPONENTS.

Dataset	CIFAR-10	CIFAR-100
Pruning Ratio (%)	90	70
SADP	89.04 $\pm$ 0.05	77.10 $\pm$ 0.12
SADP w/ Loss Score	88.37 $\pm$ 0.16	76.68 $\pm$ 0.13
SADP w/o Pruning Schedule	88.17 $\pm$ 0.13	76.49 $\pm$ 0.12
SADP w/o Smoothing	83.37 $\pm$ 0.27	72.16 $\pm$ 0.09

practicality for accelerating SNN training.

4) *SADP Scales Effectively to Large-scale Datasets*: We evaluate the scalability of SADP on the large-scale ImageNet dataset using two representative architectures: SEW-ResNet34 [20] and Meta-SpikeFormer [59]. Table IV shows that SADP consistently outperforms all baselines in both accuracy and training efficiency. Remarkably, it achieves accuracy comparable to full-data training while reducing training time by 35% on both architectures. These results demonstrate not only SADP’s effectiveness on large-scale datasets, but also its generalizability across different SNN architectures.

5) *SADP Significantly Outperforms Baselines at High Pruning Ratios*: To evaluate the effectiveness of SADP under more challenging pruning conditions, we compare its performance with existing data pruning methods at high pruning ratios (80% and 90%) on CIFAR-10 using ResNet18. As shown in Table V, SADP consistently outperforms all baselines across both pruning levels. For instance, at the 90% ratio, SADP

TABLE VII

INFLUENCE OF THE SMOOTH CONSTANT β AND THE MAXIMUM PRUNING RATIO $r_{\max}$ IN SADP ON CIFAR-10 AT A 50% PRUNING RATIO.

β	Accuracy (%)	$r_{\max}$	Accuracy (%)
0	92.36 $\pm$ 0.17	55%	94.62 $\pm$ 0.15
0.1	94.46 $\pm$ 0.06	60%	94.63 $\pm$ 0.05
0.2	94.58 $\pm$ 0.03	65%	94.72 $\pm$ 0.05
0.3	94.82$\pm$0.05	70%	94.82$\pm$0.05
0.4	94.69 $\pm$ 0.10	75%	94.81 $\pm$ 0.08
0.5	94.47 $\pm$ 0.06	80%	94.75 $\pm$ 0.08

achieves a 1.7% higher accuracy than InfoBatch and exceeds EL2N by more than 20%. These results highlight SADP’s robustness and its superior ability to retain essential informative training examples in excessive pruning regimes where existing methods degrade sharply.

B. Ablation Studies

1) *Influence of SADP Components*: We evaluate the impact of each component in SADP. As shown in Table VI, replacing the spike-aware importance score with loss or removing the dynamic pruning schedule leads to noticeable accuracy drops, especially at higher pruning ratios. Removing the smoothing mechanism causes the most severe degradation, highlighting its role in stabilizing training. These results confirm that all three components are crucial to SADP’s effectiveness.

2) *Influence of Smoothing Constant*: We investigate the effect of the smoothing constant β , which balances gradient variance reduction and training stability. Table VII shows that setting $\beta = 0$ leads to a sharp drop in accuracy. Increasing β improves stability and accuracy, with performance peaking at $\beta = 0.3$, indicating an optimal trade-off. Notably, this optimal setting generalizes well across datasets and architectures, as evidenced by the superior accuracy in Table III.

3) *Influence of Dynamic Pruning Schedule*: We analyze the impact of the maximum pruning ratio $r_{\max}$ in the dynamic

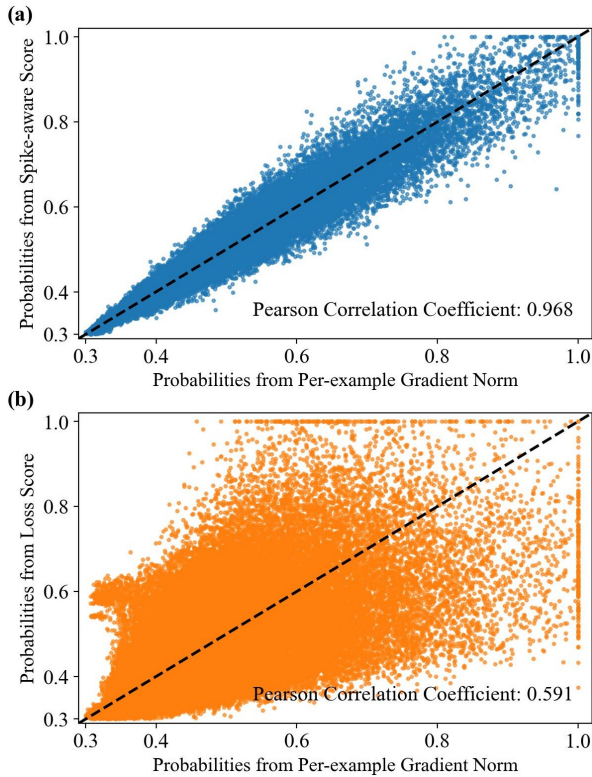


Fig. 4. Comparison of selection probabilities derived from (a) the spike-aware importance score and (b) the loss score, against target probabilities based on the per-example gradient norm. The CIFAR10 dataset with a 50% pruning ratio and a smoothing constant of 0.3 is adopted. Pearson correlation coefficients are provided in the legend.

pruning schedule. Table VII shows that increasing $r_{\max}$ generally improves accuracy, indicating that preserving more data in the early stages is crucial for better accuracy than later epochs. However, setting $r_{\max}$ too high reduces data availability in the final epochs, leading to diminishing returns. Notably, the optimal setting also generalizes well across datasets and architectures.

C. Empirical Analysis of Gradient Approximation, Variance Minimization, and Scaling Factors

1) *Spike-aware Importance Score Strongly Correlates with the Per-example Gradient Norm*: To assess the efficacy of our spike-aware score as a proxy for the gradient norm, we compare the selection probabilities derived from the spike-aware score, the loss score, and the gradient norm for all examples on CIFAR-10. Figure 4 shows that the spike-aware score exhibits a significantly stronger correlation with the gradient norm compared to the loss score. This confirms that our score more accurately captures each example’s training contribution, thereby enabling effective data pruning.

2) *SADP Effectively Reduces Gradient Variance*: We evaluate the gradient variance of SADP in comparison to Random Pruning and InfoBatch [55] on CIFAR-10 [56] using a 50% pruning ratio. Figure 5 shows that SADP consistently maintains significantly lower gradient variance than both baselines throughout training, and the gap in variance widens over epochs. Additionally, SADP’s gradient variance increases with

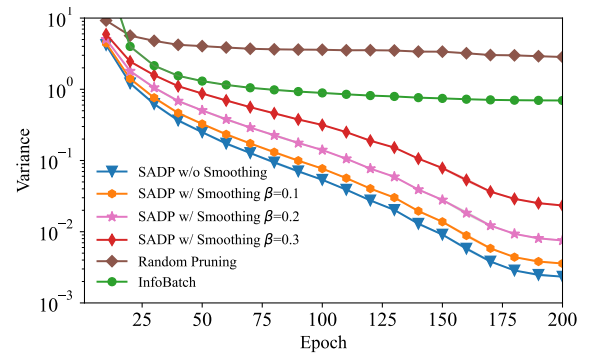


Fig. 5. Comparison of gradient variance. Results are shown on CIFAR-10 at a 50% pruning ratio.

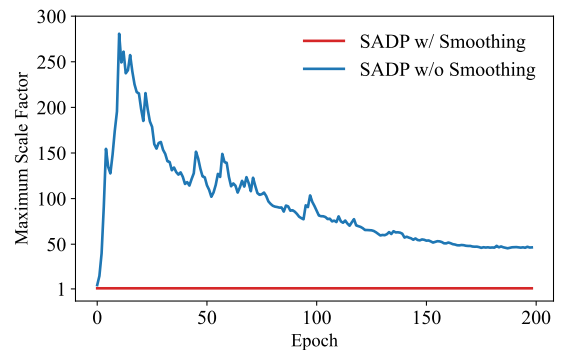


Fig. 6. Comparison of maximum gradient scale factors per epoch with and without the smoothing strategy in SADP. Experiments are conducted on CIFAR-10 at a 50% pruning ratio.

higher values of the smoothing constant, which aligns with our theoretical analysis. These results empirically validate SADP’s ability to minimize gradient variance.

3) *The Smoothing Strategy Effectively Addresses the Gradient Scaling Issue*: To ensure unbiased gradient estimation, SADP rescales the gradient of each selected example by a factor of $\frac{S}{N \cdot p_i}$ according to Eq. (11). However, when an example with a very low selection probability p_i is chosen, the resulting scale factor can become excessively large, amplifying the gradient magnitude and destabilizing the training process. To empirically validate this issue, we record the maximum gradient scale factor per epoch on CIFAR-10 under a 50% pruning ratio, comparing SADP with and without the proposed smoothing mechanism. As shown in Figure 6, SADP without smoothing frequently encounters extreme scaling factors, which leads to unstable updates and lower the accuracy to 92.36%. In contrast, SADP with smoothing maintains stable scale factors throughout training, resulting in a significantly improved accuracy of 94.82%. These results underscore the importance of the smoothing strategy in controlling scaling factors for stable training.

D. Broad Applicability

1) *SADP Integrates Seamlessly with Efficient SNN Models*: To assess the compatibility of SADP with efficient SNN models, we consider two representative examples: the Parallel Spiking Neuron (PSN) [51], which facilitates parallel training along the temporal dimension, and the Temporal Reversible

TABLE VIII

COMPATIBILITY OF SADP WITH EFFICIENT SNN MODELS. WE INTEGRATE SADP WITH A 35% PRUNING RATIO INTO PSN [51] AND T-REVSN [60] AND COMPARE ITS ACCURACY AND WALL-CLOCK TIME (IN HOURS) AGAINST FULL-DATA TRAINING ON IMAGENET FOR 100 EPOCHS.

	Net (Time Window)	Method	Acc. (%)	Time (Hours)
PSN [51]	SEW-ResNet18 (T=4)	Full Data	67.13	104
		SADP	66.94	68 ($\downarrow$ 34.6%)
T-RevSNN [60]	ResNet18 (512) (T=4)	Full Data	71.54	232
		SADP	70.83	152 ($\downarrow$ 34.5%)

TABLE IX

COMPARISON OF DATA PRUNING METHODS UNDER ONLINE LEARNING.

Online Learning Dataset	E-Prop [48]		
	GSC		
Net (Time Window)	Recurrent SNN (T=101)		
Full-data Accuracy	89.33% $\pm$ 0.08		
Prune Ratio (%)	50	70	90
Random Sampling	88.50 $\pm$ 0.14	87.68 $\pm$ 0.12	83.69 $\pm$ 0.14
InfoBatch [55]	88.74 $\pm$ 0.15	88.12 $\pm$ 0.09	84.60 $\pm$ 0.20
SADP	89.31$\pm$0.07	88.75$\pm$0.05	85.46$\pm$0.12
Online Learning Dataset	SLTT [49]		
	CIFAR10-DVS		
Net (Time Window)	VGG11 (T=10)		
Full-data Accuracy	78.10 $\pm$ 0.36		
Pruning Ratio (%)	50	70	90
Random Sampling	77.49 $\pm$ 0.34	77.27 $\pm$ 0.24	71.83 $\pm$ 0.73
InfoBatch [55]	77.50 $\pm$ 0.45	77.00 $\pm$ 0.64	73.00 $\pm$ 0.29
SADP	78.10$\pm$0.70	77.97$\pm$0.45	74.97$\pm$0.17

SNN (T-RevSNN) [60], which improves training efficiency through reversible forward propagation. We integrate SADP with both models on ImageNet under a 35% pruning ratio. As summarized in Table VIII, SADP consistently achieves a 35% reduction in training time without degrading accuracy. These findings highlight that SADP can be combined with advanced SNN models, providing additional training efficiency gains without compromising model performance.

2) *SADP is Compatible with Online Learning Rules:* In addition to conventional BPTT-based training, online learning algorithms offer an attractive alternative for memory-efficient training of SNNs by continuously updating model parameters at each time step. To assess the effectiveness of SADP under this paradigm, we examine two representative methods: E-Prop [48] and SLTT [49]. SADP is combined with the E-Prop rule on the Google Speech Command (GSC) dataset [27] using a 3-hidden-layer recurrent SNN, and with SLTT on CIFAR10-DVS using ResNet18. The results, summarized in Table IX, show that SADP consistently surpasses the InfoBatch [55] and Random Sampling baselines across different pruning ratios. These findings confirm that SADP is fully compatible with online learning rules.

3) *SADP is Compatible with Local Learning Rules:* To further assess its versatility, we evaluate the compatibility of SADP with biologically inspired local learning rules. We integrate SADP with DECOLLE [61] and ELL [62] on CIFAR-10 using ResNet18. Both methods employ layer-wise auxiliary classifiers to locally train each network layer, with DECOLLE using fixed classifiers and ELL adopting trainable ones. As shown in Table X, SADP consistently surpasses Random

TABLE X

COMPARISON OF DATA PRUNING METHODS UNDER LOCAL LEARNING ON THE CIFAR-10 DATASET USING RESNET18 WITH TIME WINDOW T=4.

Local Learning Full-data Accuracy	DECOLLE [61]		
	61.96% $\pm$ 0.44		
Prune Ratio (%)	30	50	70
Random Sampling	60.69 $\pm$ 0.54	59.07 $\pm$ 0.37	56.45 $\pm$ 0.33
InfoBatch [55]	61.39 $\pm$ 0.41	59.88 $\pm$ 0.40	57.57 $\pm$ 0.29
SADP	61.84$\pm$0.22	61.22$\pm$0.45	58.54$\pm$0.33
Local Learning Full-data Accuracy	ELL [62]		
	87.62% $\pm$ 0.07		
Prune Ratio (%)	30	50	70
Random Sampling	86.89 $\pm$ 0.08	85.87 $\pm$ 0.11	84.12 $\pm$ 0.11
InfoBatch [55]	87.26 $\pm$ 0.04	86.45 $\pm$ 0.13	84.51 $\pm$ 0.17
SADP	87.60$\pm$0.07	87.11$\pm$0.09	85.53$\pm$0.10

TABLE XI

COMPARISON OF DATA PRUNING METHODS UNDER EFFICIENT INFERENCE ON THE CIFAR-10 DATASET USING VGG16 WITH TIME WINDOW T=4.

Efficient Inference Full-data Accuracy	Quantization-aware Training [63] (8 Bits)		
	92.04% $\pm$ 0.07		
Prune Ratio (%)	30	50	70
Random Sampling	90.92 $\pm$ 0.10	89.66 $\pm$ 0.13	87.64 $\pm$ 0.07
InfoBatch [55]	91.16 $\pm$ 0.04	90.00 $\pm$ 0.13	87.93 $\pm$ 0.10
SADP	91.65$\pm$0.04	90.68$\pm$0.11	89.05$\pm$0.09
Efficient Inference Full-data Accuracy	Network Pruning [64] (Connectivity=30%)		
	90.45% $\pm$ 0.09		
Prune Ratio (%)	30	50	70
Random Sampling	89.38 $\pm$ 0.11	88.52 $\pm$ 0.03	86.61 $\pm$ 0.22
InfoBatch [55]	89.72 $\pm$ 0.03	88.98 $\pm$ 0.10	87.26 $\pm$ 0.07
SADP	90.44$\pm$0.05	89.82$\pm$0.02	88.45$\pm$0.12

Pruning and InfoBatch [55] across varying pruning ratios under both schemes, confirming its effectiveness in conjunction with local learning paradigms.

The strong generalization of SADP across diverse training schemes can be attributed to the fact that its selection criterion is based on the weight gradient norms. In particular, the spike-aware importance score can be reformulated to quantify each example's contribution to weight gradient norms, thereby ensuring that SADP remains effective and broadly applicable across different training algorithms.

4) *SADP is Compatible with Efficient Inference Methods:* Several methods have been proposed to reduce the energy consumption of SNNs during inference, such as quantization-aware training [63] and network pruning [64]. While these techniques improve inference efficiency, they often incur substantial training cost in order to preserve model accuracy. To evaluate the compatibility of SADP with such approaches, we integrated SADP into both representative frameworks [63], [64] using their publicly available implementations. As shown in Table XI, SADP consistently outperforms existing data pruning baselines across different pruning ratios under both inference-efficient methods. These results confirm that SADP can be combined with existing efficient inference techniques.

VI. CONCLUSION

In this article, we proposed SADP, the first effective and efficient data pruning method for SNNs. SADP introduces a

spike-aware importance score that identifies informative training examples and assigns selection probabilities proportional to these scores to minimize gradient variance. Both theoretical analyses and empirical results demonstrate that SADP reduces gradient variance, accelerates convergence, and provides an effective approximation of the per-example gradient norm. Extensive experiments further show that SADP consistently outperforms existing data pruning methods and can reduce training time by over 30% with lossless accuracy across diverse datasets and architectures. Therefore, this work lays a solid foundation for data-efficient training of SNNs and opens new avenues for scalable and efficient training in large-scale neuromorphic systems [69].

REFERENCES

- [1] K. Roy, A. Jaiswal, and P. Panda, "Towards spike-based machine intelligence with neuromorphic computing," *Nature*, vol. 575, no. 7784, pp. 607–617, 2019.
- [2] G. Li, L. Deng, H. Tang, G. Pan, Y. Tian, K. Roy, and W. Maass, "Brain-inspired computing: A systematic survey and future trends," *Proceedings of the IEEE*, vol. 112, no. 6, pp. 544–584, 2024.
- [3] J. K. Eshraghian, M. Ward, E. Neftci, X. Wang, G. Lenz, G. Dwivedi, M. Bennamoun, D. S. Jeong, and W. D. Lu, "Training spiking neural networks using lessons from deep learning," *Proceedings of the IEEE*, vol. 111, no. 9, pp. 1016–1054, 2023.
- [4] W. Cai, H. Sun, Q. Liao, J. He, D. Chen, D. Yao, and D. Guo, "Robust spatiotemporal prototype learning for spiking neural networks," *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–15, 2025.
- [5] Z. Wang, Y. Zhang, S. Lian, X. Cui, R. Yan, and H. Tang, "Toward high-accuracy and low-latency spiking neural networks with two-stage optimization," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 36, no. 2, pp. 3189–3203, 2025.
- [6] B. Han, F. Zhao, Y. Zeng, and G. Shen, "Developmental plasticity-inspired adaptive pruning for deep spiking and artificial neural networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 47, no. 1, pp. 240–251, 2025.
- [7] W. Maass, "Networks of spiking neurons: The third generation of neural network models," *Neural Networks*, vol. 10, no. 9, pp. 1659–1671, 1997.
- [8] M. Davies, N. Srinivasa, T. Lin, G. N. Chinya, Y. Cao, S. H. Choday, G. D. Dimou, P. Joshi, N. Imam, S. Jain, Y. Liao, C. Lin, A. Lines, R. Liu, D. Mathaikutty, S. McCoy, A. Paul, J. Tse, G. Venkataramanan, Y. Weng, A. Wild, Y. Yang, and H. Wang, "Loihi: A neuromorphic manycore processor with on-chip learning," *IEEE Micro*, vol. 38, no. 1, pp. 82–99, 2018.
- [9] J. Pei, L. Deng, S. Song, M. Zhao, Y. Zhang, S. Wu, G. Wang, Z. Zou, Z. Wu, W. He, F. Chen, N. Deng, S. Wu, Y. Wang, Y. Wu, Z. Yang, C. Ma, G. Li, W. Han, H. Li, H. Wu, R. Zhao, Y. Xie, and L. Shi, "Towards artificial general intelligence with hybrid Tianjic chip architecture," *Nature*, vol. 572, no. 7767, pp. 106–111, Aug. 2019.
- [10] D. Ma, X. Jin, S. Sun, Y. Li, X. Wu, Y. Hu, F. Yang, H. Tang, X. Zhu, P. Lin, and G. Pan, "Darwin3: A large-scale neuromorphic chip with a novel ISA and on-chip learning," *National Science Review*, vol. 11, no. 5, p. nwae102, 03 2024.
- [11] M. Yao, O. Richter, G. Zhao, N. Qiao, Y. Xing, D. Wang, T. Hu, W. Fang, T. Demirci, M. De Marchi, L. Deng, T. Yan, C. Nielsen, S. Sheik, C. Wu, Y. Tian, B. Xu, and G. Li, "Spike-based dynamic computing with asynchronous sensing-computing neuromorphic chip," *Nature Communications*, vol. 15, no. 1, p. 4464, May 2024.
- [12] X. Hao, C. Ma, Q. Yang, J. Wu, and K. C. Tan, "Toward ultralow-power neuromorphic speech enhancement with spiking-fullsubnet," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 36, no. 9, pp. 17350–17364, 2025.
- [13] Q. Yu, Y. Yao, L. Wang, H. Tang, J. Dang, and K. C. Tan, "Robust environmental sound recognition with sparse key-point encoding and efficient multispike learning," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 32, no. 2, pp. 625–638, 2021.
- [14] X. Chen, Q. Yang, J. Wu, H. Li, and K. C. Tan, "A hybrid neural coding approach for pattern recognition with spiking neural networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 46, no. 5, pp. 3064–3078, 2024.
- [15] J. Wu, E. Yilmaz, M. Zhang, H. Li, and K. C. Tan, "Deep spiking neural networks for large vocabulary automatic speech recognition," *Frontiers in Neuroscience*, vol. 14, p. 199, 2020.
- [16] J. Qu, Z. Gao, T. Zhang, Y. Lu, H. Tang, and H. Qiao, "Spiking neural network for ultralow-latency and high-accurate object detection," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 36, no. 3, pp. 4934–4946, 2025.
- [17] Y. Wu, L. Deng, G. Li, J. Zhu, and L. Shi, "Spatio-temporal backpropagation for training high-performance spiking neural networks," *Frontiers in Neuroscience*, vol. 12, p. 331, 2018.
- [18] J. Wu, Y. Chua, M. Zhang, G. Li, H. Li, and K. C. Tan, "A tandem learning rule for effective training and rapid inference of deep spiking neural networks," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 34, no. 1, pp. 446–460, 2023.
- [19] J. Wu, C. Xu, X. Han, D. Zhou, M. Zhang, H. Li, and K. C. Tan, "Progressive tandem learning for pattern recognition with deep spiking neural networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, no. 11, pp. 7824–7840, 2022.
- [20] 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.
- [21] 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.
- [22] C. Zhou, H. Zhang, Z. Zhou, L. Yu, L. Huang, X. Fan, L. Yuan, Z. Ma, H. Zhou, and Y. Tian, "QKFormer: Hierarchical spiking transformer using q-k attention," in *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.
- [23] M. Yao, X. Qiu, T. Hu, J. Hu, Y. Chou, K. Tian, J. Liao, L. Leng, B. Xu, and G. Li, "Scaling spike-driven transformer with efficient spike firing approximation training," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 47, no. 4, pp. 2973–2990, 2025.
- [24] 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.
- [25] H. Li, H. Liu, X. Ji, G. Li, and L. Shi, "Cifar10-dvs: an event-stream dataset for object classification," *Frontiers in Neuroscience*, vol. 11, p. 244131, 2017.
- [26] X. Wang, Z. Wu, B. Jiang, Z. Bao, L. Zhu, G. Li, Y. Wang, and Y. Tian, "Hardvs: Revisiting human activity recognition with dynamic vision sensors," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 38, no. 6, pp. 5615–5623, Mar. 2024.
- [27] B. Cramer, Y. Stradmann, J. Schemmel, and F. Zenke, "The heidelberg spiking data sets for the systematic evaluation of spiking neural networks," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 33, no. 7, pp. 2744–2757, 2022.
- [28] Q. Yu, H. Tang, K. C. Tan, and H. Li, "Rapid feedforward computation by temporal encoding and learning with spiking neurons," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 24, no. 10, pp. 1539–1552, 2013.
- [29] A. Taherkhani, A. Belatreche, Y. Li, and L. P. Maguire, "A supervised learning algorithm for learning precise timing of multiple spikes in multilayer spiking neural networks," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 29, no. 11, pp. 5394–5407, 2018.
- [30] Q. Yu, H. Li, and K. C. Tan, "Spike timing or rate? neurons learn to make decisions for both through threshold-driven plasticity," *IEEE Transactions on Cybernetics*, vol. 49, no. 6, pp. 2178–2189, 2019.
- [31] J. Wu, Y. Chua, M. Zhang, H. Li, and K. C. Tan, "A spiking neural network framework for robust sound classification," *Frontiers in Neuroscience*, vol. 12, p. 836, 2018.
- [32] E. O. Neftci, H. Mostafa, and F. Zenke, "Surrogate gradient learning in spiking neural networks: Bringing the power of gradient-based optimization to spiking neural networks," *IEEE Signal Processing Magazine*, vol. 36, no. 6, pp. 51–63, 2019.
- [33] P. Werbos, "Backpropagation through time: what it does and how to do it," *Proceedings of the IEEE*, vol. 78, no. 10, pp. 1550–1560, 1990.
- [34] J. Ding, J. Zhang, T. Huang, J. K. Liu, and Z. Yu, "Assisting training of deep spiking neural networks with parameter initialization," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 36, no. 8, pp. 15 015–15 028, 2025.
- [35] H. Zheng, Y. Wu, L. Deng, Y. Hu, and G. Li, "Going deeper with directly-trained larger spiking neural networks," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 12, pp. 11 062–11 070, May 2021.

- [36] C. Duan, J. Ding, S. Chen, Z. Yu, and T. Huang, "Temporal effective batch normalization in spiking neural networks," *Advances in Neural Information Processing Systems*, vol. 35, pp. 34 377–34 390, 2022.
- [37] Y. Guo, Y. Zhang, Y. Chen, W. Peng, X. Liu, L. Zhang, X. Huang, and Z. Ma, "Membrane potential batch normalization for spiking neural networks," in *ICCV*, October 2023, pp. 19 420–19 430.
- [38] Z. Wang, R. Jiang, S. Lian, R. Yan, and H. Tang, "Adaptive smoothing gradient learning for spiking neural networks," in *International Conference on Machine Learning*. PMLR, 2023, pp. 35 798–35 816.
- [39] S. Lian, J. Shen, Q. Liu, Z. Wang, R. Yan, and H. Tang, "Learnable surrogate gradient for direct training spiking neural networks," in *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence, IJCAI-23*. International Joint Conferences on Artificial Intelligence Organization, 8 2023, pp. 3002–3010, main Track.
- [40] S. Zhang, Q. Yang, C. Ma, J. Wu, H. Li, and K. C. Tan, "Tc-lif: A two-compartment spiking neuron model for long-term sequential modelling," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 38, no. 15, pp. 16 838–16 847, Mar. 2024.
- [41] W. Fang, Z. Yu, Y. Chen, T. Masquelier, T. Huang, and Y. Tian, "Incorporating learnable membrane time constant to enhance learning of spiking neural networks," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2021, pp. 2661–2671.
- [42] B. Yin, F. Corradi, and S. M. Bohté, "Accurate and efficient time-domain classification with adaptive spiking recurrent neural networks," *Nature Machine Intelligence*, vol. 3, no. 10, pp. 905–913, 2021.
- [43] H. Zheng, Z. Zheng, R. Hu, B. Xiao, Y. Wu, F. Yu, X. Liu, G. Li, and L. Deng, "Temporal dendritic heterogeneity incorporated with spiking neural networks for learning multi-timescale dynamics," *Nature Communications*, vol. 15, no. 1, p. 277, 2024.
- [44] C. Ma, X. Chen, Y. Li, Q. Yang, Y. Wu, G. Li, G. Pan, H. Tang, K. C. Tan, and J. Wu, "Spiking neural networks for temporal processing: Status quo and future prospects," *arXiv preprint arXiv:2502.09449*, 2025.
- [45] Y. Hu, H. Tang, and G. Pan, "Spiking deep residual networks," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 34, no. 8, pp. 5200–5205, 2023.
- [46] 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.
- [47] P. Lichtsteiner, C. Posch, and T. Delbruck, "A 128×128 120 db 15 μ s latency asynchronous temporal contrast vision sensor," *IEEE Journal of Solid-State Circuits*, vol. 43, no. 2, pp. 566–576, 2008.
- [48] G. Bellec, F. Scherr, A. Subramoney, E. Hajek, D. Salaj, R. Legenstein, and W. Maass, "A solution to the learning dilemma for recurrent networks of spiking neurons," *Nature Communications*, vol. 11, p. 3625, 2020.
- [49] Q. Meng, M. Xiao, S. Yan, Y. Wang, Z. Lin, and Z.-Q. Luo, "Towards memory-and time-efficient backpropagation for training spiking neural networks," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023, pp. 6166–6176.
- [50] M. Xiao, Q. Meng, Z. Zhang, D. He, and Z. Lin, "Online training through time for spiking neural networks," *Advances in Neural Information Processing Systems*, vol. 35, pp. 20 717–20 730, 2022.
- [51] W. Fang, Z. Yu, Z. Zhou, D. Chen, Y. Chen, Z. Ma, T. Masquelier, and Y. Tian, "Parallel spiking neurons with high efficiency and ability to learn long-term dependencies," in *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.
- [52] X. Chen, J. Wu, C. Ma, Y. Yan, Y. Wu, and K. C. Tan, "Pmsn: A parallel multi-compartment spiking neuron for multi-scale temporal processing," *arXiv preprint arXiv:2408.14917*, 2024.
- [53] M. Toneva, A. Sordoni, R. T. des Combes, A. Trischler, Y. Bengio, and G. J. Gordon, "An empirical study of example forgetting during deep neural network learning," in *International Conference on Learning Representations*, 2019.
- [54] M. Paul, S. Ganguli, and G. K. Dziugaite, "Deep learning on a data diet: Finding important examples early in training," in *Advances in Neural Information Processing Systems*, 2021.
- [55] Z. Qin, K. Wang, Z. Zheng, J. Gu, X. Peng, Xu Zhao Pan, D. Zhou, L. Shang, B. Sun, X. Xie, and Y. You, "Infobatch: Lossless training speed up by unbiased dynamic data pruning," in *The Twelfth International Conference on Learning Representations*, 2024.
- [56] A. Krizhevsky, "Learning multiple layers of features from tiny images," *Tech. Rep.*, 2009.
- [57] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.
- [58] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 770–778.
- [59] 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," in *The Twelfth International Conference on Learning Representations*, 2024.
- [60] J. Hu, M. Yao, X. Qiu, Y. Chou, Y. Cai, N. Qiao, Y. Tian, B. Xu, and G. Li, "High-performance temporal reversible spiking neural networks with $\mathcal{O}(l)$ training memory and $\mathcal{O}(1)$ inference cost," in *Proceedings of the 41st International Conference on Machine Learning*, vol. 235. PMLR, 21–27 Jul 2024, pp. 19 516–19 530.
- [61] J. Kaiser, H. Mostafa, and E. Neftci, "Synaptic plasticity dynamics for deep continuous local learning (decolle)," *Frontiers in Neuroscience*, vol. 14, p. 424, 2020.
- [62] C. Ma, R. Yan, Z. Yu, and Q. Yu, "Deep spike learning with local classifiers," *IEEE Transactions on Cybernetics*, vol. 53, no. 5, pp. 3363–3375, 2023.
- [63] W. Wei, M. Zhang, Z. Zhou, A. Belatreche, Y. Shan, Y. Liang, H. Cao, J. Zhang, and Y. Yang, "QP-SNN: Quantized and pruned spiking neural networks," in *The Thirteenth International Conference on Learning Representations*, 2025.
- [64] Y. Li, Q. Xu, J. Shen, H. Xu, L. Chen, and G. Pan, "Towards efficient deep spiking neural networks construction with spiking activity based pruning," in *International Conference on Machine Learning*. PMLR, 2024, pp. 29 063–29 073.
- [65] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems*, vol. 32. Curran Associates, Inc., 2019.
- [66] W. Fang, Y. Chen, J. Ding, Z. Yu, T. Masquelier, D. Chen, L. Huang, H. Zhou, G. Li, and Y. Tian, "Spikingjelly: An open-source machine learning infrastructure platform for spike-based intelligence," *Science Advances*, vol. 9, no. 40, p. eadi1480, 2023.
- [67] S. P. Boyd and L. Vandenberghe, *Convex optimization*. Cambridge university press, 2004.
- [68] Q. Yu, C. Ma, S. Song, G. Zhang, J. Dang, and K. C. Tan, "Constructing accurate and efficient deep spiking neural networks with double-threshold and augmented schemes," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 33, no. 4, pp. 1714–1726, 2022.
- [69] D. Kudithipudi, C. Schuman, C. M. Vineyard, T. Pandit, C. Merkel, R. Kubendran, J. B. Aimone, G. Orchard, C. Mayr, R. Benosman *et al.*, "Neuromorphic computing at scale," *Nature*, vol. 637, no. 8047, pp. 801–812, 2025.