SAFA-SNN: Sparsity-Aware On-Device Few-Shot Class-Incremental Learning with Fast-Adaptive Structure of Spiking Neural Network

Huijing Zhang [1] **Muyang Cao** [1] **Linshan Jiang** [2] **Xin Du** [1] **Di Yu** [1]
Changze Lv [3] **Shuiguang Deng** [1]
[1] Zhejiang University   [2] National University of Singapore   [3] Fudan University

Abstract

Continuous learning of novel classes is crucial for edge devices to preserve data privacy and maintain reliable performance in dynamic environments. However, the scenario becomes particularly challenging when data samples are insufficient, requiring on-device few-shot class-incremental learning (FSCIL) to maintain consistent model performance. Although existing work has explored parameter-efficient FSCIL frameworks based on artificial neural networks (ANNs), their deployment is still fundamentally constrained by limited device resources. Inspired by neural mechanisms, Spiking neural networks (SNNs) process spatiotemporal information efficiently, offering lower energy consumption, greater biological plausibility, and compatibility with neuromorphic hardware than ANNs. In this work, we present an SNN-based method for On-Device FSCIL, i.e., Sparsity-Aware and Fast Adaptive SNN (SAFA-SNN). We first propose sparsity-conditioned neuronal dynamics, in which most neurons remain stable while a subset stays active, thereby mitigating catastrophic forgetting. To further cope with spike non-differentiability in gradient estimation, we employ zeroth-order optimization. Moreover, during incremental learning sessions, we enhance the discriminability of new classes through subspace projection, which alleviates overfitting to novel classes. Extensive experiments conducted on two standard benchmark datasets (CIFAR100 and Mini-ImageNet) and three neuromorphic datasets (CIFAR-10-DVS, DVS128gesture, and N-Caltech101) demonstrate that SAFA-SNN outperforms baseline methods, specifically achieving at least 4.01% improvement at the last incremental session on Mini-ImageNet and 20% lower energy cost over baseline methods with practical implementation.

1 Introduction

In many real-world scenarios, mobile users are exposed to dynamic contexts, and data collection often spans years, making it challenging to acquire sufficient data for continual batch learning (Wang et al., 2022b). For example, in intelligent surveillance systems, the monitoring framework is typically trained on a predefined set of behavioral categories, yet novel anomalous behaviors may emerge over time. To ensure the system to learn newly emerging classes with scarce samples from a sequence of tasks, on-device Few-Shot Class-Incremental Learning (FSCIL) is introduced. Spiking Neural Networks (SNNs), different from traditional Artificial Neural Networks (ANNs), have recently achieved competitive results in computer vision tasks. SNNs offer natural energy-saving characteristics through event-driven computation (Maass, 1997; Yao et al., 2023; Zhang et al., 2024; Fan et al., 2024), significantly reducing unnecessary cost (Lv et al., 2024), which make them attractive for edge devices, where memory and power budgets are stringent (Yu et al., 2024b).

However, FSCIL faces two significant challenges, i.e., catastrophic forgetting and overfitting (Chen et al., 2019; Snell et al., 2017). Catastrophic forgetting reflects the model's difficulty adapting to new tasks while preserving knowledge from previously learned classes, necessitating a delicate balance between plasticity and stability. Overfitting occurs when the model memorizes limited training
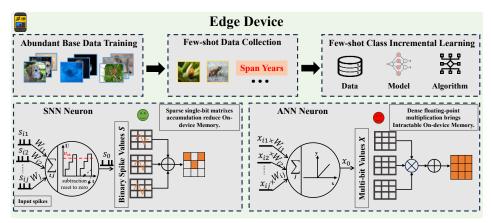
Figure 1: Top: the on-Device FSCIL scenario includes three stages: base data training, few-shot data collection and learning. Bottom: the comparison of SNN and ANN neuron on devices.

samples, leading to a loss of generalization. On the one hand, recent advanced solutions tackle these challenges by parameter-efficient fine-tuning (PEFT) (Liu et al., 2024a), which involves fine-tuning parameters on top of a frozen pre-trained model. Nonetheless, these approaches are ineffective for on-device settings due to limited memory capacity (i.e., 4–12 GB) (Ma et al., 2023). On the other hand, concurrent on-device SNNs concentrate on integrating SNN algorithms with specialized neuromorphic hardware, which shifts the consideration of the above problems to the later model. However, most existing neuromorphic systems are still trained offline and remain static during development (Safa, 2024), presenting a dependency on abundant labeled classes.

Besides, on-device FSCIL scenarios present practical challenges, including long-term data collection spanning years, extremely scarce labeled samples, and the persistent need for model updates. Edge devices cannot accumulate comprehensive data on all classes due to the limited storage capacity, and must simultaneously perform low power inference while continuously learning from real-time data streams (Liu et al., 2024a; Wang et al., 2024). We provide the process of on-device FSCIL scenario and SNN-ANN neurons comparison in Figure 1.

To tackle the above challenges, we propose a Sparsity-Aware, Fast-Adaptive SNN (SAFA-SNN) with neuronal dynamics for general on-device FSCIL. First, we align spike outputs via dynamic thresholds in incremental sessions to alleviate catastrophic forgetting. Specifically, we maintain most neurons stable and keep others active in incremental learning. Then, considering the non-differentiable gradient in SNNs, we adopt zeroth-order optimization in backpropagation. Finally, prototypes are updated via subspace projection, which calibrates biased predictions to boost incremental learning. In addition, we conduct practical implementation and extensive experiments on five datasets, demonstrating the performance of SAFA-SNN.

The main contributions of this paper are as follows:

- We emphasize practical challenges of on-device FSCIL, where both training and inference are performed on resource-constrained devices using energy-efficient, hardware-friendly SNNs, contrasting with ANN-based FSCIL that depends on large-scale server models.

- We propose a sparsity-aware FSCIL method, called SAFA-SNN, with three key features. First, incorporating spikes for prediction, we propose the Sparsity-Aware Dynamic SNN. Then, to prevent the non-differentiable spikes in back propagation, we adopt zeroth-order optimization within neuronal dynamics. Finally, we enhance the discriminability of prototypes by subspace projection in incremental inference. To the best of our knowledge, this is the first SNN-based solution towards general on-device FSCIL.

- To prove the effectiveness of our proposed SAFA-SNN for on-device FSCIL, we implement SAFA-SNN on realistic devices (i.e., Jetson Orin AGX), and evaluate on five datasets (CIFAR100, MiniImageNet, CIFAR-10-DVS, DVS128gesture, and N-Caltech101), and various models (Spiking VGG, Spiking Resnet, and Spikingformer), demonstrating SAFA-SNN achieves excellent performance and notable sparsity advantages. We further compare the actual on-device energy consumption, demonstrating our efficiency advantage.

2 RELATED WORK

2.1 FEW-SHOT CLASS-INCREMENTAL LEARNING

To the best of our knowledge, existing methods for few-shot class-incremental learning (FSCIL) are mostly ANN-based, with little exploration in SNNs. The primary work (Tao et al., 2020) preserves feature topology using a neural gas network. Researchers (Zhang et al., 2021) model prototypes as nodes in graph attention network to propagate context information. Existing methods (Zhou et al., 2022; Wang et al., 2023) commonly adopt prototypes to replace traditional MLPs and refine them (Song et al., 2023). Some studies (Shi et al., 2021; Liu et al., 2023) tackle class imbalance by exploring optimal base class distributions. Recent advances (Park et al., 2024; Sun et al., 2024; Wang et al., 2024) train a few parameters (prompts) from the frozen pre-trained model, achieving high accuracy. However, memory overhead limits their use for resource-constrained devices. Prior work focuses on a specific processor (Huo et al., 2025), while we explore FSCIL on general devices.

2.2 SPIKING NEURAL NETWORKS WITH DYNAMIC THRESHOLD

In the realm of SNNs, the dynamic threshold mechanism aims to manipulate the threshold spontaneously. BDETT computes thresholds via average membrane potentials for neuronal homeostasis (Ding et al., 2022), but excessive spiking remains. Highly active neurons skew the average, causing aggressive firing and reduced sparsity. LTMD learns initial membrane potentials for heuristic neuron thresholding (Wang et al., 2022a), but neuron behaviors across layers remain interdependent in spatial and temporal processing. Soft threshold schemes enable adaptive layer-wise sparsity (Chen et al., 2023), but independent pruning causes error accumulation, affecting overall model performance. Efforts with combinations of adaptive threshold methods have led to innovative approaches (Wei et al., 2023; Hao et al., 2024). Our paper further tackles FSCIL challenges via sparsity-aware dynamic thresholds that shift networks to task-specific local contexts.

2.3 ON-DEVICE SNN TRAINING AND INFERENCE

Previous studies on on-device SNNs can be classified into two categories: inference, and training with updating. The lightweight SNN is a crucial application for resource-constrained edge devices (Yu et al., 2024b). For instance, Lite-SNN (Liu et al., 2024b) proposes a spatial-temporal compression strategy to enable low memory and computational cost. Most researchers focus on efficient quantization for SNN-training like SNN pruning (Wei et al., 2025; Qiu et al., 2025; Yu et al., 2025) to enhance SNN deployment. Current SNN training and updating focus on gradient update (Anumasa et al., 2024) or local SNN training optimization (Mukhoty et al., 2023; Yu et al., 2024a) to enable low-latency SNN training on devices. In this work, we explore both SNN-batch training on abundant base classes and inference in FSCIL phases, which typically occur in the over-changing on-device environments, achieving high performance and low energy consumption.

3 PRELIMINARIES

3.1 SPIKING NEURAL NETWORK

We use the leaky integrate-and-fire (LIF) neuron model, a simplified but common approach to neuromorphic computing that mimics biological neurons through charging, leakage, and firing. The process of state updating in LIF neurons follows certain principles:

$$U'_t = \gamma I_t + U_{t-1}, I_t = f(x; \theta) \tag{1}$$

$$S_t = H(U'_t - U_{th}) \tag{2}$$

$$U_t = S_t U_{reset} + (1 - S_t)U'_t \tag{3}$$

where $H = \mathbf{1}_{U_{t'} > U_{th}}$, γ is the time constant, I_t is the spatial input calculated by applying function f with x as input and θ as learnable parameters, and U'_t is a temporary voltage at time step t, whose value instantly changes. At time step t, if the membrane U'_t rises and reaches the threshold U_{th}, a spike denoted by S_t is generated as 1, and U'_t will be reset to U_{reset}; otherwise, S_t remains 0.

Afterwards, U'_t changes to U_t. SNNs present challenges in terms of the non-differentiable Equation (2). The backpropagation of directly training SNNs can be described as

$$\frac{\partial L}{\partial W} = \sum_{t=1}^{T} \frac{\partial L}{\partial S_t} \frac{\partial S_t}{\partial U_t} \frac{\partial U_t}{\partial I_t} \frac{\partial I_t}{\partial W}, \tag{4}$$

where L denotes the loss, W is the weight of input, and $\frac{\partial S_t}{\partial U_t}$ represents the non-differentiable gradient in Equation 2, which is typically replaced by surrogate functions (Wu et al., 2018). However, surrogate functions introduce gradient errors. To address this issue, we employ zeroth-order optimization in backpropagation, inspired by prior work (Mukhoty et al., 2023).

3.2 CLASS PROTOTYPE

Class prototypes are widely used in few-shot learning based on the observation (Snell et al., 2017). In FSCIL, researchers use prototypes as dynamic classifiers (Zhao et al., 2021; Zhang et al., 2021). In detail, they decouple the FSCIL model into a feature encoder $\phi_\theta(\cdot)$ with parameters θ and a linear classifier W, and freeze $\phi_\theta(\cdot)$ during incremental sessions, with only the classifier being updated. For each new class i, the weight W_i in the classifier is parameterized by the mean embeddings of each class, i.e., the prototype P_i, which can be denoted as $P_i = \frac{1}{K} \sum_{j=1}^{|D_s|} \mathbb{I}_{\{y_j=i\}} \phi(x_j)$, where K denotes the number of samples in the class $\mathcal{C}_i$.

4 ON-DEVICE FSCIL PROBLEM

On-device FSCIL problem depicts data with scarce samples always coming, device models need to maintain performance on all seen classes while maintaining lower latency and security, enhanced data privacy, and reliable performance under resource-limited scenarios. The encountered data are always scarce, for example, users take photos at an average of 4.9 every day (Gong et al., 2024).

Assuming FSCIL tasks contain a base session and a sequence of S incremental sessions, the training data available for each session can be denoted as $\{D_0^{train}, D_1^{train}, ..., D_S^{train}\}$. $D_s^{train} = \{(x_i^t, y_i^t)\}_{i=0}^{|D_s^{train}|}$, where $x_i^s \in X_s$ is the training instance and $y_i^s \in Y_s$ is its corresponding label in session s. Each session s is disjoint with another session s', i.e., $D_s^{train} \cap D_{s'}^{train} = \emptyset$. The training data in session 0 have abundant instances while the incremental session has much smaller instances, i.e., $|D_0^{train}| >> |D_{s>0}^{train}|$. Each incremental session has training sets in the form of N-way K-shot, i.e, there are N classes and each class has K samples. In session s, the model's performance is assessed on validation sets from all encountered datasets ($\leq s$) to minimize the expected risk $\mathcal{R}(f, s)$ over all seen classes:

$$\mathbb{E}_{(x_i, y_i) \sim \mathcal{D}_0^{train} \cup \cdots \cup \mathcal{D}_s^{train}} \left[\mathcal{L} \left(f(x_i; \mathcal{D}_s^{train}, \theta^{s-1}), y_i \right) \right] \tag{5}$$

where the algorithm f constructs a new model using current training data $\mathcal{D}_s^{train}$ and parameters θ^{s-1} to minimize the loss $\mathcal{L}$. The general goal is to continually minimize the risk of each new session, i.e., $\sum_{s=0}^{S} \mathcal{R}(f, s)$. Note that in this paper, "session" and "task" are used interchangeably.

In many real-world scenarios, when FSCIL applications are deployed on resource-limited edge devices, the learning efficiency w.r.t. both few-shot inference speed and memory footprint becomes a crucial metric. Being able to quickly adapt deep prediction models in low computational cost at the edge is necessary to better suit the needs of personal mobile users. However, these aspects are rarely explored in prior FSCIL research, which motivates us to solve these on-device FSCIL challenges beyond catastrophic forgetting and overfitting.

5 METHODOLOGY

5.1 MOTIVATION

Conventional FSCIL methods tackle catastrophic forgetting by freezing most parameters to preserve learned knowledge, keeping only a few parameters trainable in incremental sessions (Zhou et al., 2022; Zhao et al., 2023). Inspired by this, we propose neuronal dynamics that balance plasticity
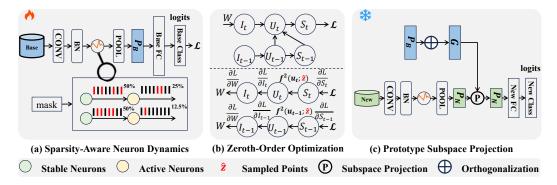
Figure 2: SAFA-SNN framework include three main components: (a) Training abundant data and selecting active and stable neurons by masks. (b) Top: forward propagation through FSCIL process; Bottom: backpropagation using zeroth-order optimization only in the base class training. (c) Freezing backbones and updating the prototypes by subspace projection in the incremental inference.

and stability by dividing neurons into active neurons and stable types. Active neurons facilitate knowledge updating while stable neurons align with base-class representations to preserve learned knowledge. Despite ongoing efforts, overfitting remains a challenge. Subspace projection, using class-specific matrices, offers robustness in few-shot learning (Simon et al., 2020). Inspired by this, we propose subspace projection to calibrate biased prototypes in incremental inference.

Algorithm 1 Overall SAFA-SNN Algorithm

Input: Dataset D_0, $D_{s>0}$, membrane $\mathbf{U}$, feature extractor ϕ
Parameter: $\eta, T, S, \mathbf{M}, \Theta$
Output: Spike trains S_t, parameters θ

1: Random Initialize membrane $\mathbf{U}$
2: Initialize mask $\mathbf{M}$ by randomly setting $\eta\%$ adaptive neurons 1 and setting others 0 in each channel
3: Initialize threshold Θ with zeros
4: **for** $t = 1$ to time step T **do**
5:     $\mathbf{U_t} = \tau U_{t-1} + X_t$
6:     $S_t = \mathbb{I}(\mathbf{U_t} \geq \Theta)$
7:     $\mathbf{U_t} = \mathbf{U_t}(1 - S_t)$
8: **end for**
9: Update Θ by Eq. (7)
10: Sample point $z_i \sim \mathcal{N}(0, 1), i = 1, 2, ..., S$
11: **if** $|x| < \delta|z_i|$ **then**
12:     Estimate the gradient $\hat{g}_i = m_i \cdot |z_i|$
13: **end if**
14: Estimate $\frac{\partial L}{\partial x}$ by all selected points $\frac{\partial L}{\partial y}\left(\frac{1}{2\delta S}\Sigma_{i=1}^{S}\hat{g}_i\right)$
15: Conduct base session optimization L and update $\phi(\theta)$ by Loss in Eq. (13)
16: Compute the projection subspace by Eq. (14)
17: Conduct incremental inference with $\tilde{\mathbf{P}}$ in Eq. (15)

5.2 OVERVIEW OF SAFA-SNN

SAFA-SNN for on-device FSCIL comprises three parts: a sparsity-aware dynamic mechanism to mitigate forgetting, the formulation of zeroth-order optimization for non-differentiable spikes, and a fast-adaptive prototype subspace projection for fast adaptation. Figure 2 illustrates the overview of SAFA-SNN in FSCIL, and the full FSCIL procedure is detailed in Algorithm 1.

5.3 SPARSITY-AWARE NEURONAL DYNAMICS

Complex neuronal dynamics force a trade-off between biological realism and computational complexity in simulations. To address catastrophic forgetting while maintaining the model's discrim-

inability, most neurons remain stable by aligning with spike firing to preserve stability, while others adapt to incremental learning. Define the firing rate of neuron i with spike output $\boldsymbol{S}_t^{(i)}$ as $\bar{r}_i = \frac{1}{T}\Sigma_{t=0}^T \boldsymbol{S}_t^{(i)}$, which can also be denoted as sparsity. However, the formulation may limit the expressiveness of latent dynamics for real-world applications. We introduce a channel-wise mask $\mathbf{M} = \{m_c\}, c = 1, 2, ..., C$, where C is the number of channels and $m_c = \mathbf{1}_{c \leq \lfloor \eta C \rfloor}$, where $\eta \in (0,1)$ is the adaptive ratio controlling active neurons to enable new class learning. Other neurons align their firing with old neurons to reduce spiking bias in new tasks.

$$\mathbf{A} = \beta(\mathbf{I} - \mathbf{M}) + \gamma \mathbf{M} \tag{6}$$

where β and γ are hyper-parameters. Finally, the updated threshold can be described as

$$\boldsymbol{\Theta}_{t+1} = \boldsymbol{\Theta}_t - \mathbf{A}\,(\bar{r}_n - \bar{r}_b) \tag{7}$$

where $\bar{r}_n$ and $\bar{r}_b$ denote firing rates of neurons in incremental and base tasks, respectively.

5.4 ZEROTH-ORDER OPTIMIZATION

Multi-point estimate. Zeroth-order optimization is beneficial to computation-difficult or infeasible gradient problems, since it can approximate the full gradients or stochastic gradients through function value (Liu et al., 2020). We start by presenting the gradient estimate of $f(x)$ with two-point directional derivative as

$$\hat{\nabla}f(x) := \frac{\phi(d)}{\mu}\Sigma_{i=1}^b[f(x + \delta\mathbf{u}_i) - f(x - \delta\mathbf{u}_i)] \tag{8}$$

where b is the number of i.i.d. samples $\{\mathbf{u}_i\}_{i=0}^b$, and d is a dimension-dependent factor. Let $g(u)$ denotes the central finite difference approximation $\frac{\partial S_t}{\partial U_t}$ with membrane potential $u = u_t - u_{th}$. Then, for $u \in \mathbb{R}^n$,

$$g^2(u; \delta, \boldsymbol{z}) = \frac{H(u + \delta\boldsymbol{z}) - H(u - \delta\boldsymbol{z})}{2\delta}\boldsymbol{z} = \begin{cases} 0, |u| > \delta\,|\boldsymbol{z}| \\ \frac{|\boldsymbol{z}|}{2\delta}, |u| < \delta\,|\boldsymbol{z}| \end{cases} \tag{9}$$

where $\boldsymbol{z}$ obeys a specific distribution P over b empirical samples $\{z_i\}_{i=0}^b$ and H is the heaviside function defined in Section 3.1. Assessing whether perturbations $u + z\delta$ and $u - z\delta$ induce a spike in the neuron allows for an approximation of the gradient. Through random permutation, $g^2(u)$ accommodates the influence of all neighbors, for a balanced depiction of local dynamics. The approximation to $\hat{\nabla}g(x)$ at a given input u can be computed as

$$\frac{\partial \mathcal{S}_t}{\partial u_t} := \frac{1}{b}\sum_{i=1}^b g^2(u; \delta, \mathbf{z}_i), \tag{10}$$

Gradients are typically approximated by averaging multiple perturbed samples, where each perturbation produces a local gradient, resulting in a more reliable estimate (See Appendix A.1).

Convergence analysis of ZOO. We start with some conventional assumptions for convergence analysis in zeroth-order optimization literature (See Appendix A.2). The non-differentiability of Equation 2 renders the entire optimization problem non-convex. The convergence is evaluated using the first-order stationary condition via the squared gradient norm (Proof in Appendix A.3):

$$\frac{1}{\mathcal{T}}\sum_{t=1}^{\mathcal{T}} \mathbb{E}\left[\|g(u)\|_2^2\right] = O(\delta^2 + \frac{1}{b}) \tag{11}$$

where $\mathcal{T}$ is the iteration. The convergence upper bound is

$$\mathbb{E}[g(u_{\mathcal{T}}) - g(u^*)] \leq O(\frac{1}{\sqrt{\mathcal{T}}}) \tag{12}$$

Loss function. Globally, the total loss function, incorporating both the temporal error term (TET) (Deng et al., 2022) and the MSE loss, given by

$$\mathcal{L} = (1 - \lambda)\frac{1}{T}\sum_{t=1}^T \mathcal{L}_{\text{CE}}(u_t, \mathbf{y}) + \lambda\mathcal{L}_{\text{MSE}}(u_t, \mathbf{y}) \tag{13}$$

where T denotes the number of time steps, and λ is a hyperparameter that governs the relative contribution of the temporal prediction error term and $\mathcal{L}_{\text{MSE}}$.

Table 1: Comparison with SOTA methods on MiniImageNet dataset for On-Device FSCIL.

Method	Acc. in each session (%) ↑									Avg.↑	Δ_{last}
	0	1	2	3	4	5	6	7	8		
CEC (SNN)	$54.33_{\pm2.14}$	$49.81_{\pm1.98}$	$46.82_{\pm1.82}$	$44.28_{\pm1.69}$	$42.28_{\pm1.63}$	$40.31_{\pm1.40}$	$38.26_{\pm1.36}$	$36.75_{\pm1.33}$	$35.78_{\pm1.29}$	$43.18_{\pm1.63}$	+12.92
FACT (SNN)	$63.03_{\pm0.91}$	$52.51_{\pm1.52}$	$49.25_{\pm1.28}$	$46.59_{\pm1.13}$	$44.32_{\pm1.19}$	$42.35_{\pm1.24}$	$40.16_{\pm1.17}$	$38.47_{\pm1.08}$	$37.44_{\pm1.17}$	$46.01_{\pm1.18}$	+11.26
S3C (SNN)	$31.55_{\pm0.51}$	$16.52_{\pm0.97}$	$17.89_{\pm0.84}$	$31.17_{\pm0.97}$	$24.69_{\pm0.82}$	$29.02_{\pm0.86}$	$15.35_{\pm0.62}$	$27.30_{\pm0.22}$	$26.56_{\pm0.04}$	$24.45_{\pm0.19}$	+22.14
BIDIST (SNN)	$43.72_{\pm0.32}$	$40.66_{\pm0.45}$	$37.81_{\pm0.69}$	$35.51_{\pm0.79}$	$33.58_{\pm0.61}$	$31.72_{\pm0.57}$	$30.12_{\pm0.64}$	$28.95_{\pm0.70}$	$27.85_{\pm0.49}$	$34.44_{\pm0.58}$	+20.85
SAVC (SNN)	$41.75_{\pm0.41}$	$38.54_{\pm0.30}$	$35.79_{\pm0.21}$	$33.40_{\pm0.13}$	$31.31_{\pm0.06}$	$29.47_{\pm1.00}$	$27.83_{\pm0.94}$	$26.37_{\pm0.89}$	$25.05_{\pm0.85}$	$32.17_{\pm1.08}$	+23.65
TEEN (SNN)	$62.87_{\pm1.69}$	$52.75_{\pm1.15}$	$49.64_{\pm1.01}$	$46.91_{\pm0.85}$	$44.82_{\pm0.85}$	$42.81_{\pm0.78}$	$40.59_{\pm0.77}$	$39.18_{\pm0.85}$	$38.01_{\pm0.84}$	$46.40_{\pm0.96}$	+10.69
WARP (SNN)	$50.07_{\pm4.16}$	$31.67_{\pm3.72}$	$29.93_{\pm3.50}$	$28.07_{\pm3.41}$	$25.49_{\pm4.23}$	$24.28_{\pm4.32}$	$22.84_{\pm4.34}$	$22.13_{\pm4.16}$	$21.30_{\pm3.96}$	$27.47_{\pm3.71}$	+27.40
CLOSER (SNN)	$65.88_{\pm0.09}$	$61.39_{\pm0.08}$	$57.72_{\pm0.03}$	$54.74_{\pm0.10}$	$52.34_{\pm0.07}$	$49.98_{\pm0.21}$	$47.69_{\pm0.18}$	$45.94_{\pm0.14}$	$44.69_{\pm0.07}$	$53.38_{\pm0.09}$	+4.01
ALADE (FSCIL)	$57.91_{\pm0.45}$	$47.04_{\pm0.62}$	$44.07_{\pm0.55}$	$41.75_{\pm0.45}$	$39.72_{\pm0.50}$	$37.92_{\pm0.47}$	$36.04_{\pm0.48}$	$34.57_{\pm0.45}$	$33.73_{\pm0.49}$	$41.42_{\pm0.47}$	+14.97
SAFA-SNN	$\mathbf{74.66}_{\pm0.62}$	$\mathbf{68.93}_{\pm0.46}$	$\mathbf{64.62}_{\pm0.57}$	$\mathbf{61.29}_{\pm0.88}$	$\mathbf{58.30}_{\pm0.95}$	$\mathbf{55.38}_{\pm0.78}$	$\mathbf{52.60}_{\pm0.62}$	$\mathbf{50.58}_{\pm0.60}$	$\mathbf{48.70}_{\pm0.67}$	$\mathbf{59.45}_{\pm0.67}$	

5.5 FAST ADAPTIVE PROTOTYPE SUBSPACE PROJECTION

Class prototypes averaging extracted features often causes discrepancies with actual data distributions (See Appendix A.4). As denoted in Section 3.2, we further define two specific prototypes that belong to $\mathcal{C}_B$ base classes and $\mathcal{C}_N$ new classes as $\tilde{\mathbf{B}} = \left[\frac{\mathbf{P}_1^b}{\|\mathbf{P}_1^b\|_2}, \frac{\mathbf{P}_2^b}{\|\mathbf{P}_2^b\|_2}, \ldots, \frac{\mathbf{P}_B^b}{\|\mathbf{P}_B^b\|_2} \right]^{\top} \in \mathbb{R}^{B \times D}$ and $\tilde{\mathbf{C}} = \left[\frac{\mathbf{P}_1^n}{\|\mathbf{P}_1^n\|_2}, \frac{\mathbf{P}_2^n}{\|\mathbf{P}_2^n\|_2}, \ldots, \frac{\mathbf{P}_N^n}{\|\mathbf{P}_N^n\|_2} \right]^{\top} \in \mathbb{R}^{N \times D}$, respectively, where D is the feature dimension, $B = |\mathcal{C}_B|$, and $N = |\mathcal{C}_N|$. To obtain information on the subspace structure spanned by the base classes, we construct a new prototype representation $\mathbf{G}$ to represent a generalized inverse of a covariance-like matrix in projection calculation, given by

$$\mathbf{G} = \tilde{\mathbf{B}}(\tilde{\mathbf{B}}^{\top}\tilde{\mathbf{B}})^{-1}\tilde{\mathbf{B}}^{\top} \tag{14}$$

Note that the normalization term $\tilde{\mathbf{B}}^{\top}\tilde{\mathbf{B}}$ is required because the bases $\{\mathbf{P}_i^b\}$ are not orthogonal to each other. Hence, we construct $|\mathcal{C}_B|$ distinct orthogonal subspaces represented by $\mathbf{G}$ for base classes. As classes increase, diverse class information allows extending classifiers with prototypes. The projection vectors $\tilde{\mathbf{P}}_{proj}$ are updated by mapping the coordinates in the base subspace back to the original D-dimensional space, denoted as $\tilde{\mathbf{P}}_{proj} = \tilde{\mathbf{C}}\mathbf{G}$. Finally, we integrate base and new knowledge by reconstructing the new information within the base subspace by

$$\tilde{\mathbf{P}} = (1 - \alpha)\tilde{\mathbf{C}} + \alpha\tilde{\mathbf{P}}_{proj} \tag{15}$$

where $\alpha \in (0, 1)$ is a trade-off weight factor for $\tilde{\mathbf{C}}$ and $\tilde{\mathbf{P}}_{\mathbf{proj}}$. The current classifier weights are updated by $\tilde{\mathbf{P}}$ to enable new class prediction (See Appendix A.5 for analysis).

6 EXPERIMENTS

6.1 EXPERIMENTS SETUP

Datasets and Spiking architecture. We evaluate the generalization performance of SAFA-SNN on two standard benchmark datasets, i.e., CIFAR100 (Krizhevsky et al., 2009) and Mini-ImageNet (Russakovsky et al., 2015), each split into eight 5-way 5-shot incremental tasks. We also extend experiments on three neuromorphic datasets: CIFAR-10-DVS (Li et al., 2017), DVS128gesture (Amir et al., 2017), and N-Caltech101 (Orchard et al., 2015). CIFAR-10-DVS is split into four 1-way 1-shot tasks,
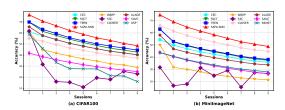

Figure 3: Accuracy in each session.

DVS128gesture into five 1-way 1-shot tasks, and N-Caltech101 into eight 5-way 5-shot incremental tasks. We adopt Spiking VGG variants (5, 9, 11), Spiking Resnet variants (18, 19, 20), and Spikingformer. Details can be found in Appendix A.6.

Realistic Implementation. All experiments are conducted on a mobile platform NVIDIA Jetson AGX Orin (NVIDIA, 2025), which features a 12-core Arm Cortex-A78AE processor, supporting 64-bit Armv8.2 architecture, with 60W peak power and idle power below 15W.

Training Details. Our implementation uses PyTorch with Adam optimizer. Models are trained for 300 epochs on CIFAR-100 and MiniImageNet, and for 100 epochs on CIFAR-10-DVS, DVS128Gesture, and N-Caltech101, all with a batch size of 128. The learning rate, β, θ, S, and δ are set to 0.001, 1.2, 0.001, 5, and 0.5, respectively. Each experiment is repeated three times with different seeds.

6.2 COMPARISON WITH STATE-OF-THE-ART METHODS

Baselines. We select nine FSCIL methods in their SNN version, i.e., CEC (Zhang et al., 2021), FACT (Zhou et al., 2022), S3C (Kalla & Biswas, 2022), BIDIST (Zhao et al., 2023), SAVC (Song et al., 2023), TEEN (Wang et al., 2023), WARP (Kim et al., 2023), CLOSER (Oh et al., 2024) and FSCIL-ASP (Liu et al., 2024a), as baselines. We adopt an SNN-based CIL method, ALADE-SNN (Ni et al., 2025), and two SNN training methods (Kim et al., 2022; Meng et al., 2023) to evaluate FSCIL performance. See Appendix A.7 for details.

Table 2: Comparative results on neuromorphic datasets.

Dataset	Method	Avg.(%)	Lst.(%)
CIFAR-10-DVS	WARP	$30.92_{\pm 0.70}$	$12.75_{\pm 0.14}$
	TEEN	$40.57_{\pm 1.02}$	$34.60_{\pm 0.79}$
	SAFA-SNN	$\mathbf{47.56}_{\pm 0.94}$	$\mathbf{36.96}_{\pm 0.38}$
DVS128gesture	WARP	$35.41_{\pm 1.76}$	$13.02_{\pm 1.25}$
	TEEN	$83.13_{\pm 0.80}$	$74.31_{\pm 1.42}$
	SAFA-SNN	$\mathbf{86.74}_{\pm 0.69}$	$\mathbf{77.91}_{\pm 0.63}$
N-Caltech101	WARP	$25.34_{\pm 1.63}$	$14.40_{\pm 2.07}$
	TEEN	$34.23_{\pm 0.18}$	$27.86_{\pm 0.49}$
	SAFA-SNN	$\mathbf{45.68}_{\pm 0.63}$	$\mathbf{39.69}_{\pm 0.67}$

Accuracy. We test the Top-1 accuracy on all seen tasks $0, 1, ...s$ after the session s. Δ_{last} means our relative improvement in the last session. Harmonic Accuracy (HAcc) measures the balance between base and novel class performance after each session s, i.e., $A_h = \frac{2 \times A_b \times A_n}{A_b + A_n}$, where A_b denotes test accuracy in base session, and A_n the average accuracy over sessions $s > 0$. As reported in Table 1, SAFA-SNN surpasses the second-best approach by 4.01% for improvement in the final session and boosts the average performance by 6.07% on Mini-ImageNet. The performance curves presented in Figure 3 show that SAFA-SNN achieves state-of-the-art performance across CIFAR100 and MiniImagenet, respectively. Table 2 shows accuracy on average (Avg.) and the last session (Lst.) compared with WARP and TEEN on three neuromorphic datasets, verifying its generalization and robustness performance. We find baseline WARP has the worst performance, indicating that the ability of space compaction to obtain effective parameter presentation is totally limited, far inferior to our spiking alignment and prototype adaptation. Table 3 shows that our method achieves accuracy of 31.06%, outperforming SNN training methods. This confirms the effectiveness and potential of our method on the SNN deployment. Additional results are provided in Appendix A.8.

Table 3: Comparative results with SNN-based training methods for on-device FSCIL.

Method	Dataset	Time step	Backbone	Param Size (M)	HAcc.(%)	Lst.(%)	Avg.(%)
Early Bird	CIFAR100	4	Spiking VGG5	221.35	$24.20_{\pm 0.54}$	$41.21_{\pm 0.66}$	$50.87_{\pm 0.60}$
(Kim et al., 2022)	CIFAR100	5	Spiking VGG9	106.87	$22.25_{\pm 0.85}$	$39.92_{\pm 1.07}$	$49.30_{\pm 1.49}$
SLTT	CIFAR100	6	Spiking VGG5	32.85	$25.17_{\pm 1.05}$	$41.63_{\pm 0.27}$	$51.12_{\pm 0.28}$
(Meng et al., 2023)	Mini-Imagenet	4	Spiking VGG9	106.87	$18.93_{\pm 0.31}$	$32.58_{\pm 0.77}$	$40.45_{\pm 0.64}$
	CIFAR-10-DVS	4	Spiking VGG9	262.87	$10.87_{\pm 0.62}$	$26.60_{\pm 1.27}$	$34.46_{\pm 0.85}$
SAFA-SNN	CIFAR100	4	Spiking VGG5	32.86	$\mathbf{27.32}_{\pm 0.91}$	$\mathbf{46.47}_{\pm 0.53}$	$\mathbf{56.72}_{\pm 0.25}$
	Mini-Imagenet	4	Spiking VGG9	22.63	$\mathbf{24.67}_{\pm 0.55}$	$\mathbf{49.25}_{\pm 0.24}$	$\mathbf{60.98}_{\pm 0.62}$
	CIFAR-10-DVS	4	Spiking VGG9	262.63	$\mathbf{14.47}_{\pm 0.86}$	$\mathbf{35.65}_{\pm 0.48}$	$\mathbf{41.67}_{\pm 0.34}$

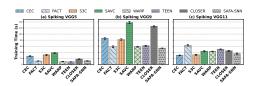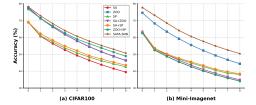Figure 4: Training Energy Cost.

Figure 5: Average Training Time.

Energy consumption. Following (Yao et al., 2023), we present the theoretical *inference energy* of different ANNs and SNNs (See Appendix for A.9 for formulations and results). The measurement of *Training Energy Consumption* uses the built-in sensor values (e.g., GPU, I/O) between the RAM and storage on the Jetson Orin AGX device (NVIDIA, 2025), which is gauged by multiplying power (W) by time. We put the training energy consumption in Figure 4. It can be seen that SAFA-SNN exhibits notably lower energy consumption than baselines.

The Run-Time Cost. Figure 5 compares the run-time cost of SAFA-SNN and baselines with Spiking VGG variant. Our neuronal dynamics, combined with ZOO and prototype subspace projection, reduce unnecessary training time without modifying synaptic weights or adjusting parameter spaces, achieving lower runtime and outperforming other baselines in both efficiency and effectiveness.

6.3 ABLATION STUDY

Ablation on SAFA-SNN. We conduct an ablation study to analyze the importance of each component in SAFA-SNN: Sparsity-Aware neuronal dynamics (SA), Zeroth-Order Optimization (ZOO), and subspace projection of prototypes (SP). We report incremental performance curves on CIFAR100 and MiniImageNet with time step 4 and Spiking VGG-9, as shown in Figure 6. We can infer that SA has the worst performance, since it does not consider possible adjustments in feature space, and we view it as the baseline. When equipped with SP, it shows significant performance gains as the model adapts to extract more informative features from base prototypes. We then use ZOO for gradient estimation, as shown in the highest curve, which corresponds to our full SAFA-SNN. Ablations verify that every component in SAFA-SNN boosts FSCIL performance.
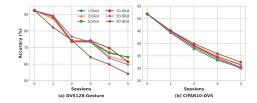

Figure 6: Ablation results of SAFA-SNN.　　Figure 7: All-session accuracy of variant shots

Effects on different N-way-K-shots. We assess shot count impact on accuracy by varying 1,2,5,10,15,20,50 on DVS128gesture and CIFAR-10-DVS in Figure 7 (Analysis in Appendix A.10).

Hyper-parameters Analysis. Results on the effects of key parameters (i.e., β, γ, S, δ, η and λ) and time step T, are provided in the Appendix A.11 and A.12, respectively.

Sparsity-Accuracy trade-off Analysis. The sparsity remain up to 80% even when setting $T = 2, 3, 4$ on different datasets through the training process, indicating a balance in spiking sparsity and accuracy, showing potential computation efficiency (Results in Appendix A.13).

7 DISCUSSION

A possible limitation is the performance degradation in deep networks, such as Spiking ResNet-34, which suggests that additional training epochs are required to maintain performance. In addition, assuming a fixed number of classes per session oversimplifies the dynamics of real-world data streams. Future work will address imbalanced way-shot settings to better reflect practical on-device FSCIL.

8 CONCLUSION

This paper focuses on few-shot class-incremental learning with SNN on-device scenarios. We proposed SAFA-SNN by incorporating sparsity-aware dynamics to preserve learned classes, zeroth-order optimization for non-differential spike, and subspace projection for incremental learning. Extensive experiments on benchmark and neuromorphic datasets indicate that SAFA-SNN outperforms existing FSCIL methods in both performance and energy efficiency on realistic implementation.

REFERENCES

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

Srinivas Anumasa, Bhaskar Mukhoty, Velibor Bojkovic, Giulia De Masi, Huan Xiong, and Bin Gu. Enhancing training of spiking neural network with stochastic latency. In *AAAI*, 2024.

Albert S Berahas, Liyuan Cao, Krzysztof Choromanski, and Katya Scheinberg. A theoretical and empirical comparison of gradient approximations in derivative-free optimization. *Foundations of Computational Mathematics*, 22(2):507–560, 2022.

Wei-Yu Chen, Yen-Cheng Liu, Zsolt Kira, Yu-Chiang Frank Wang, and Jia-Bin Huang. A closer look at few-shot classification. In *International Conference on Learning Representations*, 2019.

Yanqi Chen, Zhengyu Ma, Wei Fang, Xiawu Zheng, Zhaofei Yu, and Yonghong Tian. A unified framework for soft threshold pruning. *arXiv e-prints*, pp. arXiv–2302, 2023.

Shikuang Deng, Yuhang Li, Shanghang Zhang, and Shi Gu. Temporal efficient training of spiking neural network via gradient re-weighting. In *International Conference on Learning Representations*, 2022.

Jianchuan Ding, Bo Dong, Felix Heide, Yufei Ding, Yunduo Zhou, Baocai Yin, and Xin Yang. Biologically inspired dynamic thresholds for spiking neural networks. *Advances in neural information processing systems*, 35:6090–6103, 2022.

Yimeng Fan, Wei Zhang, Changsong Liu, Mingyang Li, and Wenrui Lu. Sfod: Spiking fusion object detector. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 17191–17200, 2024.

Li Fei-Fei, Rob Fergus, and Pietro Perona. Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories. *Computer Vision and Pattern Recognition Workshop*, 2004.

Wulfram Gerstner, Werner M Kistler, Richard Naud, and Liam Paninski. *Neuronal dynamics: From single neurons to networks and models of cognition*. Cambridge University Press, 2014.

Saeed Ghadimi, Guanghui Lan, and Hongchao Zhang. Mini-batch stochastic approximation methods for nonconvex stochastic composite optimization. *Mathematical Programming*, 155(1):267–305, 2016.

Chen Gong, Zhenzhe Zheng, Fan Wu, Xiaofeng Jia, and Guihai Chen. Delta: A cloud-assisted data enrichment framework for on-device continual learning. In *Proceedings of the 30th Annual International Conference on Mobile Computing and Networking*, pp. 1408–1423, 2024.

Zecheng Hao, Xinyu Shi, Yujia Liu, Zhaofei Yu, and Tiejun Huang. Lm-ht snn: Enhancing the performance of snn to ann counterpart through learnable multi-hierarchical threshold model. *Advances in Neural Information Processing Systems*, 37:101905–101927, 2024.

Elad Hazan et al. Introduction to online convex optimization. *Foundations and Trends® in Optimization*, 2(3-4):157–325, 2016.

Mark Horowitz. 1.1 computing's energy problem (and what we can do about it). In *2014 IEEE international solid-state circuits conference digest of technical papers (ISSCC)*, pp. 10–14. IEEE, 2014.

Dexuan Huo, Jilin Zhang, Xinyu Dai, Jian Zhang, Chunqi Qian, Kea-Tiong Tang, and Hong Chen. Anp-g: A 28-nm 1.04-pj/sop sub-mm 2 asynchronous hybrid neural network olfactory processor enabling few-shot class-incremental on-chip learning. *IEEE Journal of Solid-State Circuits*, 2025.

Sashank J Reddi, Suvrit Sra, Barnabas Poczos, and Alexander J Smola. Proximal stochastic methods for nonsmooth nonconvex finite-sum optimization. *Advances in neural information processing systems*, 29, 2016.

Jayateja Kalla and Soma Biswas. S3c: Self-supervised stochastic classifiers for few-shot class-incremental learning. In *European Conference on Computer Vision*, pp. 432–448. Springer, 2022.

Do-Yeon Kim, Dong-Jun Han, Jun Seo, and Jaekyun Moon. Warping the space: Weight space rotation for class-incremental few-shot learning. In *The Eleventh International Conference on Learning Representations*, 2023.

Youngeun Kim, Yuhang Li, Hyoungseob Park, Yeshwanth Venkatesha, Ruokai Yin, and Priyadarshini Panda. Exploring lottery ticket hypothesis in spiking neural networks. In *European Conference on Computer Vision*, pp. 102–120. Springer, 2022.

Alex Krizhevsky et al. Learning multiple layers of features from tiny images. *d*, 2009.

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

Xiao Li, Chenghua Lin, Ruizhe Li, Chaozheng Wang, and Frank Guerin. Latent space factorisation and manipulation via matrix subspace projection. In *International conference on machine learning*, pp. 5916–5926. PMLR, 2020.

Binghao Liu, Boyu Yang, Lingxi Xie, Ren Wang, Qi Tian, and Qixiang Ye. Learnable distribution calibration for few-shot class-incremental learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(10):12699–12706, 2023.

Chenxi Liu, Zhenyi Wang, Tianyi Xiong, Ruibo Chen, Yihan Wu, Junfeng Guo, and Heng Huang. Few-shot class incremental learning with attention-aware self-adaptive prompt. In *European Conference on Computer Vision*, pp. 1–18. Springer, 2024a.

Qianhui Liu, Jiaqi Yan, Malu Zhang, Gang Pan, and Haizhou Li. Lite-snn: Designing lightweight and efficient spiking neural network through spatial-temporal compressive network search and joint optimization. *arXiv preprint arXiv:2401.14652*, 2024b.

Sijia Liu, Pin-Yu Chen, Bhavya Kailkhura, Gaoyuan Zhang, Alfred O Hero III, and Pramod K Varshney. A primer on zeroth-order optimization in signal processing and machine learning: Principals, recent advances, and applications. *IEEE Signal Processing Magazine*, 37(5):43–54, 2020.

Changze Lv, Yansen Wang, Dongqi Han, Xiaoqing Zheng, Xuanjing Huang, and Dongsheng Li. Efficient and effective time-series forecasting with spiking neural networks. In *International Conference on Machine Learning*, pp. 33624–33637. PMLR, 2024.

Xinyue Ma, Suyeon Jeong, Minjia Zhang, Di Wang, Jonghyun Choi, and Myeongjae Jeon. Cost-effective on-device continual learning over memory hierarchy with miro. In *Proceedings of the 29th Annual International Conference on Mobile Computing and Networking*, pp. 1–15, 2023.

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

Qingyan Meng, Mingqing Xiao, Shen Yan, Yisen Wang, Zhouchen Lin, and Zhi-Quan Luo. Towards memory-and time-efficient backpropagation for training spiking neural networks. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 6166–6176, 2023.

Bhaskar Mukhoty, Velibor Bojkovic, William de Vazelhes, Xiaohan Zhao, Giulia De Masi, Huan Xiong, and Bin Gu. Direct training of snn using local zeroth order method. *Advances in Neural Information Processing Systems*, 36:18994–19014, 2023.

Wenyao Ni, Jiangrong Shen, Qi Xu, and Huajin Tang. ALADE-SNN: adaptive logit alignment in dynamically expandable spiking neural networks for class incremental learning. In Toby Walsh, Julie Shah, and Zico Kolter (eds.), *AAAI-25, Sponsored by the Association for the Advancement of Artificial Intelligence, February 25 - March 4, 2025, Philadelphia, PA, USA*, pp. 19712–19720. AAAI Press, 2025. doi: 10.1609/AAAI.V39I18.34171. URL https://doi.org/10.1609/aaai.v39i18.34171.

NVIDIA. Nvidia jetson orin. `https://www.nvidia.cn/autonomous-machines/embedded-systems/jetson-orin/`, 2025. Accessed: 2025-08-02.

Hyunseok Oh and Youngki Lee. Sign gradient descent-based neuronal dynamics: Ann-to-snn conversion beyond relu network. *arXiv preprint arXiv:2407.01645*, 2024.

Junghun Oh, Sungyong Baik, and Kyoung Mu Lee. Closer: Towards better representation learning for few-shot class-incremental learning. In *European Conference on Computer Vision*, pp. 18–35. Springer, 2024.

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

Keon-Hee Park, Kyungwoo Song, and Gyeong-Moon Park. Pre-trained vision and language transformers are few-shot incremental learners. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 23881–23890, 2024.

Xuerui Qiu, Malu Zhang, Jieyuan Zhang, Wenjie Wei, Honglin Cao, Junsheng Guo, Rui-Jie Zhu, Yimeng Shan, Yang Yang, and Haizhou Li. Quantized spike-driven transformer. *arXiv preprint arXiv:2501.13492*, 2025.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115:211–252, 2015.

Ali Safa. Continual learning with hebbian plasticity in sparse and predictive coding networks: a survey and perspective. *Neuromorphic Computing and Engineering*, 4(4):042001, 2024.

Guangyuan Shi, Jiaxin Chen, Wenlong Zhang, Li-Ming Zhan, and Xiao-Ming Wu. Overcoming catastrophic forgetting in incremental few-shot learning by finding flat minima. *Advances in neural information processing systems*, 34:6747–6761, 2021.

Christian Simon, Piotr Koniusz, Richard Nock, and Mehrtash Harandi. Adaptive subspaces for few-shot learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 4136–4145, 2020.

Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. *Advances in neural information processing systems*, 30, 2017.

Zeyin Song, Yifan Zhao, Yujun Shi, Peixi Peng, Li Yuan, and Yonghong Tian. Learning with fantasy: Semantic-aware virtual contrastive constraint for few-shot class-incremental learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 24183–24192, 2023.

Hongbo Sun, Jiahuan Zhou, Xiangteng He, Jinglin Xu, and Yuxin Peng. Finefmpl: fine-grained feature mining prompt learning for few-shot class incremental learning. In *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence*, pp. 1299–1307, 2024.

Xiaoyu Tao, Xiaopeng Hong, Xinyuan Chang, Songlin Dong, Xing Wei, and Yihong Gong. Few-shot class-incremental learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 12183–12192, 2020.

Qi-Wei Wang, Da-Wei Zhou, Yi-Kai Zhang, De-Chuan Zhan, and Han-Jia Ye. Few-shot class-incremental learning via training-free prototype calibration. *Advances in Neural Information Processing Systems*, 36:15060–15076, 2023.

Siqi Wang, Tee Hiang Cheng, and Meng-Hiot Lim. Ltmd: learning improvement of spiking neural networks with learnable thresholding neurons and moderate dropout. *Advances in Neural Information Processing Systems*, 35:28350–28362, 2022a.

Xuan Wang, Zhong Ji, Xiyao Liu, Yanwei Pang, and Jungong Han. On the approximation risk of few-shot class-incremental learning. In *European Conference on Computer Vision*, pp. 162–178. Springer, 2024.

Zifeng Wang, Zheng Zhan, Yifan Gong, Geng Yuan, Wei Niu, Tong Jian, Bin Ren, Stratis Ioannidis, Yanzhi Wang, and Jennifer Dy. Sparcl: Sparse continual learning on the edge. *Advances in Neural Information Processing Systems*, 35:20366–20380, 2022b.

Wenjie Wei, Malu Zhang, Hong Qu, Ammar Belatreche, Jian Zhang, and Hong Chen. Temporal-coded spiking neural networks with dynamic firing threshold: Learning with event-driven back-propagation. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 10552–10562, 2023.

Wenjie Wei, Malu Zhang, Zijian Zhou, Ammar Belatreche, Yimeng Shan, Yu Liang, Honglin Cao, Jieyuan Zhang, and Yang Yang. Qp-snn: Quantized and pruned spiking neural networks. *arXiv preprint arXiv:2502.05905*, 2025.

Davis Wertheimer and Bharath Hariharan. Few-shot learning with localization in realistic settings. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 6558–6567, 2019.

Yujie Wu, Lei Deng, Guoqi Li, Jun Zhu, and Luping Shi. Spatio-temporal backpropagation for training high-performance spiking neural networks. *Frontiers in neuroscience*, 12:331, 2018.

Man Yao, Guangshe Zhao, Hengyu Zhang, Yifan Hu, Lei Deng, Yonghong Tian, Bo Xu, and Guoqi Li. Attention spiking neural networks. *IEEE transactions on pattern analysis and machine intelligence*, 45(8):9393–9410, 2023.

Di Yu, Xin Du, Linshan Jiang, Shunwen Bai, Wentao Tong, and Shuiguang Deng. Fedlec: Effective federated learning algorithm with spiking neural networks under label skews. *arXiv e-prints*, pp. arXiv–2412, 2024a.

Di Yu, Xin Du, Linshan Jiang, Wentao Tong, and Shuiguang Deng. Ec-snn: splitting deep spiking neural networks for edge devices. In *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence*, pp. 5389–5397, 2024b.

Di Yu, Changze Lv, Xin Du, Linshan Jiang, Wentao Tong, Zhenyu Liao, Xiaoqing Zheng, and Shuiguang Deng. Ecc-snn: Cost-effective edge-cloud collaboration for spiking neural networks. *arXiv preprint arXiv:2505.20835*, 2025.

Chi Zhang, Nan Song, Guosheng Lin, Yun Zheng, Pan Pan, and Yinghui Xu. Few-shot incremental learning with continually evolved classifiers. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 12455–12464, 2021.

Han Zhang, Zhaokun Zhou, Liutao Yu, Liwei Huang, Xiaopeng Fan, Li Yuan, Zhengyu Ma, Huihui Zhou, Yonghong Tian, et al. Qkformer: Hierarchical spiking transformer using qk attention. *Advances in Neural Information Processing Systems*, 37:13074–13098, 2024.

Hanbin Zhao, Yongjian Fu, Mintong Kang, Qi Tian, Fei Wu, and Xi Li. Mgsvf: Multi-grained slow versus fast framework for few-shot class-incremental learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 46(3):1576–1588, 2021.

Linglan Zhao, Jing Lu, Yunlu Xu, Zhanzhan Cheng, Dashan Guo, Yi Niu, and Xiangzhong Fang. Few-shot class-incremental learning via class-aware bilateral distillation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 11838–11847, 2023.

Da-Wei Zhou, Fu-Yun Wang, Han-Jia Ye, Liang Ma, Shiliang Pu, and De-Chuan Zhan. Forward compatible few-shot class-incremental learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 9046–9056, 2022.

A APPENDIX

A.1 CONVERGENCE

The mean squared approximation error (MSE) of the gradient estimate of g as

$$\mathbb{E}\left[\left\|\hat{\nabla}g(x) - \nabla g(x)\right\|_2^2\right] = O(\delta^2) + O(b\delta^2 d^2) \tag{16}$$

where $\hat{\nabla}g(x)$ is the ture gradient obeys (Berahas et al., 2022), d is the dimensionality and b is the number of samples. At the non-differentiable point $u = 0$, the true gradient does not exist. However, the expected value of the estimation is zero when z is drawn from a symmetric distribution, so the bias is zero. The key is that the non-zero contribution of the estimation is concentrated around $u = 0$, where the true gradient is non-zero.

The perturbation radius (or smoothing factor) δ affects gradient estimation error. As it becomes smaller, the gradient estimate gets better. A smaller δ reduces the bias but drastically increases the variance. In practice, an overly small δ may cause the function difference to be masked by system noise, rendering it ineffective in representing the true differential. In contrast, a larger δ reduces variance but increases the bias. The choice of δ is critical to minimize the total error. As the number of samples b increases, this approximation converges to the true gradient, and as the sample size tends to infinity, it converges to the expected value $\mathbb{E}$ of the perturbation distribution.

At a differentiable point with $u \neq 0$, both the bias and variance of the single point estimation vanish, implying that the MSE of the multi-point average is also zero, reflecting a form of deterministic convergence. In contrast, when multi-point averaging is applied, the variance becomes $\frac{1}{4\delta^2 b}$. Thus, by increasing the number of samples b, we can effectively offset the variance explosion caused by letting $\delta \to 0$, thus ensuring a bounded and low variation gradient estimate.

A.2 ASSUMPTION

We provide four assumptions regarding convergence rates on zeroth-order algorithms which can be applicable to different types of problems according to the previous literature (Liu et al., 2020).

Assumption 1. (Convex optimization) The convergence error is measured by the optimality gap of function values:

$$\mathbb{E}[f(x_T) - f(x^*)] \tag{17}$$

for a convex objective f, where x_T denotes the updated point at the final iteration T, x^* denotes the optimal solution, and the expectation is taken over the full probability space.

Assumption 2. (Online convex optimization) The cumulative regret (Hazan et al., 2016) is typically used in place of the optimality gap for an online convex cost function f_t as

$$\mathbb{E}[\sum_{t=1}^{T} f_t(x_t) - \min_x \sum_{t=1}^{T} f_t(x)] \tag{18}$$

Assumption 3. (Unconstrained nonconvex optimization) The convergence is evaluated by the first-order stationary condition in terms of the squared gradient norm for the nonconvex objective f:

$$\frac{1}{T}\sum_{t=1}^{T} \mathbb{E}\left[\|\nabla f(x_t)\|_2^2\right] \tag{19}$$

Assumption 4. (Constrained nonconvex optimization) The criterion for convergence is commonly determined by detecting a sufficiently small squared norm of the gradient mapping (Ghadimi et al., 2016; J Reddi et al., 2016):

$$P_X(x_t, \nabla f(x_t), \eta_t) := \frac{1}{\eta_t}\left[x_t - \Pi_X\left(x_t - \eta_t \nabla f(x_t)\right)\right] \tag{20}$$

where $P_X(x_t, \nabla f(x_t), \eta_t)$ can naturally be interpreted as the projected gradient, which offers a feasible update from the previous point x_t.

A.3   PROOF OF ZEROTH-ORDER OPTIMIZATION

We provide a theoretical proof of the estimation function $g(t)$ according to literature (Mukhoty et al., 2023) as follows.

Definition 1. We say that a function $g : \mathbb{R} \to \mathbb{R}_{\geq 0}$ is a surrogate function (gradient surrogate) if it is even, non-decreasing on the interval $(-\infty, 0)$, and $c := \int_{-\infty}^{\infty} g(z) \, dz < \infty$.

Assume that

$$\int_0^{\infty} z^{\alpha+1} \lambda(z) \, dz < \infty. \tag{21}$$

Then, $\mathbb{E}_{z \sim \lambda}[g^2(u; z, \delta)]$ is a surrogate function.

Theorem 1. *Let p be a distribution and $p(t)$ its corresponding probability density function(PDF). Assume that the integrals $\int_0^{\infty} t^{\alpha} p(t) dt$ and $\int_0^{\infty} t^{\alpha+1} p(t) dt$ exist and are finite. Let further $\tilde{\lambda}$ be the distribution with corresponding PDF function*

$$\tilde{\lambda}(z) = \frac{1}{c} |z| \int_{-\infty}^{\infty} t^{\alpha} \lambda(t) \, dt, \tag{22}$$

where c is the scaling constant such that $\int_{-\infty}^{\infty} \tilde{\lambda}(z) dz = 1$. Then,

$$\mathbb{E}_{z \sim p}[g^2(u; z, \delta)] = \frac{d}{du} \mathbb{E}_{z \sim \tilde{p}}[\mathrm{ch}(u + \delta z)]. \tag{23}$$

The probability density function characterizing the standard normal distribution $\mathcal{N}(0, 1)$ takes the form $\frac{1}{\sqrt{2\pi}} \exp\left(-\frac{z^2}{2}\right)$. Consequently, it is straightforward to obtain

$$\mathbb{E}_{z \sim p}[g^2(u; z, \delta)] = \int_{-\infty}^{\infty} \frac{|z|}{2\delta} \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{z^2}{2}\right) dz = \frac{1}{\delta\sqrt{2\pi}} \exp\left(-\frac{u^2}{2\delta^2}\right) \tag{24}$$

Theorem 3. *Let $g(u)$ be a surrogate function. Suppose further that $c = \int_{-\infty}^{\infty} g(z) dz < \infty$ and define $p(z) = \frac{z^{\alpha} g'(z\delta)}{\int_0^{\infty} z^{\alpha} g'(z\delta) \, dz}$ (so that $\lambda(z)$ is a PDF). Then,*

$$c \, \mathbb{E}_{z \sim p}[g^2(u; z, \delta)] = \mathbb{E}_{z \sim p}[cG^2(u; z, \delta)] = g(u)$$

Consider the Sigmoid surrogate function, where the differentiable Sigmoid function approximates the Heaviside. The corresponding surrogate gradient is given by

$$\frac{d}{du}\left(\frac{1}{1 + \exp(-ku)}\right) = \frac{k \exp(-ku)}{(1 + \exp(-ku))^2} =: g(u). \tag{25}$$

Observe that $g(t)$ satisfies our definition of a surrogate (i.e., $g(t)$ is even, non-decreasing on $(-\infty, 0)$, and $\int_{-\infty}^{\infty} g(t) \, du = 1 < \infty$). Thus, we have $c = -2\delta^2 \int_0^{\infty} g'(t\delta) t dt$, where $a := \sqrt{\frac{1}{0.4262}}$. The corresponding PDF is given by

$$p(z) = -\frac{\delta^2}{c} \cdot \frac{g'(\delta t)}{z} = \frac{a^2 \exp(-k\delta z)(1 - \exp(-k\delta z))}{z(1 + \exp(-k\delta z))^3}. \tag{26}$$

Table 4: Detailed prediction results of MBR and MAR (%) between soft calibration-based method TEEN and SAFA on CIFAR100.

Method	CBN↑	MBN↓	MNN↓	CNN↑	MBR↓	MAR↓	Seen↑	Unseen↑
TEEN	649778	4222	12108	5092	0.704	**0.005**	62.35	14.95
SAFA-SNN	**650935**	**3065**	**10180**	**7020**	**0.592**	0.006	**73.57**	**15.43**

We define "Misclassified to Base classes Ratio" (MBR) for new classes and "Misclassified to most similar New classes Ratio" (MNR) for base classes. As shown in Table 4, MBR is much higher than the MNR, indicating that new classes are misclassified as base than base classes as new, and poor base prototypes rarely occurs. Correspondingly, the "Correctly classified to Base class Number and to New class number" as CBN and CNN, otherwise, the "Mistakenly classified to Base class Number and to New class number" as MBN and MNN. We also report the accuracy on seen classes and unseen clsses.

Table 5: Performance Comparison across Sessions (HMean / NAcc).

Session	1	2	3	4	5	6	7	8
TEEN	36.05 / 25.00	29.07 / 18.80	26.40 / 16.67	25.53 / 16.00	24.10 / 14.92	24.74 / 15.43	24.57 / 15.31	24.06 / 14.95
SAFA-SNN	41.90 / 28.80	34.51 / 22.30	29.25 / 18.13	27.13 / 16.55	27.74 / 17.04	27.30 / 16.73	26.53 / 16.17	27.14 / 16.68
Δ	+5.85 / +3.80	+5.44 / +3.50	+2.85 / +1.46	+1.60 / +0.55	+3.64 / +2.12	+2.56 / +1.30	+1.96 / +0.86	+3.08 / +1.73

Table 6: Comparison with FSCIL baselines on static dataset CIFAR100 with 5way-5shot incremental learning setting and Spiking VGG-9.

Method	Acc. in each session (%) ↑									Avg.↑	Δ_{last}
	0	1	2	3	4	5	6	7	8		
CEC	60.82	57.49	53.93	50.80	48.18	45.75	43.69	41.95	40.22	49.20	+7.87
FACT	69.82	61.46	57.83	54.33	51.35	48.64	46.20	44.45	42.61	52.97	+5.48
S3C	61.72	31.60	16.00	15.40	11.00	19.40	18.00	25.20	23.00	24.59	+25.09
BIDIST	61.30	57.69	54.17	50.88	48.33	45.72	43.82	42.08	40.36	49.37	+7.73
SAVC	41.75	38.54	35.79	33.40	31.31	29.47	27.83	26.37	25.05	32.17	+23.04
TEEN	69.87	63.20	59.49	55.67	52.79	50.11	48.12	46.33	44.51	59.34	+3.58
WARP	61.32	44.88	41.84	39.11	36.68	34.60	32.20	31.07	28.74	38.94	+19.35
CLOSER	55.22	52.55	49.39	46.41	44.23	42.06	40.23	38.67	37.21	45.10	+10.88
ASP	58.55	46.05	33.29	27.76	27.52	27.48	21.37	21.06	16.04	31.01	+32.05
SAFA-SNN	**76.03**	**70.91**	**66.1**	**61.55**	**58.19**	**55.22**	**52.77**	**50.33**	**48.09**	**56.59**	

Table 7: Comparison with FSCIL baselines on static dataset CIFAR100 with 5way-5shot incremental learning setting and Spiking VGG-11.

Method	Acc. in each session (%) ↑									Avg.↑	Δ_{last}
	0	1	2	3	4	5	6	7	8		
CEC	64.37	59.85	55.89	52.04	48.88	46.37	43.92	41.72	39.60	50.63	+7.30
FACT	61.98	57.48	53.54	50.05	47.06	44.48	42.13	39.92	38.02	48.62	+8.88
S3C	45.28	24.80	14.40	16.20	13.60	15.80	11.00	22.40	18.00	20.28	+28.90
SAVC	62.95	58.11	53.96	50.36	47.21	44.44	41.97	39.76	37.77	48.39	+9.13
TEEN	64.30	59.79	55.73	51.91	48.86	46.14	43.68	41.42	39.34	50.79	+7.56
WARP	62.28	57.95	53.99	50.41	47.25	44.69	42.34	40.20	38.28	48.16	+8.62
CLOSER	67.87	63.52	59.57	56.15	52.83	49.89	47.54	45.43	43.38	54.02	+3.52
SAFA-SNN	**76.95**	**71.20**	**66.00**	**61.81**	**58.14**	**54.74**	**51.94**	**49.16**	**46.90**	**58.12**	

A.5 SUBSPACE PROJECTION

Prototypical networks are a few-shot learning framework designed to recognize novel classes using only a small number of labeled examples. The model uses a shared feature extractor to embed both labeled support images into a common feature space. For each class, the embeddings of its support samples are averaged to form a prototype vector that represents the class in the feature space. Classification is performed by comparing embeddings to these prototypes using a distance metric, typically Euclidean distance. Predictions are made on embeddings based on L2 proximity to each class prototype (Wertheimer & Hariharan, 2019). Since classification in prototypical networks is non-parametric, training focuses on optimizing the feature extractor. The model is trained using episodic tasks consisting of small support and query sets, enabling it to produce discriminative prototypes from limited examples. Constructing complex classifiers $f(\cdot)$ is challenging under such constraints. In SAFA-SNN, we employ a simple yet effective prototype update strategy, Subspace Projection, which isolates attribute-relevant information without relying on auxiliary networks or weighted loss terms (Li et al., 2020).

Table 8: Comparison with FSCIL baselines on static dataset CIFAR100 with 5way-5shot incremental learning setting and Spiking Resnet20.

Method	Acc. in each session (%) ↑									Avg.↑	Δ_{last}
	0	1	2	3	4	5	6	7	8		
CEC	12.95	12.22	11.27	10.55	10.04	9.48	8.93	8.53	8.15	10.43	+38.75
FACT	64.03	56.51	53.11	49.77	47.31	44.51	42.53	40.86	39.09	47.13	+7.81
S3C	49.15	26.40	22.80	22.20	22.80	22.00	25.20	33.40	30.40	27.38	+16.5
TEEN	65.68	57.79	54.64	51.07	48.30	45.53	43.69	42.00	40.25	49.21	+6.65
WARP	46.85	36.85	34.39	31.56	29.64	27.88	26.40	25.25	23.93	31.64	+22.97
CLOSER	56.02	53.43	50.33	47.21	44.65	42.48	40.44	38.86	37.10	45.61	+9.80
SAFA-SNN	**76.95**	**71.20**	**66.00**	**61.81**	**58.14**	**54.74**	**51.94**	**49.16**	**46.90**	**58.12**	

Table 9: Comparison with FSCIL baselines on static dataset CIFAR100 with 5way-5shot incremental learning setting and Spiking Resnet19.

Method	Acc. in each session (%) ↑									Avg.↑	Δ_{last}
	0	1	2	3	4	5	6	7	8		
FACT	67.67	62.17	58.03	54.41	51.23	48.64	46.17	44.14	42.09	53.28	+4.17
TEEN	66.71	61.71	57.32	53.69	50.35	47.67	45.13	43.14	41.03	52.46	+5.23
WARP	66.58	6.12	2.94	3.47	2.46	1.91	1.87	1.92	1.30	10.83	+44.96
SAFA-SNN	**73.98**	**68.74**	**63.99**	**59.99**	**56.38**	**53.49**	**51.03**	**48.60**	**46.26**	**56.35**	

A.6 DATASETS AND SPIKING ARCHITECTURE DETAILS

Neuromorphic datasets exhibit sparse features, typically obtained from event-based simulators or converted frame-based datasets. CIFAR10-DVS (Li et al., 2017) is a neuromorphic dataset derived from the original CIFAR10, where visual inputs are recorded by a Dynamic Vision Sensor (DVS), capturing changes in pixel intensity as asynchronous events instead of static frames. It contains 9,000 training samples and 1,000 test samples. DVS128-Gesture (Amir et al., 2017) is a gesture recognition dataset consisting of 11 hand gesture categories collected from 29 individuals under three different lighting conditions. N-Caltech101 (Orchard et al., 2015) comprises 8,246 event streams, each 300 milliseconds in duration, recorded by an event-based camera as it captured dynamic visual inputs of Caltech101 (Fei-Fei et al., 2004) images displayed on an LCD screen. These recordings span 101 object categories, preserving the diversity of the original dataset while incorporating temporal event-based representations. We provide our neuromorphic dataset splitting method in the codes.

We modify three types of ANNs (VGG, Resnet, and Transformers) to their SNN counterparts with no floating-point multiplication and division, aiming to offer a guideline for proper SNN model selection for FSCIL. In addition, we configure the Spikingformer with a depth of 2, five tokenizer convolutional blocks, and a spike-form dimensionality of 128.

We directly encode spikes with layers of LIF neurons. Each formulated unit S generating spikes can be represented as

$$S = \mathcal{SN}((\mathrm{BN}(\mathrm{CONV}(\boldsymbol{X})))), \tag{27}$$

where $\boldsymbol{X} \in \mathbb{R}^{T \times B \times C \times H \times W}$ is the input with time, $BN(\cdot)$ is the batch normalization layer and $\mathcal{SN}(\cdot)$ is the LIF neuron model.

A.7 BASELINES DESCRIPTION

We establish several baselines to better evaluate our proposed framework. To this end, we explore nine of the existing SOTA from the FSCIL literature, CEC (Zhang et al., 2021), FACT (Zhou et al., 2022), S3C (Kalla & Biswas, 2022), BIDIST (Zhao et al., 2023), SAVC (Song et al., 2023), TEEN (Wang et al., 2023), WARP (Kim et al., 2023), CLOSER (Oh et al., 2024) and ASP (Liu et al., 2024a). CEC, BIDIST, and S3C establish dynamically evolving architectures to effectively support incremental learning. CEC incrementally transforms newly added linear classifiers into a graph-based structure. BIDIST assigns a learnable weight W_t to each task and employs bilateral distillation between the representations of current and previous tasks. S3C expands its stochastic classifiers by progressively incorporating four angular representations. FACT, SAVC, TEEN,

Table 10: Comparison with FSCIL baselines on static dataset CIFAR100 with 5way-5shot incremental learning setting and Spikingformer.

Method	Acc. in each session (%) ↑									Avg.↑	Δ_{last}
	0	1	2	3	4	5	6	7	8		
TEEN	**68.37**	**52.42**	48.67	45.43	42.59	40.08	37.86	35.86	34.07	45.37	+2.48
WARP	63.30	6.45	4.04	4.20	2.48	3.34	2.36	2.35	2.59	10.38	+33.96
CLOSER	23.53	21.86	20.24	18.87	17.95	16.92	15.97	15.20	14.47	18.89	+22.08
SAFA-SNN	62.65	52.09	**48.90**	**46.01**	**43.65**	**41.34**	**39.70**	**38.25**	**36.55**	**45.68**	

Table 11: Comparison with FSCIL baselines on static dataset MiniImageNet with 5way-5shot incremental learning setting and Spiking VGG-5.

Method	Acc. in each session (%) ↑									Avg.↑	Δ_{last}
	0	1	2	3	4	5	6	7	8		
CEC	57.28	52.93	**49.98**	47.12	44.90	42.76	40.29	38.77	37.57	45.73	+1.06
FACT	54.51	43.91	41.16	39.27	37.68	35.80	33.95	32.64	31.82	38.97	+6.81
S3C	41.01	12.56	17.43	29.85	24.01	25.87	21.32	30.95	28.50	25.28	+10.13
TEEN	55.32	44.89	42.23	41.12	39.19	37.49	35.59	34.34	33.34	40.72	+5.29
WARP	50.24	25.84	24.13	22.34	21.00	19.97	18.75	18.38	17.66	24.70	+20.97
CLOSER	48.07	44.54	41.89	40.16	38.28	36.41	34.54	33.39	32.67	38.66	+5.96
SAFA-SNN	**61.31**	**53.06**	49.92	**48.12**	**46.19**	**43.88**	**41.40**	**39.86**	**38.63**	**46.93**	

and CLOSER represent prototype tuning approaches that employ a prototype-based classifier $g_P(\cdot)$ instead of the conventional MLP classifier $g_\phi(\cdot)$. FACT facilitates forward compatibility by synthesizing virtual prototypes during the base training stage, thereby enabling more flexible adaptation to future tasks. Meanwhile, SAVC advances base task performance through leveraging semantic-aware fantasy classes combined with contrastive learning, which enhances feature representation. TEEN presents a training-free prototype rectification technique, which effectively improves classification robustness without requiring additional training. Lastly, CLOSER strikes an effective balance between transferability and discriminability by employing a mechanism of compressed feature spreading, thereby improving generalization across tasks.WARP trains backbone and classifier on base tasks, then fine-tunes classifier and subnetworks of θ with multi-axis rotations for distinct representations. ASP efficiently fine-tunes small prompts on a frozen backbone, leveraging task-aware and task-invariant prompts to improve selection during testing.

A.8 MORE COMPARATIVE RESULTS

HMean Accuracy. Table 5 reports the novel class accuracy and harmonic mean comparison against the runner-up method on CIFAR100 with Spiking VGG-9, further emphasizing the consistent performance improvements and the effectiveness of SAFA-SNN.

Comparison of all-session Accuracy. Comparisons with SOTA methods on CIFAR100 using Spiking VGG-9, VGG-11, Spiking ResNet-20, Spiking ResNet-19, and Spikingformer are presented in Table 6, 7, 8, 9, and 10 and on Mini-ImageNet with Spiking VGG-5, Spiking Resnet-18 are listed in Table 11 and 12. These results demonstrate that SAFA-SNN consistently achieves the highest final-session performance, highlighting its robustness and effectiveness.

A.9 ENERGY CONSUMPTION

According to previous studies (Horowitz, 2014; Yao et al., 2023; Lv et al., 2024), for SNNs, the theoretical energy consumption of layer l can be calculated as:

$$E(l) = E_{AC} \times SOPs(l) \tag{28}$$

where SOPs is the number of spike-based accumulate (AC) operations.

For traditional artificial neural networks (ANNs), the theoretical energy consumption required by layer b can be estimated by:

$$E(b) = E_{MAC} \times FLOPs(b) \tag{29}$$

Table 12: Comparison with FSCIL baselines on static dataset MiniImageNet with 5way-5shot incremental learning setting and Spiking RESNET18.

| Method | Acc. in each session (%) ↑ | | | | | | | | | Avg.↑ | Δ_{last} |
	0	1	2	3	4	5	6	7	8		
CEC	55.23	51.07	47.62	45.15	42.84	40.64	38.42	36.64	35.03	43.63	+13.06
TEEN	54.53	47.59	44.54	42.19	40.14	38.03	36.06	34.53	33.11	41.41	+14.98
CLOSER	56.93	52.77	49.16	46.72	44.31	42.12	40.04	38.47	37.17	45.97	+10.92
SAFA-SNN	**76.03**	**70.91**	**66.1**	**61.55**	**58.19**	**55.22**	**52.77**	**50.33**	**48.09**	**56.59**	

Table 13: Inference Energy Consumption Comparison of ANNs and SNNs.

Model	**Energy (J)**	**Model**	**Energy (J)**
Spiking VGG5	7916.53	VGG5	10115.57
Spiking VGG9	3958.41	VGG9	5057.97
Spiking VGG11	494.92	VGG11	632.39
Spiking VGG13	495.00	VGG13	5689.97
Spiking VGG16	495.08	VGG16	5689.97

where FLOPs denotes the number of floating-point multiply-and-accumulate (MAC) operations. We assume that both MAC and AC operations are implemented on 45nm hardware (Yao et al., 2023), where $E_{MAC} = 4.6$pJ and $E_{AC} = 0.9$pJ. Note that $1 \text{ J} = 10^3 \text{ mJ} = 10^{12} \text{ pJ}$.

The number of synaptic operations at the layer l of an SNN is estimated as:

$$SOPs(l) = T \times \zeta \times FLOPs(l) \tag{30}$$

where T is the number of time steps in the simulation, ζ is the firing rate of the input spike train of layer l.

Based on these theoretical analysis, we report the theoretical inference energy consumption results with variable structures in Table 13, which shows that the energy consumption of ANNs is obviously higher than SNNs, the larger the model is, the more significant the differences between them, which further hardens the ability of ANNs to implement on edge devices.

A.10   EFFECT ON N-WAY K-SHOT

SAFA-SNN leverages N-way K-shot datasets to estimate prototypes for novel classes. To evaluate the impact of the number of shots on accuracy, we fix the incremental way and vary shots 1,2,5,10,15,20,50 on DVS128gesture and CIFAR-10-DVS. We can infer that with more instances per class, the estimation of prototypes will be more precise, and the performance will correspondingly improve.

A.11   HYPER-PARAMETER SENSITIVITY

We report the last-session accuracy on CIFAR100 varying the β in {0.4,0.9,1.2} and γ in {0.01, 0.04,0.07}. We also change the sampled point number S from {5,10,15} and constant δ from {0.1,0.3,0.5}, resulting compared results in Figure 8. Our method is robust to the choices of hyper-parameters, and it achieves the best performance when $\delta = 0.5$ and $S = 5$, $\gamma = 0.01$ and $\beta = 1.2$. The adaptive ratio η varys in $\{0.1, 0.3, 0.5, 0.7, 0.9\}$ and λ in $\{0.01, 0.05, 0.1\}$ (See Table 14).

A.12   EFFECTS ON DIFFERENT TIME STEPS

Upon proving the importance of SAFA-SNN modules, we further evaluate the impacts of the introduced time steps in SNN. We choose time steps T among 1,2,4,6,8 and 10. We report the performance in incremental sessions on CIFAR100 and Mini-ImageNet in Figure 9, respectively. As illustrated in the figures, the best performance is observed when T is approximately between 4 and 6. Since the larger time step often corresponds to challenges like gradient vanishing or exploding (Yu et al., 2024b), we suggest $T = 4$ as the default in our experiment.
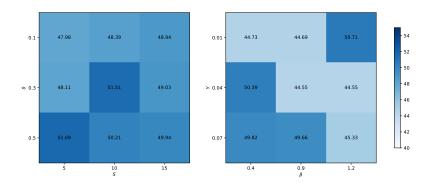
Figure 8: Comparison of different hyper-parameters on δ, S, γ and β for on-device FSCIL.

Table 14: Comparison on various η and λ.

λ	η	Acc. in each session (%) ↑									Avg.↑
		0	1	2	3	4	5	6	7	8	
0.05	0.1	76.12	70.06	65.31	60.76	57.50	54.65	51.99	48.96	45.77	59.01
0.05	0.3	76.12	70.51	65.44	60.83	57.81	54.72	52.30	49.44	46.63	59.31
0.05	0.5	**77.93**	**72.69**	**68.16**	**64.09**	**60.49**	**57.35**	**54.80**	**52.20**	**49.88**	**61.96**
0.05	0.7	76.12	70.57	65.89	61.71	58.09	55.74	52.33	49.85	47.60	59.77
0.05	0.9	75.18	69.45	64.97	60.84	57.14	54.20	51.74	49.23	47.21	58.88
0.01	0.5	76.73	70.88	66.41	61.73	59.04	55.67	53.24	50.40	45.40	59.95
0.1	0.5	76.67	70.59	66.11	61.43	58.51	55.02	52.13	49.52	45.40	59.49

A.13 ACCURACY AND SPARSITY TRADE-OFF

We trained the CIFAR100 dataset on the original Spiking VGG network with 285,448 neurons in all layers with dynamic threshold. We found that the training is very robust to even extreme values of adaptive ratio, as shown in Figure 10 and 11. The points feature red dot with black border is the start point in each setting while the points feature green dot with black border is the end point. We can see that both the sparsity and accuracy convergence at high level, showing the promise in sparsity-accuracy trade-off in SAFA-SNN.

A.14 VISUALIZATION OF FIRING RATE

We visualize the average firing rates in each time step of non-adaptive (dashed lines) neurons and adaptive neurons (solid lines) on CIFAR100 dataset in Figure 12, where "S-k-A" and "S-k-N" are the firing rete of adaptive neurons in the k session and non-adaptive neurons in the $k-th$ session, respectively. It is observed that non-adaptive neurons exhibit higher firing rates than adaptive neurons in the same session. This implies that adaptive neurons constrain their activity to remain within threshold limits, potentially reducing their responsiveness to novel classes, whereas non-adaptive neurons sustain stable firing behavior to mitigate the risk of catastrophic forgetting.

A.15 NEURONAL DYNAMICS IN SNN

We provide Discrete temporal dependency representation in SNNs following (Oh & Lee, 2024).The continuous neuronal dynamics of a general one-dimensional integrate-and-fire neuron is a linear differential equation with a thresholding criterion (Gerstner et al., 2014):

$$\tau_m \frac{du}{dt} = -f(u(t)) + RI(t) \tag{31}$$

$$\lim_{\delta \to 0^+} u(t+\delta) = u_{rest} \quad \text{(Integration)} \tag{32}$$

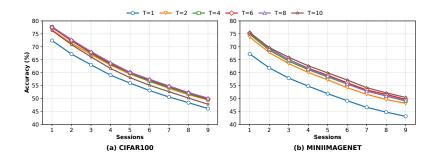$$u(t) \geq \theta_{th} \quad \text{(Thresholding)} \tag{33}$$

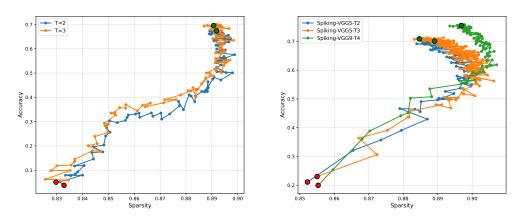Figure 9: Accuracy in each session on different time steps.

Figure 10: Sparsity-Accuracy on Mini-Imagenet.   Figure 11: Sparsity-Accuracy on CIFAR100.

where time $t \in \mathbb{R}^+$, dynamics function $f(u) : \mathbb{R} \rightarrow \mathbb{R}$, input current $I(t)$ at time t, membrane potential u, resting potential u_{rest}, threshold θ_{th}, membrane resistance R, membrane capacitance C, and membrane constant $\tau_m = RC$.

Integrate-and-fire models are simplified phenomenological models of biological neuronal dynamics (Gerstner et al., 2014). They consist of two components: (i) a time-evolution of membrane potential (Integration) and (ii) a firing mechanism to create a spike (Thresholding). Its continuous neuronal dynamics is a differential equation with a thresholding criterion. For computational tractability, the general one-dimensional integrate-and-fire model is discretized as follows:

$$u_{pre(t)} = u(t-1) + f\, u(t-1) + \frac{R}{\tau_m} I(t) \tag{34}$$

$$s(t) = H(u_{pre}(t) - \theta_{th}) \tag{35}$$

where time $t \in \mathbb{N}$, dynamics function $f(u) : \mathbb{R} \rightarrow \mathbb{R}$, pre-firing potential $u_{pre(t)}$, $s(t) \in \{0, 1\}$ a spike, post-firing potential $u(t)$, Heaviside step function H, influx current $I(t)$, threshold θ_{th}, membrane resistance R, and membrane constant τ_m. The potential $u_{pre(t)}$ resets to u after the spike $s(t)$ fires based on its pre-defined reset mechanism:

$$u(t) = u_{pre(t)} - \theta_{th} s(t) \quad \text{(reset-by-subtraction)} \tag{36}$$

$$u(t) = u_{pre(t)}(1 - s(t)) \quad \text{(reset-to-zero)} \tag{37}$$

Integrate-and-fire (IF) neuron has the simplest neuronal dynamics defined as $f(u) = 0$, $\tau_m = 1$ in equation 34:
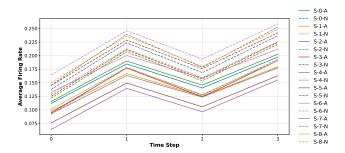
Figure 12: Firing rate of adaptive and non-adaptive neurons.

$$u_{pre(t)} = u(t-1) + RI(t) \tag{38}$$

Leaky-Integrate-and-Fire (LIF) neuron introduces linear leakage $f(u) = -(u - u_{rest})/\tau_m$ into the dynamics:

$$u_{pre(t)} = u(t-1) - \frac{u(t-1) - u_{rest}}{\tau_m} + \frac{R}{\tau_m}I(t) \tag{39}$$

A.16 DISCUSSION

Our method significantly improves the recognition accuracy of new classes in on-device FSCIL scenarios, where catastrophic forgetting and resource constraints severely limit model performance. By addressing the underperformance of new classes, our approach offers new insights into the overlooked challenges of class imbalance and forgetting. To the best of our knowledge, this is the first study in the on-device FSCIL literature to systematically analyze the performance degradation of new classes, highlighting the need to prioritize their evaluation in future on-device FSCIL research. In addition, our method may also have offer potential heuristic effects on initiating training with sparse models for implementing edge intelligence on neuromorphic hardware.