EC-SNN: Splitting Deep Spiking Neural Networks for Edge Devices

Di Yu[1] , **Xin Du**[1] * , **Linshan Jiang**[2] , **Wentao Tong**[1] and **Shuiguang Deng**[1]

[1]Zhejiang University

[2]National University of Singapore

yudi2023@zju.edu.cn, jsjduxin@gmail.com, linshan@nus.edu.sg,
{toldzera, dengsg}@zju.edu.cn

Abstract

Deep Spiking Neural Networks (SNNs), as an advanced form of SNNs characterized by their multilayered structure, have recently achieved significant breakthroughs in performance across various domains. The biological plausibility and energy efficiency of SNNs naturally align with the requisites of edge computing (EC) scenarios, thereby prompting increased interest among researchers to explore the migration of these deep SNN models onto edge devices such as sensors and smartphones. However, the progress of migration work has been notably challenging due to the influence of the substantial increase in model parameters and the demanding computational requirements in practical applications. In this work, we propose a deep SNN splitting framework named EC-SNN to run the intricate SNN models on edge devices. We first partition the full SNN models into smaller sub-models to allocate their model parameters on multiple edge devices. Then, we provide a channel-wise pruning method to reduce the size of each sub-model, thereby further reducing the computational load. We design extensive experiments on six datasets (i.e., four non-neuromorphic and two neuromorphic datasets) to substantiate that our approach can significantly diminish the inference execution latency on edge devices and reduce the overall energy consumption per deployed device with an average reduction of 60.7% and 27.7% respectively while keeping the effectiveness of the accuracy.

1 Introduction

The rapid advancement in machine learning technologies has precipitated an escalated demand for computational resources. Traditional computational architectures, characterized by the dichotomy between processing and memory functions, often render the implementation of large-scale AI systems financially prohibitive for many organizations. In this milieu, the development of neuromorphic chips like Loihi and TrueNorth heralds the potential of third-generation neural networks (i.e., spiking neural networks) to replace traditional artificial neural networks (ANNs) gradually. These SNN models are distinguished by their energy efficiency and biological fidelity, positioning them as a promising avenue for addressing computational and energy constraints.

Recent larger and deeper SNNs have shown breakthroughs in performance improvements across diverse domains [Li *et al.*, 2023; Yao *et al.*, 2024; Su *et al.*, 2023], enabling the possibility of implementing more extensive functionalities. Nevertheless, These more complex SNN-related models result in the loss of their advantage for widespread application in resource-constrained edge devices, such as smartphones and sensors. The deployment of these models remains constrained by the limited computational power and storage capability of edge devices. To tackle this challenge, we propose a novel framework named *EC-SNN*[1], which can utilize the collaboration of multiple edge devices to offer an energy-efficient strategy for deploying complex SNN models on the edge devices.

In the EC-SNN, we mainly adopt and integrate two approaches to solve this issue. The first type of the approach is partitioning the original large deep SNN model into several smaller sub-models, also known as *split learning* (SL) [Thapa *et al.*, 2022], deploying them across multiple edge devices, and orchestrating their cooperation to execute the same task collectively. This method involves segmenting the complete machine learning (ML) model into several smaller network segments, which further undergo individual training on either a cloud server or distributed clients and leverage localized datasets. The main focus of SL is to alleviate the computational burden and processing requirements. However, none of them ever considered splitting SNN on edge devices. In EC-SNN, we propose to allocate the spitted SNN sub-models on the edge devices to reduce energy consumption.

The other way involves simplifying the computational mechanisms within the initial SNN model. The previous pruning methods for SNNs [Kim *et al.*, 2022; Chen *et al.*, 2022a] mainly involved the direct application of some well-established pruning techniques from ANNs [Frankle and Carbin, 2018; Hoefler *et al.*, 2021]. These transferred methods have demonstrated their effectiveness through numerous

*Corresponding Author: Xin Du.

[1] Code is available at https://github.com/AmazingDD/EC-SNN

experiments in the domain of SNN pruning. In this study, we employ a channel-wise pruning method [Chen *et al.*, 2023] to conduct the initial pruning phase, dramatically reducing the sub-model's execution time without significant impact on performance. By the integration of the two approaches, EC-SNN is a novel splitting framework, which minimizes the execution latency and maximizes the utilization of all available edge devices, while adhering to the individual energy constraints of each device simultaneously.

Our main contributions are summarized as follows.

- This is the first study that focuses on the splitting problem of deep SNNs. The resolution of this issue offers a viable solution for deploying deep SNNs across multiple edge devices through parallel operation. It not only effectively utilizes the capabilities of deep SNNs but also maintains the advantage of low energy consumption of the model. Furthermore, solving this problem allows for the full utilization of multiple edge devices in practical implementations.
- To solve the splitting problem of deep SNNs, we propose a novel framework named EC-SNN to run the intricate SNN models on edge devices. The EC-SNN first decomposes a deep SNN model into smaller sub-models according to its characteristics. Then, we utilize a channel-wise pruning method to reduce the size of each sub-model. By making a trade-off of model size and accuracy on each edge device, we effectively reduce the energy consumption, computational load, and inference latency of the deep SNN model.
- We conduct extensive experiments with four non-neuromorphic datasets and two neuromorphic datasets within two sensing modalities to demonstrate that our framework markedly decreases the inference execution latency on edge devices and reduces the overall energy consumption with negligible loss of model accuracy in various SNN applications.

The remainder of this paper is organized as follows. Section 2 reviews related works. Section 3 describes the design of EC-SNN. Section 4 presents the evaluation results. Section 5 concludes this paper.

2 Related Work

Spiking Neural Networks on Edge Computing. Most existing works on SNNs primarily focus on their notable energy efficiency, a characteristic that renders them particularly suitable for applications on edge computing (EC). EC, a paradigm that facilitates expedited data streaming via nodes proximal to the data source, enables real-time, low-latency data processing. Meanwhile, an increasing number of SNNs have demonstrated exceptional computational performance across a wide array of edge computing applications, including facial recognition [Barchid *et al.*, 2023], robotics [Jiang *et al.*, 2023], and object detection [Su *et al.*, 2023]. However, the limitation on resource-constraint edge devices poses a significant challenge in utilizing the power of the SNNs, especially in the training and inference phases in SNN.

There exist several approaches to address the concern of the limitation of the resources on SNN. The key component to address this concern involves devising a computation-efficient framework aimed at reducing unnecessary computations and optimizing weight parameters in SNNs. [Liu *et al.*, 2024] utilizes the sparsity property of SNN to perform weight parameter pruning by Lottery Ticket Hypothesis (LTH) to discover a very sparse subnet that maintains similar performance. [Nguyen *et al.*, 2021] introduces an innovative connection-reducing approach to pruning the SNN with the TTFS coding during the on-chip STDP-based learning. [Chowdhury *et al.*, 2021] incorporates a PCA-based spatial pruning method with temporal pruning to reduce total time steps and enhance inference efficiency. Differently, we employ a filter-wise pruning technique tailored for deep SNNs, aiming to adjust the model size and reduce energy consumption to enhance the compatibility with diverse resource-constraint limitations of various edge devices.

Split Learning. We propose another potential direction to address the limitation of the resource-constrained edge devices incorporated with SNN. A promising direction to address the deployment issues of deep CNN models on the edge devices involves decomposing the model and deploying it in a distributed manner across multiple edge devices, which is also known as Split Learning (SL) [Gupta and Raskar, 2018]. In SL, it performs part of the model's training on servers to tackle the full-offloading problem among edge clients, while minimizing the delays in a resource-constrained network and preventing sensitive information leakage through intermediate information communication. Based on the way to split a huge top-down network structure, SL can be classified into two categories, which are *horizontal* and *vertical* SL. Vertical SL scales down the scope of the problem addressed by the full model, diffusing specific local data to sub-models which will then be trained on cloud servers. [Kim *et al.*, 2017] proposes a solution model to split the deep network into a tree of sub-networks by clustering the classes into groups. [Chen *et al.*, 2023] splits the large DNN into multiple lightweight class-specific models to satisfy the sensor's memory and energy constraints. [Hou *et al.*, 2022] employs a deep reinforcement learning method to determine optimal partitioning decisions for CNN, intended for deployment across heterogeneous edge devices for collaborative inference. In contrast, Horizontal SL aims at finding the optimal points to split the whole model, and train each part on a cloud server. [Bakhtiarnia *et al.*, 2023] introduces a dynamic split computing method based on the state of the communication channel. [Kim *et al.*, 2020] divides a single deep learning architecture into a common extractor, a cloud model, and a local classifier for distributed learning. [Chen *et al.*, 2021a] proposes a loss-based asynchronous training and search-based quantization method for the best combination of exponent bits and bias.

However, their work solely considers splitting the CNN on edge devices. To the best of our knowledge, we are the pioneering contributors to applying SL on the SNN. Besides, unlike the previous works, in this paper, we adopt a hybrid SL strategy on deep SNNs, in which the complete large model is partitioned into multiple feature extractors responsible for

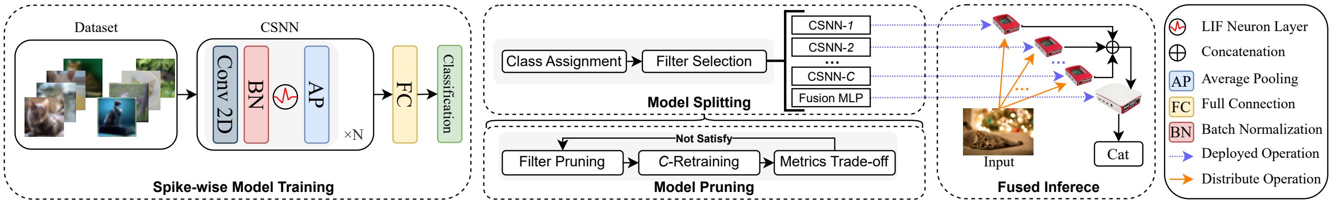

Figure 1: The overview of EC-SNN.

class-specific tasks and a fusion module that integrates the final inference.

3 Methodology

3.1 Design Overview

To tackle a classification task with multiple classes, large DNN models, especially CNN models [Sengupta *et al.*, 2019], perform well on the application. Analogously, deep SNN models rely on intricate network structures and extensive parameter training to enhance their efficacy. However, such original SNN models are hard to deploy, especially on a resource-constrained edge device. To address this issue, we propose to utilize the characteristics of SNNs and the collaboration of multiple edge devices. In detail, the key idea of our method is to split the original SNN model into several class-specific sub-models that meet resource-constrained limitations and sufficiently take advantage of multiple available devices.

We consider a classification system that involves N edge devices for sensing and an aggregation server for the classification results. Figure 1 overviews the workflow of our proposed method: **E**dge **C**ollaboration with **S**piking **N**eural **N**etwork (EC-SNN), which mainly contains four components: *spike-wise model training*, *model splitting*, *model pruning*, and *fused inference*. In the spike-wise model training, we train and convert the deep SNNs from the conventional CNN models. In the model splitting, we split a deep SNN model into sub-models which contain a subset of classes. To reduce the computation overhead, we prune the sub-models in the model pruning. Lastly, we assign the sub-models on the corresponding edge devices while the aggregation server fuses the extracted features to obtain the final inference results. The details of each component are as follows.

3.2 Spike-wise Model training

Diverging from the conventional CNN models, convolutional Spiking Neural Network (CSNN) employs spiking neurons in place of ReLU activation layers while retrained with equal intensity [Deng and Gu, 2020]. In CSNNs, all bias components are eliminated and an average strategy is highly recommended for pooling operations to mimic the behavior of neural cells which integrates all information perceived to release spikes and enhance bio-fidelity.

In this study, we use *Leaky Integrate-and-Fire (LIF)* model [Gerstner and Kistler, 2002], one of the most promising spiking neuron models, to build CSNNs due to its low computation cost and bio-plausibility. Rather than directly passing the sum of weighted inputs to an activation function like artificial neurons, LIF neurons integrate the input over time with a leakage. When the integration exceeds a threshold, the neurons emit a spike [Eshraghian *et al.*, 2023], namely a discrete event. These neurons abstract the profile of spikes among layers, hence the transmitted information is stored in the frequency of spikes instead of the spikes themselves, also known as temporal information concentration [Kim *et al.*, 2023]. This distinct information transmission compared with traditional artificial neurons leads to the energy-efficiency merit of CSNN.

Inspired by [Chen *et al.*, 2022b], we describe the layer of LIF neurons with a series of discrete-time equations:

$$V[t^-] = V[t-1] + \frac{1}{\tau}(X[t] - (V[t-1] - V_{reset})) \tag{1}$$

$$S[t] = H(V[t^-] - V_{threshold}) \tag{2}$$

$$V[t] = V[t^-](1 - S[t]) + V_{reset}S[t] \tag{3}$$

where τ, $V_{threshold}$, and V_{reset} represent the membrane time constant, the firing threshold, and the reset potential respectively. At time step t, $X[t]$ is the current input features; $V[t^-]$ and $V[t]$ are the membrane potential after neuronal dynamic integration and after the trigger of firing separately. $H(x)$ is the Heaviside step function:

$$H(x) = \begin{cases} 1, & \text{if } V[t^-] \geq V_{threshold} \\ 0, & \text{otherwise} \end{cases} \tag{4}$$

to simulate the firing process, which generates a spike $S[t]$ with a binary value.

Unlike the training process in ANNs, direct gradient-based training in SNNs is impossible because the Heaviside function $H(\cdot)$ in Equation (3) is not differentiable. Surrogate functions [Zenke and Vogels, 2021], which are differentiable alternatives of $H(\cdot)$, is one method to tackle this issue. Specifically, we choose the arc tangent spiking function proposed by [Fang *et al.*, 2023]

$$H^*(x) = \frac{1}{\pi}arctan(\pi x) + \frac{1}{2} \tag{5}$$

as the surrogate function for training.

Moreover, the overall effectiveness of SNNs is contingent upon the time step t, as depicted in Equation (1). Analogous to recurrent networks, a large time step often corresponds to challenges such as gradient vanishing or exploding phenomena during the backpropagation process. As t escalates, it also leads to more computational operations and more energy consumption. Thus, selecting an appropriate t holds paramount importance for our forthcoming experiments to compare diverse metrics.

Algorithm 1 Model Splitting in EC-SNN

Input: The number of devices N; initial pruning threshold ζ_0; step size for pruning threshold $\Delta\zeta$; available memory size M_i and remaining energy limitation E_i for device i; training dataset (X, y)
Parameter: the classes set C; trained original CSNN$_0$
Output: class-specific sub-models {CSNN$_1$,..., CSNN$_N$} and a fusion MLP model M

```
Let $\zeta = \zeta_0$, $\gamma = True$, $I^-$={device 1, ..., device $N$}.
Let $E = \{E_1, ..., E_N\}$, $M = \{M_1, ..., M_N\}$.
repeat
    Let $C = \{C_1, C_2, ..., C_N\}, s.t.|C| = \sum_{i=1}^{N} |C_i|$.
until $\forall C_a, C_b \in C, \big||C_a| - |C_b|\big| \leq 1$.
while $\gamma$ is $True$ do
    for $i$ in $N$ do
        CSNN$_i$ $= prune(\zeta, C_i,$ CSNN$_0, X, y)$
        $i^* = greedySearchAssign(E, M,$ CSNN$_i, I^-)$.
        $I^- = I^- \backslash \{i^*\}$, $E = E \backslash \{E_{i^*}\}$, $M = M \backslash \{M_{i^*}\}$.
    end for
    if $I^- = \phi$ then
        $\gamma = False$.
    else
        $\zeta = \zeta - \Delta\zeta$ .
    end if
end while
$out_f$ = $concat$(CSNN$_1(X)$,..., CSNN$_N(X)$).
$M$ = $train(out_f, y)$.
return $M$, CSNN$_1$, ..., CSNN$_N$
```

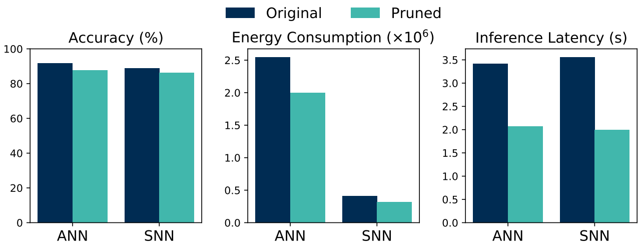

Figure 2: The performance of ANN-based and SNN-based CifarNet before and after pruned to the equivalent model size with identical methods

Algorithm 2 Model Pruning in EC-SNN

Input: pruning threshold ζ; assigned classes subset C_i
Parameter: the set of filters in trained raw CSNN F; training dataset (X, y)
Output: pruned CSNN$_i$

```
$X_i, y_i$ = $resample(X, y, C_i)$.
$F_i$ = $filterPrune(F, \zeta, X_i)$.
CSNN$_i$ = $reconstruct(F_i$, weight of $F$).
CSNN$_i$ = $retrain$(CSNN$_i$, $X_i$, $y_i$).
return CSNN$_i$
```

3.3 Model Splitting

In conventional CNN, each filter has its exclusive impact when learning and inferring certain classes [Wang *et al.*, 2020], offering insights to split the original CSNN model into several class-specific CSNN sub-models. Each sub-model is curated to encompass solely the essential filters pertinent to its responsible classes. As illustrated in Algorithm 1, each CSNN sub-model undergoes pruning based on a threshold ζ and its corresponding categories, following the relatively equitable workload distribution. Subsequently, a greedy search mechanism is initiated to identify the most fitting edge device for deploying a particular sub-model, considering energy and memory constraints. In cases where no suitable device is found, an iterative approach is employed to fine-tune the threshold and repeat the allocation process, until all sub-models succeed to be assigned on the edge devices.

3.4 Model Pruning

A conventional deep CNN model usually contains four parts, i.e., convolution, activation, pooling, and fully connected (FC) layers, among which the convolution is the most computation-intensive. Pruning the filters contributes to accelerating inference execution speed and lessening overheads for storing the parameters in conventional CNN. Considering the similarities between CNNs and CSNNs, we believe that pruning the filters to build CSNN sub-models can also significantly reduce the computation overhead. For example, typical results of CifarNet [Chen *et al.*, 2021b] with 6 same convolutional filters are exemplified in Figure 2. Thus, we can reduce the energy consumption and shrink sub-models with a tolerable compromise on test accuracy.

In detail, inspired by [Chen *et al.*, 2023], we utilize a ranking metric called the average percentage of zero (APOZ) [Hu *et al.*, 2016] to evaluate the importance of each filter on learning and inference. Concretely, each convolution layer needs a LIF layer to generate an activation map for each inside filter. The activation map may comprise some zero spikes that convey less useful feature information. Consequently, we apply the trained CSNN model to infer sensing data that consists of N samples in a specific bundle of classes C. Then, the APOZ of filter f in the convolution layer l is denoted as:

$$APOZ_{f,C}^{l} = \frac{\sum_{n=1}^{N} \sum_{p=1}^{hw} \psi(A_{f,C}^{l}(n,p))}{Nhw} \tag{6}$$

where w and h represent the width and the height of the activation map and $A_{f,C}^{l}(n,p)$ denote the activated value at point p in the activation map generated by filter f in the convolution layer l with the input sample n when recognizing classes set C. Note that$\psi(x) = 1$ if x is 0 and $\psi(x) = 0$, otherwise. The pruning process is shown in Algorithm 2. The filter with a larger APOZ value is less important for the specific-class bundle C, so that we define a threshold ζ_{APOZ} to screen the filter importance. Any filter with an APOZ value higher than ζ_{APOZ} is considered to be pruned. Additionally, a reduced value of ζ_{APOZ} results in the diminished size and energy cost of the class-specific sub-model, albeit at the cost of increased accuracy loss induced by filter pruning. The threshold ζ_{APOZ} assumes a crucial role in achieving a balance between model performance and resource-constrained limitations.

3.5 Fused Inference

In the result fusion phase, each sub-model on the edge devices perceives inputs and extracts corresponding features. The server aggregates the generated features through concatenation and feeds them to an MLP to obtain the final prediction result. Note that the MLP for result fusion requires training once all trained sub-models are provided. Given the pre-trained weights derived from sub-models, extensive training epochs for MLP are deemed unnecessary.

4 Experiments

4.1 Experimental Settings

Considering the versatile applicability of the framework, we choose 4 non-neuromorphic datasets (*CIFAR*, *Caltech*, *GTZAN*, and *UrbanSound*) and 2 neuromorphic datasets (*CIFARDVS*, *NCaltech*) to construct the classification task in our experiments. Moreover, we use VGG-structured architectures [Simonyan and Zisserman, 2014] with different depths (5, 9, and 16 relatively) to classify these public datasets. For simplicity, we exemplify EC-SNN-VGG9 to denote our proposed EC-SNN approach with VGG9 structure.

All models are implemented based on Pytorch and SpikingJelly[2]. We set the time step to 5 and the total number of training epochs to 70 for all networks. During the training process, we utilize Adam optimizer with a cosine-decay learning rate initiated to $1e$-4 and set the batch size to 16. For LIF neurons, the time constant τ is set to 1.33. Each trial is carried out on one NVIDIA GeForce RTX 4090 GPU and 9 Raspberry Pi-4B as the edge devices for evaluating the execution time of a specific sub-model processing one sample.

4.2 Performance Evaluation

Test Accuracy. In the work, the paramount concern is not primarily centered around accuracy. For the metric, we aim to keep the effectiveness of the accuracy. Compared with ANN-structured and SNN-structured baselines across various scenarios on one edge device, EC-SNN exhibits an accuracy deviation ranging from 1% to 3%, which can be deemed as acceptable. Based on the outcomes, it is evident that as the network depth increases, the ANN-structured models typically achieve slightly superior accuracy performance compared to the SNN-structured ones by the involvement of considerable computational operations. Nonetheless, it is impressive that under specific circumstances, SNN models exhibit the surprising capability to slightly surpass the accuracy of ANN models. These results substantiate the feasibility of converting from ANN to SNN without concerns of excessive compromise in accuracy.

When considering the EC scenario, Table 1 demonstrates the consistency of accuracy for EC-SNN-VGG9 as the number of edge devices increases. In most cases, the accuracy fluctuation for the final fusion prediction remains within a variance of less than one percentage point. The involvement of more sub-models illuminates the feasibility of deploying larger-scale models without concerns about high accuracy loss.

Dataset	**The Number of Devices**				***Avg.***	***Std.***
	3	5	7	9		
CIFAR	85.25	85.28	85.11	84.89	85.13	0.18
Caltech	97.92	97.77	97.02	97.62	97.58	0.40
CIFARDVS	64.65	62.10	61.70	62.55	62.75	1.31
NCaltech	92.48	91.74	91.00	91.45	91.67	0.62
GTZAN	87.24	87.30	87.68	86.49	87.18	0.50
UrbanSound	89.41	89.01	89.52	89.58	89.38	0.26

Table 1: Accuracy (%) performance of EC-SNN-VGG9 across six datasets in different edge resources

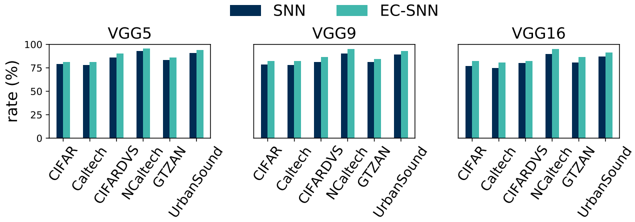

Figure 3: The energy savings rate achieved by SNN-structure and EC-SNN-structure compared with ANN-structure when APOZ=95.

Energy Consumption. A spike is related to some AC operations and how much is associated with the architecture design of the network, so spike counts of SNNs are positively correlated with energy cost [Yao *et al.*, 2023]. The lower the spike counts, the higher the energy efficiency is. We define a metric called average total spiking activity number(ATSAN) to compute the spike counts, which can be quantified as

$$ATSAN = \frac{1}{N}\sum_{n=1}^{N}\sum_{t=1}^{T}\sum_{l=1}^{L} S_l^t(n) \tag{7}$$

where N is the number of inputs, L represents the total number of convolution layers for a specific network, and T is the time step. $S_l^t(n)$ denotes the output spikes number of convolution layer l at time step t when making inference task of input n. The metric is comparable on ANN architecture since [Rueckauer *et al.*, 2017] shows that the LIF neuron in SNN is an unbiased estimator of the ReLU activation function over time. After the whole network architecture and time step are determined, T and L are fixed constants, and the spike counts are only related to S. We mainly focus on ATSAN to depict energy consumption and delve into how SL and LIF nodes bring energy savings for the original neural networks.

Figure 3 presents the energy-saving tendencies exhibited by SNN-related architectures in contrast to ANN structures across six datasets, calculated by the ratio of total saving $ATSAN$ from SNN-related structure to that from ANN structure. Notably, SNN exhibits a minimum energy saving of approximately 75% compared to the original ANN architecture. Furthermore, our proposed EC-SNN approach with one device demonstrates additional energy efficiencies ranging from 2% to 4% beyond these established savings. This enhancement stems from the fact that the EC-SNN approach selectively retains the critical convolution layers, thereby transmitting reduced information to the full-connection layers.

Figure 4 illustrates the variation in $ATSAN$ of different

[2] https://github.com/fangwei123456/spikingjelly

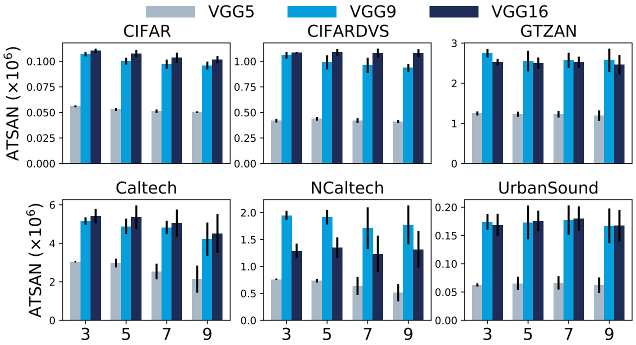

Figure 4: Mean ATSAN performance of EC-SNN per device across six datasets as the number of edge devices increases when APOZ=95.

EC-SNN structures with an increasing number of deployed edge devices across six datasets. The mean energy consumption per individual edge device appears relatively consistent. Additionally, a gradual decline is observed in this metric across results associated with static image datasets. When compared with the sparsity of event-based datasets and the complexity of the spectrogram, static image datasets are comparatively simpler, requiring less energy for the SNN model to process due to the reduced target classification scale. Nevertheless, the incorporation of extra devices results in a notable increase in the fluctuation of energy consumption, possibly due to the diversity of classification tasks allocated across different edge devices.

Model Size. Given the constrained storage capacity of edge devices, it is imperative to assess both the original SNN model size and the corresponding pruned sub-models to ascertain their suitability for further deployment. Meanwhile, variations in the depth of the VGG structure and the diversity of input feature sizes significantly impact the resultant model size. In this paper, we introduce a metric termed Shrink Ratio (SR), which quantifies the minimum rate at which EC-SNN can reduce the model size:

$$SR = 1 - \frac{max(Size(\text{EC-SNN}_i))}{Size(\text{origin SNN})} \tag{8}$$

$Size(\text{EC-SNN}_i)$ denotes the memory size of EC-SNN deployed on edge device i. Figure 5 illustrates a significant reduction in the sizes of all initial SNN models when implemented across various scenarios, providing compelling evidence supporting the efficacy of our current pruning methodology. In most cases, the SR values are larger than 95%, and an increasing trend in pruning efficiency is observed as the model complexity rises. This demonstrates that many complex model designs and computational operations are redundant for problem-solving.

We also conduct a comparative analysis of size variations among three VGG-structured sub-models across six datasets while varying the number of edge devices involved in experiments. This approach aims to validate the stability of our method as intuitively illustrated in Figure 6. In most scenarios, we observe that the sizes of the sub-models allocated to each edge device do not exhibit notable fluctuations despite the increase in the number of devices, which is advantageous for us as it enables unified device management and mitigates the necessity to acquire numerous edge devices with varying specifications. Certainly, it is noteworthy that the size deviation in the UrbanSound dataset is notably higher than other datasets, particularly evident in the VGG5 cases. After a thorough investigation, we discovered this phenomenon is attributed to a specific class labeled as $Siren$ in the Urbansound dataset. The maximum size of any sub-model designated task within this category can reach up to 51 MB, while most other sub-models only require about 11 MB total model size. Moreover, this phenomenon is observed in several other datasets, although it is not as pronounced as in the Urbansound dataset. One explanation is that the models require increased structure complexity and computational capacity to discern certain categories better. Simultaneously, this phenomenon gradually diminishes with increased model depth, which aligns with our experimental findings.

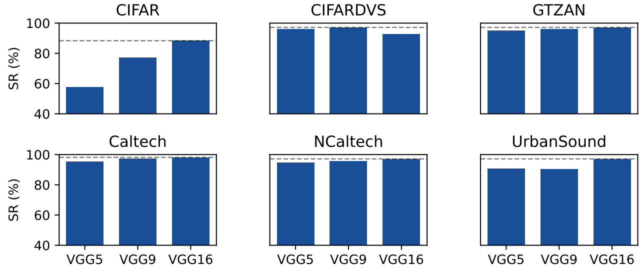

Figure 5: SR of SNN models with different structures across six datasets after pruned by EC-SNN when APOZ=95, where the dashed line represents the maximum SR.

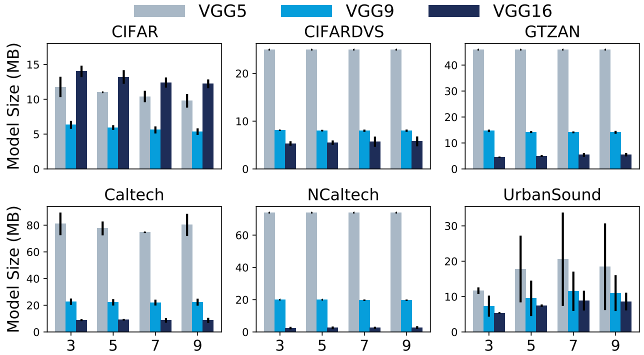

Figure 6: The changing patterns of model size after pruning as more devices are introduced in different cases when APOZ=95.

Inference Latency. Figure 7 introduces the latency patterns of each VGG structure when inferring one sample for different applications by one edge device. It can be intuitively concluded that: (1) ANN models, which have the same architecture as SNN, generally demonstrate faster inference times with an average of about 0.2 seconds due to their reduced complexity in activation parts. (2) The latency length is jointly determined by the model's complexity and the input frame size. For instance, the size of the SNN-VGG16 model for Caltech is roughly 4× more than that of processing CIFAR. (3) Leveraging EC and SL yields significant acceleration, with an average improvement of approximately 60.7%

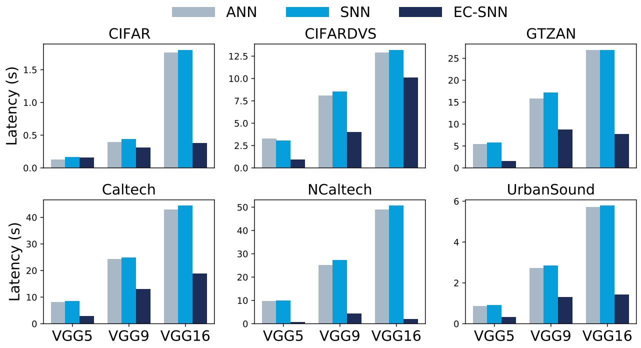

Figure 7: The patterns of inference latency among ANN-/SNN-/EC-SNN-VGG model across different datasets when APOZ=95

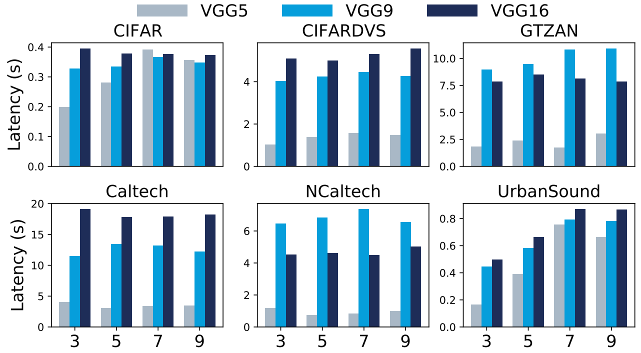

Figure 8: The patterns of inference latency in different scenarios as the number of devices changes when APOZ=95.

in inference latency.

Figure 8 depicts the latency distribution patterns observed across varying quantities of devices engaged in inference processes within the same varieties of scenarios. The latency exhibits a similar trend across varying numbers of devices, showing a slight downward trajectory. However, more edge devices involved do not always lead to latency reduction, as the situations with 7 and 9 devices illustrated. This phenomenon could potentially be attributed to the high computational demand arising from the expanding input dimensions within the fusion model, which is a result of the concatenation tensors derived from the output of these devices.

4.3 APOZ Sensitivity Analysis

Based on the evaluation above, it is observed that the performance of EC-SNN remains consistent across each dataset. We opt to utilize CIFAR10 as the reference dataset to assess the performance changes in the four metrics concerning different APOZ values. Due to the sparsity in inter-layer activations of the SNN, we need a smoother approach to select crucial filters through APOZ threshold screening. Besides, we set a minimum reserved filter count, such as 16, to prevent a scenario where a layer's filters are almost entirely removed.

Figure 9 shows the details of our APOZ sensitivity experiments. As the APOZ value decreases, there is a proportional decrease in energy consumption, model size, and latency, and the reduction rate is linked to the complexity of the model. During the initial stages of APOZ value reduction, ranging from 94 to 100, a significant reduction in model size and latency does not necessarily imply a proportional decrease in energy consumption and accuracy, which indicates that EC-SNN is capable of retaining fundamental computational aspects while upholding accuracy levels. However, as the APOZ continues to decrease, specifically within the range of 90-94, these four metrics will simultaneously decrease at a certain rate. This is because EC-SNN eliminates certain computation operations crucial for prediction accuracy due to excessively low APOZ values. When the APOZ value is lower than 90, our proposed smooth threshold mechanism comes into effect, and the pruned model ensures that each layer retains a minimum number of filters to maintain normal computational flow. Overall, a high APOZ value like 95 is adequate for EC-SNN to achieve relatively satisfactory results, showcasing its high sensitivity.

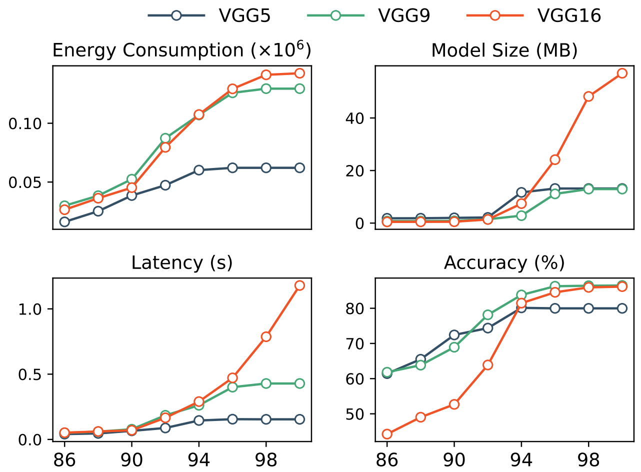

Figure 9: The changing patterns of energy consumption, model size, accuracy, and latency as the APOZ value varies in CIFAR10

5 Conclusions

In this study, a novel model-splitting framework named EC-SNN is specifically designed to solve the splitting problem of deep SNNs. The resolution of the problem offers a viable solution for deploying deep SNNs across multiple edge devices. The EC-SNN decomposes a deep SNN model into smaller sub-models and utilizes a channel-wise pruning method to streamline the complex network architecture and enables more efficient inferences by significantly reducing energy consumption and latency. Extensive experiments are constructed to evaluate the proposed framework based on two sensing modalities, three network architectures, and six datasets. The results demonstrated that the EC-SNN can significantly diminish the inference execution latency on edge devices and reduce the overall energy consumption while maintaining the inference accuracy.

Acknowledgments

The work of this paper is supported by the National Key Research and Development Program of China under Grant 2022YFB4500100, the National Natural Science Foundation of China under Grant 62125206, and the Key Research Project of Zhejiang Province under Grant 2022C01145.

References

[Bakhtiarnia *et al.*, 2023] Arian Bakhtiarnia, Nemanja Milošević, Qi Zhang, Dragana Bajović, and Alexandros Iosifidis. Dynamic split computing for efficient deep edge intelligence. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 1–5. IEEE, 2023.

[Barchid *et al.*, 2023] Sami Barchid, Benjamin Allaert, Amel Aissaoui, José Mennesson, and Chaabane C Djeraba. Spiking-fer: spiking neural network for facial expression recognition with event cameras. In *Proceedings of the 20th International Conference on Content-based Multimedia Indexing*, pages 1–7, 2023.

[Chen *et al.*, 2021a] Xing Chen, Jingtao Li, and Chaitali Chakrabarti. Communication and computation reduction for split learning using asynchronous training. In *IEEE Workshop on Signal Processing Systems*, pages 76–81. IEEE, 2021.

[Chen *et al.*, 2021b] Yanqi Chen, Zhaofei Yu, Wei Fang, Tiejun Huang, and Yonghong Tian. Pruning of deep spiking neural networks through gradient rewiring. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence*, pages 1713–1721, 8 2021.

[Chen *et al.*, 2022a] Yanqi Chen, Zhengyu Ma, Wei Fang, Xiawu Zheng, Zhaofei Yu, and Yonghong Tian. A unified framework for soft threshold pruning. In *The Eleventh International Conference on Learning Representations*, 2022.

[Chen *et al.*, 2022b] Yanqi Chen, Zhaofei Yu, Wei Fang, Zhengyu Ma, Tiejun Huang, and Yonghong Tian. State transition of dendritic spines improves learning of sparse spiking neural networks. In *International Conference on Machine Learning*, pages 3701–3715, 2022.

[Chen *et al.*, 2023] Jiale Chen, Duc Van Le, Rui Tan, and Daren Ho. Nnfacet: Splitting neural network for concurrent smart sensors. *IEEE Transactions on Mobile Computing*, 2023.

[Chowdhury *et al.*, 2021] Sayeed Shafayet Chowdhury, Isha Garg, and Kaushik Roy. Spatio-temporal pruning and quantization for low-latency spiking neural networks. In *International Joint Conference on Neural Networks*, pages 1–9. IEEE, 2021.

[Deng and Gu, 2020] Shikuang Deng and Shi Gu. Optimal conversion of conventional artificial neural networks to spiking neural networks. In *International Conference on Learning Representations*, 2020.

[Eshraghian *et al.*, 2023] Jason K Eshraghian, Max Ward, Emre O Neftci, Xinxin Wang, Gregor Lenz, Girish Dwivedi, Mohammed Bennamoun, Doo Seok Jeong, and Wei D Lu. Training spiking neural networks using lessons from deep learning. *Proceedings of the IEEE*, 2023.

[Fang *et al.*, 2023] Wei Fang, Yanqi Chen, Jianhao Ding, Zhaofei Yu, Timothée Masquelier, Ding Chen, Liwei Huang, Huihui Zhou, Guoqi Li, and Yonghong Tian. Spikingjelly: An open-source machine learning infrastructure platform for spike-based intelligence. *Science Advances*, 9(40):eadi1480, 2023.

[Frankle and Carbin, 2018] Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *International Conference on Learning Representations*, 2018.

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

[Gupta and Raskar, 2018] Otkrist Gupta and Ramesh Raskar. Distributed learning of deep neural network over multiple agents. *Journal of Network and Computer Applications*, 116:1–8, 2018.

[Hoefler *et al.*, 2021] Torsten Hoefler, Dan Alistarh, Tal Ben-Nun, Nikoli Dryden, and Alexandra Peste. Sparsity in deep learning: Pruning and growth for efficient inference and training in neural networks. *Journal of Machine Learning Research*, 22(241):1–124, 2021.

[Hou *et al.*, 2022] Xueyu Hou, Yongjie Guan, Tao Han, and Ning Zhang. Distredge: Speeding up convolutional neural network inference on distributed edge devices. In *IEEE International Parallel and Distributed Processing Symposium*, pages 1097–1107. IEEE, 2022.

[Hu *et al.*, 2016] Hengyuan Hu, Rui Peng, Yu-Wing Tai, and Chi-Keung Tang. Network trimming: A data-driven neuron pruning approach towards efficient deep architectures. *arXiv preprint arXiv:1607.03250*, 2016.

[Jiang *et al.*, 2023] Xiaoyang Jiang, Qiang Zhang, Jingkai Sun, and Renjing Xu. Fully spiking neural network for legged robots. *arXiv preprint arXiv:2310.05022*, 2023.

[Kim *et al.*, 2017] Juyong Kim, Yookoon Park, Gunhee Kim, and Sung Ju Hwang. Splitnet: Learning to semantically split deep networks for parameter reduction and model parallelization. In *International Conference on Machine Learning*, pages 1866–1874. PMLR, 2017.

[Kim *et al.*, 2020] Jongwon Kim, Sungho Shin, Yeonguk Yu, Junseok Lee, and Kyoobin Lee. Multiple classification with split learning. In *The 9th International Conference on Smart Media and Applications*, pages 358–363, 2020.

[Kim *et al.*, 2022] 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*, pages 102–120. Springer, 2022.

[Kim *et al.*, 2023] Youngeun Kim, Yuhang Li, Hyoungseob Park, Yeshwanth Venkatesha, Anna Hambitzer, and Priyadarshini Panda. Exploring temporal information dynamics in spiking neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 8308–8316, 2023.

[Li *et al.*, 2023] Jintang Li, Zhouxin Yu, Zulun Zhu, Liang Chen, Qi Yu, Zibin Zheng, Sheng Tian, Ruofan Wu, and Changhua Meng. Scaling up dynamic graph representation learning via spiking neural networks. In *Proceedings of* *the AAAI Conference on Artificial Intelligence*, volume 37, pages 8588–8596, 2023.

[Liu *et al.*, 2024] Yue Liu, Shanlin Xiao, Bo Li, and Zhiyi Yu. Sparsespikformer: A co-design framework for token and weight pruning in spiking transformer. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 6410–6414. IEEE, 2024.

[Nguyen *et al.*, 2021] Thao NN Nguyen, Bharadwaj Veeravalli, and Xuanyao Fong. Connection pruning for deep spiking neural networks with on-chip learning. In *International Conference on Neuromorphic Systems*, pages 1–8, 2021.

[Rueckauer *et al.*, 2017] Bodo Rueckauer, Iulia-Alexandra Lungu, Yuhuang Hu, Michael Pfeiffer, and Shih-Chii Liu. Conversion of continuous-valued deep networks to efficient event-driven networks for image classification. *Frontiers in neuroscience*, 11:294078, 2017.

[Sengupta *et al.*, 2019] Abhronil Sengupta, Yuting Ye, Robert Wang, Chiao Liu, and Kaushik Roy. Going deeper in spiking neural networks: Vgg and residual architectures. *Frontiers in neuroscience*, 13:95, 2019.

[Simonyan and Zisserman, 2014] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[Su *et al.*, 2023] Qiaoyi Su, Yuhong Chou, Yifan Hu, Jianing Li, Shijie Mei, Ziyang Zhang, and Guoqi Li. Deep directly-trained spiking neural networks for object detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6555–6565, 2023.

[Thapa *et al.*, 2022] Chandra Thapa, Pathum Chamikara Mahawaga Arachchige, Seyit Camtepe, and Lichao Sun. Splitfed: When federated learning meets split learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 8485–8493, 2022.

[Wang *et al.*, 2020] Guanhua Wang, Zhuang Liu, Siyuan Zhuang, Brandon Hsieh, Joseph Gonzalez, and Ion Stoica. Sensai: Fast convnets serving on live data via class parallelism. In *MLOps Systems workshop in MLSys*, 2020.

[Yao *et al.*, 2023] 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.

[Yao *et al.*, 2024] Man Yao, Jiakui Hu, Zhaokun Zhou, Li Yuan, Yonghong Tian, Bo Xu, and Guoqi Li. Spike-driven transformer. *Advances in Neural Information Processing Systems*, 36, 2024.

[Zenke and Vogels, 2021] Friedemann Zenke and Tim P Vogels. The remarkable robustness of surrogate gradient learning for instilling complex function in spiking neural networks. *Neural computation*, 33(4):899–925, 2021.