ECC-SNN: Cost-Effective Edge-Cloud Collaboration for Spiking Neural Networks

Di Yu[1], **Changze Lv**[2], **Xin Du**[1*], **Linshan Jiang**[3], **Wentao Tong**[1], **Zhenyu Liao**[1], **Xiaoqing Zheng**[2] and **Shuiguang Deng**[1*]
[1]Zhejiang University
[2]Fudan University
[3]National University of Singapore
yudi2023@zju.edu.cn, czlv22@m.fudan.edu.cn, xindu@zju.edu.cn, linshan@nus.edu.sg
{toldzera, liaozy}@zju.edu.cn, zhengxq@fudan.edu.cn, dengsg@zju.edu.cn

Abstract

Most edge-cloud collaboration frameworks rely on the substantial computational and storage capabilities of cloud-based artificial neural networks (ANNs). However, this reliance results in significant communication overhead between edge devices and the cloud and high computational energy consumption, especially when applied to resource-constrained edge devices. To address these challenges, we propose ECC-SNN, a novel edge-cloud collaboration framework incorporating energy-efficient spiking neural networks (SNNs) to offload more computational workload from the cloud to the edge, thereby improving cost-effectiveness and reducing reliance on the cloud. ECC-SNN employs a joint training approach that integrates ANN and SNN models, enabling edge devices to leverage knowledge from cloud models for enhanced performance while reducing energy consumption and processing latency. Furthermore, ECC-SNN features an on-device incremental learning algorithm that enables edge models to continuously adapt to dynamic environments, reducing the communication overhead and resource consumption associated with frequent cloud update requests. Extensive experimental results on four datasets demonstrate that ECC-SNN improves accuracy by 4.15%, reduces average energy consumption by 79.4%, and lowers average processing latency by 39.1%.

1 Introduction

Collaborative edge-cloud computing [Wang *et al.*, 2024] leverages the extensive computational power of the cloud alongside the low-latency benefits of edge computing, enhancing data processing efficiency and real-time performance in Internet of Things (IoT) scenarios. As Spiking Neural Networks (SNNs) have emerged as a promising alternative to Artificial Neural Networks (ANNs), offering greater energy efficiency and responsiveness for on-device intelligence [Yu *et al.*, 2024; Lv *et al.*, 2024], there is a growing trend toward replacing edge-side models in conventional edge-cloud frameworks with SNNs. However, this substitution, while advantageous, introduces several significant challenges.

First, SNNs often underperform compared to ANNs of similar scale in ANN-optimized tasks, such as conventional RGB image classification [Deng *et al.*, 2020]. This performance gap is primarily attributed to accumulating gradient errors [Deng *et al.*, 2023] during the back-propagation through time (BPTT) training process with surrogate functions [Wu *et al.*, 2019]. While SNNs effectively handle most inputs, they struggle with corner cases in the long tail of data distributions—scenarios where fine-tuned ANNs typically excel.

Second, IoT devices continuously collect vast amounts of heterogeneous data from dynamic environments. Uploading all new data to the cloud for updates and retrieving revised models introduces significant communication overhead, increased latency, and excessive energy consumption [Long *et al.*, 2021]. When numerous devices request updates simultaneously, the cloud server's responsiveness and efficiency are further compromised. These limitations underscore the urgent need for efficient on-device incremental learning algorithms explicitly tailored for SNNs to enable adaptive and scalable edge-cloud collaboration frameworks.

To tackle these challenges, we propose a novel and adaptive edge-cloud collaboration framework, termed *ECC-SNN*[1], which integrates the high inference accuracy of cloud-based ANN models with the responsiveness and energy efficiency of edge-based SNN models. In ECC-SNN, we first establish a robust edge SNN backbone model, enhanced through a joint ANN-SNN training approach [Xu *et al.*, 2023]. Unlike previous approaches [Jiang *et al.*, 2023; Wang *et al.*, 2023] that improve SNN image classification accuracy by integrating ANN-inspired structures, often requiring similar architectures for both ANN and SNN models, our approach uses knowledge distillation to transfer rich information from pre-trained cloud-based ANNs to edge SNNs with arbitrary architectures, bypassing structural constraints. This method accelerates the training process while minimizing memory consumption during the *Setup* stage of ECC-SNN.

The inference *Execution* process of ECC-SNN also involves both the edge and the cloud with an ambiguity-aware

*Corresponding Authors: Xin Du and Shuiguang Deng.

[1]https://github.com/AmazingDD/ECC-SNN

strategy [Huang *et al.*, 2021]. During inference, inputs with low inference confidence—often representing corner cases or unlearned samples [Li *et al.*, 2021]—are regarded as ambiguous or 'difficult' cases and are offloaded to the robust cloud-based ANN model for reevaluation. In contrast, those with high confidence, identified as 'simple' samples, are directly processed [Bolukbasi *et al.*, 2017] by the edge SNN model. Hence, ECC-SNN can achieve a superior cost-accuracy trade-off when performing conventional image classification inference tasks compared to edge- or cloud-only solutions.

Once the edge device completes its allocated tasks, it transitions to the offline *Update* stage (e.g., during charging at the base station), where it performs adaptive on-device incremental learning (IL). During this *Execution* stage, the edge SNN learns from previously ambiguous data and corresponding logits provided by the cloud-based ANN. Given the resource constraints of edge devices, implementing complex IL methods [Xiao *et al.*, 2024; Zhou *et al.*, 2022] that demand significant memory and computational resources may not be feasible. Therefore, ECC-SNN adopts a simple yet effective self-distillation IL approach [Li and Hoiem, 2017], specifically designed to mitigate *catastrophic forgetting*. Through local updates, the edge SNN improves its performance on difficult inputs, enabling it to continuously adapt to dynamic environmental changes without additional computational overhead from the cloud.

To assess the effectiveness of ECC-SNN, we conduct extensive experiments across four image classification datasets. Results demonstrate that ECC-SNN can adaptively develop a more robust edge-based SNN model in dynamic IoT environments that performs with greater confidence while effectively offloading computational and communication costs from the cloud server. To conclude, our main contributions are summarized as follows:

- We propose **ECC-SNN**, the first edge-cloud collaboration framework to integrate cloud-based ANNs with edge-based SNNs. This novel integration leverages the complementary strengths of ANNs (high inference accuracy) and SNNs (energy efficiency and low latency), marking the first exploration of such a collaboration strategy.
- To address the dynamic IoT environmental changes, we propose an on-device incremental learning method that enhances the adaptability of edge SNN models in ECC-SNN. Furthermore, we incorporate an ambiguity filtering strategy in edge-cloud co-inference, ensuring stable accuracy performance while significantly reducing energy consumption.
- Extensive experiments on **four** diverse datasets demonstrate that ECC-SNN significantly outperforms standalone edge-based SNNs and cloud-based ANNs, achieving an average accuracy improvement of 4.15%, a 79.4% reduction in energy consumption, and a 39.1% decrease in processing latency.

2 Related Work

Collaborative Edge-Cloud Computing. Most edge-cloud collaboration frameworks heavily rely on computationally powerful servers for execution, with edge devices deploying only small models to handle basic tasks [Zhang *et al.*, 2024]. For example, ECSeg [Zhou *et al.*, 2024b] conducts an edge-cloud switched image segmentation system in autonomous vehicles with different delay tolerances. [Li *et al.*, 2024; Long *et al.*, 2021] investigate distributed inference through fine-grained model partitioning, enabling collaborative computation between servers and IoT devices. In GKT [Yao *et al.*, 2024b], edge models generate final responses with guidance prompts from larger language models in the cloud. In these works, the functionality of edge models remains highly limited due to resource constraints, motivating the exploration of more efficient alternatives to enhance edge capabilities.

Spiking Neural Networks on the Edge. The low power cost and rapid response capabilities of SNNs [Maass, 1997] align with resource-constrained edge scenarios, enabling the implementation of more extensive functionalities at the edge. EC-SNN [Yu *et al.*, 2024] employs a device collaboration method to distribute deep SNNs at the edge. Tr-Spiking-YOLO [Yuan *et al.*, 2024] is conducted on Jetson devices to implement low-latency objective detection tasks. [Zhu *et al.*, 2024] proposes to apply an end-to-end training strategy for SNNs on autonomous driving scenarios. However, existing SNN applications on edge devices primarily focus on efficient deployment. As devices operate and continuously collect new data, shifts in data distribution can lead to performance degradation, driving the introduction of on-device incremental learning methods tailored for SNNs.

On-Device Incremental Learning. Several studies have been conducted to enhance IL on edge devices, alleviating the catastrophic forgetting problem caused by dynamic environmental changes. Rehearsal-based IL methods [Zhou *et al.*, 2022] achieve favorable performance-cost trade-offs for on-device scenarios [Kwon *et al.*, 2021]. [Paissan *et al.*, 2024] introduces latent replay with sparse weight updates to reduce the learning cost. [Ma *et al.*, 2023; Lee *et al.*, 2022] employ systematic optimization methods to enhance the efficiency of limited exemplar buffers. However, these studies are primarily ANN-oriented. For on-device incremental training with SNNs, the training cost of BPTT scales with the number of time steps, making allocating additional memory for exemplars impractical. Hence, we focus on devising exemplar-free incremental learning methods tailored for SNNs.

3 Preliminary

3.1 Spiking Neural Networks

Due to the energy efficiency, SNNs are suitable for deployment on resource-constrained edge devices to implement inference tasks [Yu *et al.*, 2024]. In this study, we build the edge SNN architecture with a promising neuron model termed *Leaky Integrate-and-Fire* (LIF) [Maass, 1997], described by a series of discrete-time expressions:

$$U(t) = (1-\tau)\cdot H(t-1) + \tau \cdot I(t) \tag{1}$$

$$O(t) = \Theta(U(t) - \overline{V}) \tag{2}$$

$$H(t) = U(t)\cdot(1-O(t)) + V_r \cdot O(t) \tag{3}$$

where $\tau, \overline{V}$, and V_r denote the decay factor, threshold, and reset potential. $I(t)$ is the spatial input at time step t, $U(t)$ and $H(t)$ are the corresponding membrane potential before and after firing. Additionally, $\Theta(\cdot)$ is the Heaviside step function determining whether a spike is generated. In this study, we adopt BPTT [Dampfhoffer *et al.*, 2023] to train SNNs with surrogate gradients [Wu *et al.*, 2019].

3.2 Prior Probability Distribution Drift

The rapid influx of data (e.g., a sequence of data stream $\mathcal{D} = \{\mathcal{D}_1, ..., \mathcal{D}_N\}$) in real-world applications frequently leads to distribution drifts driven by evolving environments [Souza *et al.*, 2020]. Let the features and labels of $\mathcal{D}_n$ be denoted as $\mathbf{X}_n$ and Y_n, respectively, where $n \in \{1, ..., N\}$. *Prior probability drift* arises specifically in problems where the features depend on the labels (i.e., $\mathcal{Y} \rightarrow \mathcal{X}$), characterized by $P(Y_i|\mathbf{X}_i) = P(Y_j|\mathbf{X}_j)$ while $P(Y_i) \neq P(Y_j)$. This drift encapsulates critical challenges in adapting models to dynamic data distributions and is prevalent across various domains [Zhou *et al.*, 2024a; Su *et al.*, 2023; Wang and Sun, 2022], among which *class-incremental learning* represents a typical scenario for this shift [Masana *et al.*, 2022]. In this study, we primarily investigate the foundational capability of ECC-SNN to tackle prior probability drifts, laying the groundwork for extending its application to more complex drift scenarios.

4 Methodology

4.1 Problem Statement

Image classification is one of the conventional cognitive services provided by SNNs [Yao *et al.*, 2024a; Shi *et al.*, 2024; Qiu *et al.*, 2024b]. Although the performance of SNNs on RGB-based image classification tasks still lags behind that of ANNs [Xu *et al.*, 2023], their unique energy efficiency makes them more suitable for deployment on resource-constrained edge devices in practical applications tackling these tasks.

Considering a robot vacuum cleaner equipped with an SNN-based image classifier as the edge device for obstacle avoidance, the input image $\mathbf{x} \in \mathcal{X}$ is the data collected from the cleaner's camera, and its corresponding label $y \in \mathcal{Y}$ (e.g., table, sofa, or pets) might be a random variable drawn with a joint data distribution $P(\mathbf{x}, y)$, in which $\mathcal{Y} = \{1, ..., K\}$ and K is the number of labels. Due to the dynamic and non-ideal conditions in which inputs are collected, objects in the images may not always behave predictably (e.g., *corner cases* like a black cat might hide in a dark area, and new incoming images with *distribution drifts*). Such scenarios could lead to incorrect predictions and decisions by the locally equipped SNN model in the cleaner. In this context, relying solely on edge devices to perform all inferences locally may lead to unsatisfactory accuracy and potentially adverse outcomes.

4.2 Edge-Cloud Collaboration Framework

An edge-cloud collaborative framework [Li *et al.*, 2021] can tackle the problem mentioned in section 4.1. We first assume two discriminative models, i.e., one is a computationally intensive, high-performance large ANN-based model f_0 operating in the cloud with considerable resource utilization, and the other is a smaller, energy-efficient SNN-based model f_1 with limited capacity deployed at the edge. Both are trained on a dataset drawn *i.i.d.* from $P(\mathbf{x}, y)$, denoted by $f_0 : \mathcal{X} \rightarrow \mathcal{Y}$ and $f_1 : \mathcal{X} \rightarrow \mathcal{Y}$, respectively.

During the inference stage, a soft scoring function (a.k.a. *Filter*) $s(z|\mathbf{x}) \in [0, 1]$ with $z \in \{0, 1\}$ is applied to determine whether the small SNN f_1 is qualified to classify current input $\mathbf{x}$ or the input $\mathbf{x}$ should be off-loaded to the robust cloud-based ANN for processing f_0. Specifically, we assign label 1 to z if $s(z|\mathbf{x})$ is below some threshold δ and label it 0 otherwise, with the expectation that the filter can effectively upload *hard* inputs with low-confidence inferences from edge SNN to the cloud. Hence, the final output of an edge-cloud collaboration framework (f_0, f_1, s) w.r.t a specific input $\mathbf{x}$ is:

$$(f_0, f_1, s)(\mathbf{x}) = \begin{cases} f_1(\mathbf{x}), & \text{if } s(1|\mathbf{x}) \leq \delta \\ f_0(\mathbf{x}), & \text{o.w.} \end{cases} \tag{4}$$

and the performance of each model f_z can be evaluated by $\mathcal{L}(f_z(\mathbf{x}), y)$, where $\mathcal{L}$ could be a *cross-entropy* loss for image classification tasks. Hence, the overall expected loss of the edge-cloud collaboration framework can be calculated :

$$\begin{aligned} &\min_{f_0, f_1, s} \mathbb{E}_{P(\mathbf{x},y)} \mathbb{E}_{s(z|\mathbf{x})}[\mathcal{L}(f_z(\mathbf{x}), y)] \\ &\text{s.t. } \mathbb{E}_{P(\mathbf{x},y)} \mathbb{E}_{s(z|\mathbf{x})}[\mathcal{C}(f_z, s, \mathbf{x})] \leq b \end{aligned} \tag{5}$$

$\mathcal{C}(f_z, s, \mathbf{x})$ refers to the total cost of using a specific model for inference, e.g., energy consumption, inference latency, etc. Equation (5) seeks to minimize the expected loss of models in the edge-cloud collaborative system within a specified cost budget, and its objective can be further extended as follows:

$$\mathbb{E}_{P(\mathbf{x},y)}[s(1|\mathbf{x})\mathcal{L}(f_1(\mathbf{x}), y) + (1 - s(1|\mathbf{x}))\mathcal{L}(f_0(\mathbf{x}), y)] \tag{6}$$

In most resource-constrained edge applications, a particular budget constraint b should be established within the framework to ensure the feasibility of system operation.

However, the cloud-based ANN f_0 might be highly complex in achieving state-of-the-art (SOTA) performance, and collaboratively optimizing f_0 and f_1 could lead to slow convergence. To reduce complexity, we assume a machine learning service vendor at a remote data center might provide a fixed, pre-trained complex ANN f_0 with high accuracy performance and f_0 serves as an oracle function $\mathcal{O}(\cdot)$ to the small SNN f_1 during the optimization phase, i.e., it can correctly classify each input from $P(\mathbf{x}, y)$. Therefore, the loss term $\mathcal{L}(f_0(\mathbf{x}), y) = \mathcal{L}(\mathcal{O}(\mathbf{x}), y)$ in Equation (6) turns zero, as the outputs of the oracle function consistently correspond to the ground-truth values. Equation (5) can then be simplified as:

$$\begin{aligned} &\mathbb{E}_{P(\mathbf{x},y)}[s(1|\mathbf{x})\mathcal{L}(f_1(\mathbf{x}), y)] \\ &\text{s.t. } \mathbb{E}_{P(\mathbf{x},y)} \mathbb{E}_{s(z|\mathbf{x})}[\mathcal{C}(f_z, s, \mathbf{x})] \leq b \end{aligned} \tag{7}$$

4.3 System Design

Figure 1 outlines the entire workflow of the proposed **E**dge-**C**loud **C**ollaboration framework for on-device **S**piking **N**eural **N**etwork applications (ECC-SNN for brevity). The ECC-SNN framework comprises three primary stages. The first *Setup* stage is executed on the cloud, during which a well-trained cloud-based ANN is utilized as a teacher model for

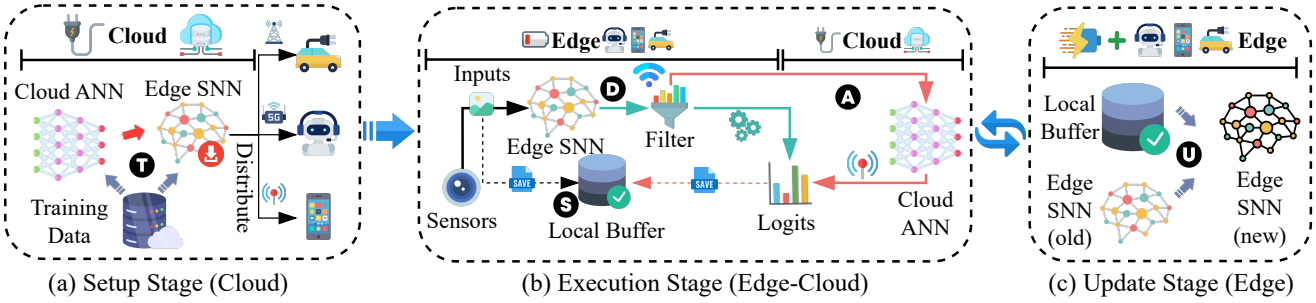

Figure 1: Overview of the proposed ECC-SNN. In all three stages, the cloud-based ANN model directly or indirectly supports the edge SNN model in preparation, inference, and adaptive updates.

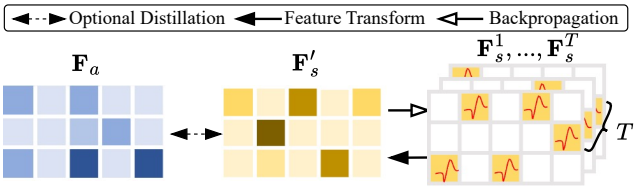

Figure 2: Optional feature distillation in the joint training approach. T is the number of time steps of the features out of the last SNN layer overlapping with the corresponding ANN layer.

jointly optimizing the initial, small-scale SNN, which will be deployed on edge devices to conduct inference tasks. In the *Execution* stage, the edge SNN will request assistance from the cloud-based ANN to classify ambiguous inputs collected from sensor data, which it cannot confidently infer. These inputs, which are hard for the current SNN to classify, will subsequently be labeled by the cloud-based ANN and stored in the local device buffer. These labeled 'hard' samples will be utilized for on-device updates in the *Update* stage to improve the SNN's performance in the next *Execution* stage.

Joint Training Approach

Although surrogate gradient methods enable directly training non-differentiable SNN, converging the training process is still complicated due to the self-accumulating gradient errors [Deng *et al.*, 2023]. To alleviate this problem, we adopt a joint training approach (process **T** in Figure 1), which distills the logit information[2] from the fine-tuned cloud-based ANN to the rough SNN model when prepared at *Setup* stage. Hence, the loss term in Equation (7) is extended as:

$$\mathcal{L}(f_1(\mathbf{x}), y) = \mathcal{L}_{ce} + \lambda_1 \mathcal{L}_{logit} \tag{8}$$

where $\mathcal{L}_{ce}$ is the cross-entropy loss to help the SNN learn from the samples in task-specific training datasets, and $\mathcal{L}_{logit}$ refers to the KL-divergence that lets the rough SNN learn the prediction distribution of the cloud-based ANN with λ_1 controls the corresponding weight.

This overlap motivates the transfer of feature knowledge between the corresponding parts of the cloud-based ANN and the edge SNN. However, these overlapping intermediate features differ in data format between ANNs and SNNs: in ANNs, features are represented in floating-point format, whereas in SNNs, they are conveyed as time-varying spike trains. To solve the disparity, we introduce external linear modules [Qiu *et al.*, 2024a] for aligning features, mapping them to the same feature space, as depicted in Figure 2. Hence, for the last overlapping layer i in edge SNN, we define the feature alignment loss as $\mathcal{L}^i_{align} = ||\mathbf{F}_{a,i} - \mathbf{F}'_{s,i}||_2$ to measure the similarity between feature-pair, in which:

$$\mathbf{F}'_{s,i} = \mathrm{BatchNorm}(\mathrm{Linear}(\sum_t \mathbf{F}^t_{s,i})) \tag{9}$$

and $\mathbf{F}_{a,i} \in \mathbb{R}^{N\times D}$ and $\mathbf{F}^t_{s,i} \in \mathbb{R}^{N\times D}$ at time step t represent the corresponding decimal and binary feature matrix for the last overlapping ANN and SNN layer i, respectively. In summary, the loss term in Equation (7) for *overlapping* cases turns to:

$$\mathcal{L}(f_1(\mathbf{x}), y) = \mathcal{L}_{ce} + \lambda_1 \mathcal{L}_{logit} + \lambda_2 \mathcal{L}^i_{align} \tag{10}$$

where λ_2 controls the weight of feature alignment loss. For *non-overlapping* cases, we remain the loss as Equation (8).

Collaborative Inference Strategy

The core of edge-cloud co-inference lies in when to filter inputs that are ambiguous to the current edge model and seek assistance from the cloud. Existing filtering strategies include rule-based [Huang *et al.*, 2020] and learning-based [Li *et al.*, 2021]. Unlike rule-based approaches, learning-based strategies incur additional computational costs and demand incremental updates as task complexity increases, posing challenges for execution on resource-constrained edge devices. Therefore, *normalized entropy* [Huang *et al.*, 2021] is employed in ECC-SNN as the filtering criterion $s(1|\mathbf{x})$ to determine the cloud upload rate for ambiguous inputs during *Execution* stage. As demonstrated in Figure 3, this metric provides a practical and interpretable measure of inference confidence for the edge SNN model and is defined as follows:

$$s(1|\mathbf{x}) = -\sum_{k=1}^{K} \frac{\sigma(f_{1,k}(\mathbf{x}))\log(\sigma(f_{1,k}(\mathbf{x})))}{\log(K)} \tag{11}$$

where $\sigma(\cdot)$ denotes the soft-max function converting the output of each edge SNN's classification head $f_{1,k}(\cdot)$ into a probability distribution. To be more specific, the filter in ECC-SNN will compute the corresponding entropy values for each input with the edge SNN (process **D** in Figure 1). Inputs

[2]Logit refers to the output of the model's final classification layer

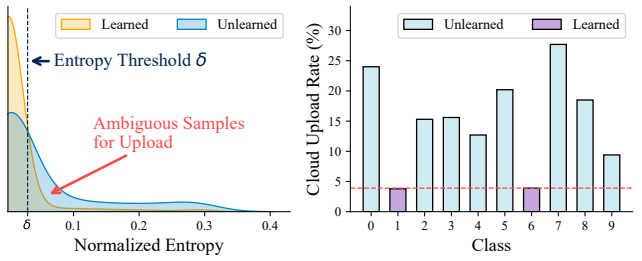

Figure 3: Case Study: a *spiking VGG-9* model learned with CIFAR-10 limited to samples labeled 1 and 6. The entropy distributions and corresponding cloud upload rates for each label are derived from each test sample's model output predictive distribution.

Algorithm 1 Collaborative Inference in ECC-SNN

Input: collected input feature $\mathbf{x}$; filter threshold δ.
Parameter: cloud ANN model f_0; edge SNN model f_1.
Output: the inference result $\hat{y}$.

1: $\mathbf{z} \leftarrow f_1(\mathbf{x})$, $\mathbf{p} \leftarrow \sigma(\mathbf{z})$;
2: Compute $s(1|\mathbf{x})$ for $\mathbf{p}$ with Equation (11);
3: **if** $s(1|\mathbf{x}) \leq \delta$ **then**
4: $\quad \hat{y} \leftarrow \text{argmax}_k(\mathbf{p})$, $k \in \{1, ..., K\}$;
5: **else**
6: $\quad$ `# line 7-9: process` **A** `in Figure 1`
7: $\quad$ Upload $\mathbf{x}$ to the cloud, $\mathbf{z} \leftarrow f_0(\mathbf{x})$;
8: $\quad \hat{y} \leftarrow \text{argmax}_k(\mathbf{z})$, $k \in \{1, ..., K\}$;
9: $\quad$ Broadcast $\hat{y}$ back to the edge device;
10: $\quad$ `# line 11: process` **S** `in Figure 1`
11: $\quad$ Store sample $(\mathbf{x}, \hat{y})$ in local buffer of the edge device;
12: **end if**
13: **return** $\hat{y}$

with normalized entropy greater than the pre-set threshold δ are determined ambiguous samples that require uploading to the cloud for further assistance.

Algorithm 1 describes how ECC-SNN performs collaborative inference during the *Execution* stage. Each newly collected image from the sensor will first undergo inference using the edge SNN, and the resulting filtering score $s(1|\mathbf{x})$ will be computed to assess the inference confidence. Images with low confidence will be uploaded to the cloud, where the powerful ANN will perform re-inference and return the results. These samples requiring assistance will subsequently use the predicted logits from the cloud-based ANN as their label references, forming local training samples stored in the edge device's local buffer for on-device updates of the SNN model.

On-device Incremental Learning Method

Although the co-inference mechanism in ECC-SNN substantially enhances accuracy by refining the results produced by the edge SNN, it may also introduce additional communication latency and energy overheads. This is particularly true when the edge SNN faces challenges related to environmental mobility, such as data distribution drift [Shao *et al.*, 2024], inducing numerous ambiguous inferences with low confidence. For instance, a robot may be delivered to a user's home with default object recognition capabilities. However, it may fail to recognize new, site-specific objects reliably (a.k.a. suffering from *prior probability distribution drift*).

Therefore, IL methods are employed to progressively improve the performance of the edge SNN in handling these ambiguities. (process **U** in Figure 1). Meanwhile, this learning process should be conducted on edge devices rather than on a centralized cloud server during the offline stage (e.g., recharging at the base station). In-situ processing helps avoid transmitting high-volume information over networks, thereby reducing the bandwidth requirements [Kukreja *et al.*, 2019].

Unlike humans who continually learn evolving tasks throughout their lifetime, the edge SNN model suffers from *catastrophic forgetting* problems [Li and Hoiem, 2017] when conducting IL. Although many IL methods can mitigate this issue [Xiao *et al.*, 2024; Zhou *et al.*, 2022], they often require additional memory and computational resources, making them unsuitable for deployment on resource-constrained edge devices. Therefore, our proposed framework will provide a bio-plausible explanation for this method) for conducting on-device IL for the edge SNN, which continually training the edge SNN model with a new loss function during *Update* stage:

$$\mathcal{L}(f_1(\mathbf{x}), y) = \mathcal{L}_{new} + \lambda_3 \mathcal{L}_{old} \tag{12}$$

where $\mathcal{L}_{new}$ is analogous to Equation (8) but is specifically computed using samples stored in the local buffer, supplemented by logits obtained from the cloud-based ANNs. $\mathcal{L}_{old}$, evaluated by KL-divergence with the probabilities from the old SNN model $\sigma(f_1^{old}(\mathbf{x}))$ and the new SNN model $\sigma(f_1^{new}(\mathbf{x}))$, prevents forgetting the knowledge of previous tasks by forcing the model to predict similar outputs as the previous SNN model for old task data. The importance of $\mathcal{L}_{old}$ is controlled by a hyper-parameter λ_3. The local buffer will be flushed after each incremental update is completed.

5 Experiments

5.1 Experimental Settings

Following [Zhou *et al.*, 2024a], we evaluate the effectiveness of our proposed ECC-SNN using the standard class-incremental learning setting, as it represents a typical form of prior probability distribution drift commonly encountered in real-world applications [Diao *et al.*, 2024]. We denote the data split setting as 'w/ B-u, Inc-v,' i.e., the first dataset contains u classes, and each following dataset contains v classes. $u = 0$ means the total classes are equally divided into each task. By default, we adopt the *spiking VGG-9* model as the SNN deployed on the edge device featuring a neuromorphic chip [Ma *et al.*, 2024] while utilizing a widely recognized pre-trained ViT model *vit-base-patch16* as the cloud-based ANN.

5.2 Performance Evaluation

Test Accuracy. Denote the average Top-1 accuracy ($\bar{\mathcal{A}}_n$) after the n-th task as $\bar{\mathcal{A}}_n = \frac{1}{n}\sum_{m=1}^{n} a_{n,m}$, where $a_{n,m} \in [0, 1]$ is the accuracy of task m after learning task n ($m \leq n$). We first evaluate the effectiveness of the proposed joint training approach for accuracy improvement at *Setup* stage. As listed in Table 1, we compare the performance of this approach with that of standalone training approaches for edge SNN and cloud-based ANN models, which demonstrate that

Dataset	ANN Arch.	$\bar{\mathcal{A}}_{1,\mathrm{E}}$ (%)	$\bar{\mathcal{A}}_{1,\mathrm{C}}$ (%)	$\bar{\mathcal{A}}_{1,\mathrm{EC}}$ (%)
CIFAR-10	VGG-16†	86.75	93.41	(↑ 3.82) 90.57
	ResNet-34	86.75	95.23	(↑ 3.79) 90.54
	ViT-12	86.75	98.05	(↑ 3.88) 90.63
CIFAR-100	VGG-16†	83.20	89.10	(↑ 2.25) 85.45
	ResNet-34	83.20	91.50	(↑ 2.15) 85.35
	ViT-12	83.20	94.75	(↑ 2.30) 85.50
Caltech	VGG-16†	94.23	98.04	(↑ 0.76) 95.04
	ResNet-34	94.23	98.39	(↑ 0.61) 94.84
	ViT-12	94.23	99.19	(↑ 0.76) 95.04
Tiny-ImageNet	VGG-16†	62.05	69.50	(↑ 2.75) 64.80
	ResNet-34	62.05	75.50	(↑ 4.85) 66.90
	ViT-12	62.05	89.50	(↑ 6.55) 68.60

Table 1: Average accuracy performance at Task 1 ($\bar{\mathcal{A}}_1$) of edge-side *spiking VGG-9* w.r.t. various pre-trained cloud-based ANN architecture (Arch.) pairs across different RGB-based image datasets. Footnotes 'E'/'C'/'EC' represent the edge SNN, cloud-based ANN, and ECC-SNN. ANN Arch. with '†' indicates the overlapping case in which the dimensions of their early layers are identical to those of the paired SNN. All results are averaged across three random seeds.

$\mathcal{L}^i_{align}$	$\mathcal{L}_{logit}$	CIFAR-100	Tiny-ImageNet
✗	✗	83.20	62.05
✓	✗	(↑ 0.75) 83.95	(↑ 1.35) 63.40
✗	✓	(↑ 0.95) 84.15	(↑ 2.10) 64.15
✓	✓	(↑ 2.25) 85.45	(↑ 2.75) 64.80

Table 2: Ablation results of the proposed regularization terms for the overlapping VGG cases.

the joint training design in ECC-SNN can achieve an average accuracy improvement of 2.87% for edge SNNs compared to the standalone direct training approach.

The enhancement of SNN performance varies depending on the choice of the pre-trained ANN model. Although VGG16 is less complex than other pre-trained ANN models like ResNet and ViT, it can still serve as an effective teacher model to assist SNN convergence on simpler tasks such as CIFAR and Caltech, and the final performance gap is even less than 0.1%. However, we must acknowledge that in complex tasks like ImageNet, the performance limitations of VGG-16 itself constrain its ability to provide substantial guidance to Spiking VGG-9. As a result, the final accuracy improvement falls short of the improvement achieved with those complex ANNs. Observations in Table 2 elaborate on the contribution of each proposed regularization term in those VGG-structured overlapping cases, from which we conclude that both terms demonstrate effectiveness for the final accuracy, with $\mathcal{L}_{logit}$ being the dominant factor. In addition, their combination can further enhance the learning performance across different datasets.

The adaptive update strategy in ECC-SNN is designed to continuously enhance the predictive capabilities of the edge SNN model, thereby offloading as much computational burden and communication cost from the cloud-based ANN model as possible. We define the cloud upload rate (CUR) [Li *et al.*, 2021] to represent the proportion of ambiguous inputs uploaded to the cloud for assistance, representing the communication overhead of co-inference and denoted as:

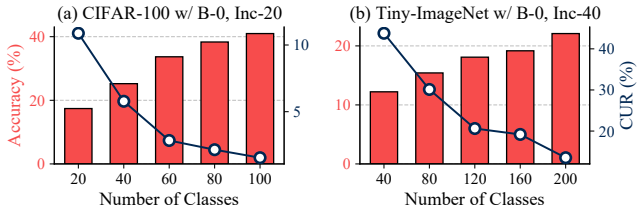

Figure 4: Changing patterns of accuracy performance and CUR as more classes learned, with a fixed filtering threshold δ=0.3.

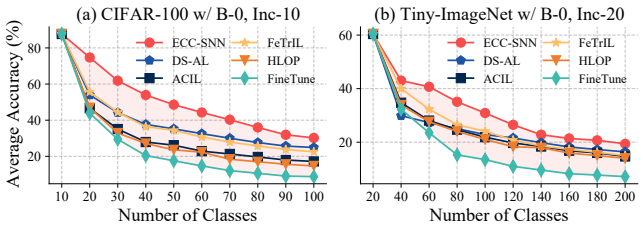

Figure 5: Average accuracy of edge SNN w.r.t. different IL methods.

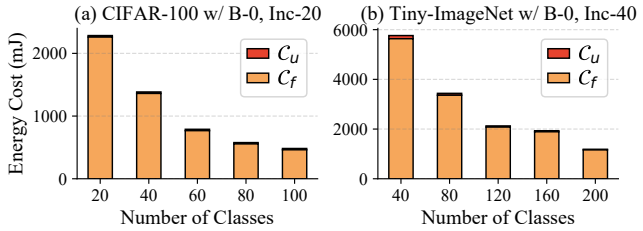

Figure 6: Average theoretical energy cost (mJ) per input for inference as more classes learned in ECC-SNN, with a fixed filtering threshold δ=0.3. The communication cost $\mathcal{C}_u$ is negligible compared to the computation cost $\mathcal{C}_f$.

$$\mathrm{CUR} = \mathbb{E}_{P(\mathbf{x},y)}\left[\mathbb{I}(s(1|\mathbf{x}) > \delta)\right] \tag{13}$$

where $\mathbb{I}(\cdot)$ is the indicator function. When using the full test dataset as a simulation of a real-world environment to evaluate the proposed system, tendencies in Figure 4 depict that the model's accuracy consistently improves with incremental updates. As the edge SNN model gains higher inference confidence in classifying, it will reduce the reliance on the complicated cloud-based ANN model to recognize ambiguous 'difficult' inputs, thereby diminishing CUR.

Figure 5 compares various SOTA IL methods proposed recently with the adaptive update approach in our ECC-SNN. Note that all methods experience a degradation in $\bar{\mathcal{A}}$ due to catastrophic forgetting as the number of classes increases. In ANN-oriented IL tasks, freezing the backbone weights effectively preserves the existing knowledge. However, in SNNs, the weights represent synaptic connections, and freezing them completely restricts the SNN's ability to continue learning. Therefore, ECC-SNN is more effective at mitigating this impact than other methods across various scenarios, achieving an improvement of 5.32% over the second-best method on CIFAR-100 and 2.98% on ImageNet, respectively.

Energy Cost. The energy cost of ECC-SNN during the *Execution* stage contains two main components: communication

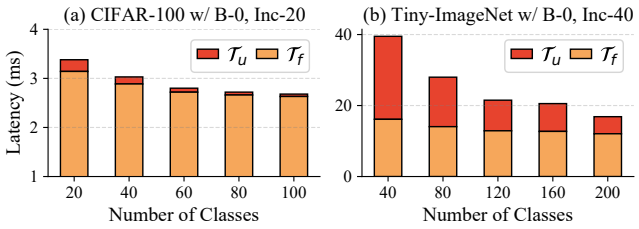

Figure 7: Average inference latency (ms) per input in ECC-SNN as more classes learned, with a fixed filtering threshold δ=0.3.

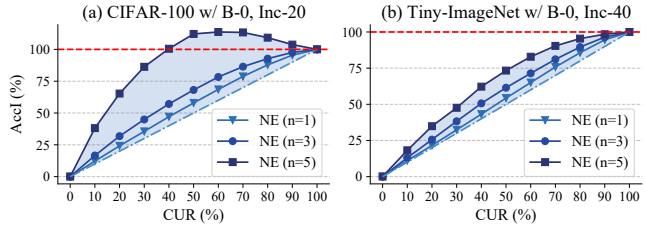

Figure 8: Accuracy improvement (AccI) w.r.t CUR (%) with different learned classes at current task.

overheads $\mathcal{C}_u$ and selected computational cost $\mathcal{C}_f$. Figure 6 introduces changing cost patterns as tasks evolve, indicating that (1) The computational cost of the cloud-based ANN model constitutes the majority of the total energy consumption. (2) As the edge model is continuously updated, its inference capability gradually improves, reducing the reliance on the cloud-based model. This enhancement leverages the energy efficiency of SNNs, reducing the average energy consumption for inference. (3) The communication overhead becomes non-negligible as the input size increases. For example, it contributes 2.2% of the total cost at the first task in the ImageNet scenario.

Latency. The inference latency of ECC-SNN also consists of communication latency $\mathcal{T}_u$ and model computation latency $\mathcal{T}_f$. Figure 7 introduces the latency patterns of ECC-SNN when inferring one sample for different datasets as tasks evolve. It can be intuitively concluded that (1) As the capability of the edge SNN improves, reliance on the powerful ANN model on the cloud diminishes, reducing the frequency of request transmissions and the latency associated with waiting for inference results. (2) Simple CIFAR-100 inputs are more likely to be processed directly on the edge rather than uploaded to the cloud. Hence, the average computational latency acceleration per input is 9.07% lower than that in ImageNet. (3) When the input size is large, running spiking VGG-9 on a neuromorphic chip is only 21.9% faster than running ViT-12 on a server GPU. Therefore, reducing the frequency of requesting assistance from the cloud cannot diminish the total computation latency but can accelerate the communication latency by around 79.7%. Therefore, when evaluating the effectiveness of the ECC-SNN system, communication latency and model inference latency should be considered equally important metrics.

5.3 Sensitive Analysis

Following [Li *et al.*, 2021], we define the relative accuracy improvement (AccI) to measure the accuracy improvement of ECC-SNN $\mathcal{A}_{(f_0,f_1,s)}$ compared to the standalone SNN $\mathcal{A}_{f_1}$ deployed at the edge, normalized by the accuracy gap between the cloud-based ANN $\mathcal{A}_{f_0}$ and the edge SNN $\mathcal{A}_{f_1}$. As shown in Figure 8, the normalized entropy (NE) strategy in ECC-SNN demonstrates its effectiveness by accurately identifying ambiguous inputs and improving the utility of requesting assistance from the cloud model, compared to random uploading. Meanwhile, it is evident that as the edge model's predictive capabilities improve, the marginal benefit of requesting assistance from the cloud gradually diminishes. A surprising observation is that ECC-SNN can improve accuracy in simple classification tasks. For example, the overall accuracy of ECC-SNN can exceed that of the standalone cloud-based ANN model (red dashed line) by up to approximately 13.3% in task 5 of CIFAR-100 when the CUR is in the range of [40, 100]. This is because, although the edge SNN initially exhibits lower accuracy across the entire dataset, it is continuously optimized with incoming data from different tasks to minimize overall expected loss. As a result, it is likely to correctly predict a subset of inputs that the cloud-based ANN model fails to classify correctly. However, for the complex ImageNet dataset, the accuracy gap between the cloud and edge models is too large to enable accuracy boosting. Under such circumstances, the edge model can continuously benefit from the cloud, allowing a better trade-off between accuracy and cost within ECC-SNN.

6 Conclusion

In this study, we propose ECC-SNN, a cost-effective and efficient edge-cloud collaborative framework designed for SNN-based classifiers. By employing the joint training approach and adaptive on-device incremental learning with the assistance of a powerful ANN model on the cloud server, the SNN model in ECC-SNN gains enhanced predictive capability compared to the standalone edge SNN, significantly reducing both energy costs and inference latency as the system operates in different dynamic IoT scenarios.

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, the Zhejiang Provincial Natural Science Foundation of China under Grant No. LD24F020014, the National Key Research and Development Program of China No. 2024YDLN0005, and the Regional Innovation and Development Joint Fund of the National Natural Science Foundation of China No. U22A6001.

References

[Bolukbasi *et al.*, 2017] Tolga Bolukbasi, Joseph Wang, Ofer Dekel, and Venkatesh Saligrama. Adaptive neural networks for efficient inference. In *International Conference on Machine Learning*, pages 527–536. PMLR, 2017.

[Dampfhoffer *et al.*, 2023] Manon Dampfhoffer, Thomas Mesquida, Alexandre Valentian, and Lorena Anghel.

Backpropagation-based learning techniques for deep spiking neural networks: A survey. *IEEE Transactions on Neural Networks and Learning Systems*, 2023.

[Deng *et al.*, 2020] Lei Deng, Yujie Wu, Xing Hu, Ling Liang, Yufei Ding, Guoqi Li, Guangshe Zhao, Peng Li, and Yuan Xie. Rethinking the performance comparison between snns and anns. *Neural networks*, 121:294–307, 2020.

[Deng *et al.*, 2023] Shikuang Deng, Hao Lin, Yuhang Li, and Shi Gu. Surrogate module learning: Reduce the gradient error accumulation in training spiking neural networks. In *International Conference on Machine Learning*, pages 7645–7657. PMLR, 2023.

[Diao *et al.*, 2024] Yiqun Diao, Yutong Yang, Qinbin Li, Bingsheng He, and Mian Lu. Oebench: Investigating open environment challenges in real-world relational data streams. *Proc. VLDB Endow.*, 17(6):1283–1296, 2024.

[Huang *et al.*, 2020] Yakun Huang, Xiuquan Qiao, Pei Ren, Ling Liu, Calton Pu, Schahram Dustdar, and Junliang Chen. A lightweight collaborative deep neural network for the mobile web in edge cloud. *IEEE Transactions on Mobile Computing*, 21(7):2289–2305, 2020.

[Huang *et al.*, 2021] Yakun Huang, Xiuquan Qiao, Jian Tang, Pei Ren, Ling Liu, Calton Pu, and Junliang Chen. An integrated cloud-edge-device adaptive deep learning service for cross-platform web. *IEEE Transactions on Mobile Computing*, 22(4):1950–1967, 2021.

[Jiang *et al.*, 2023] Haiyan Jiang, Srinivas Anumasa, Giulia De Masi, Huan Xiong, and Bin Gu. A unified optimization framework of ann-snn conversion: towards optimal mapping from activation values to firing rates. In *International Conference on Machine Learning*, pages 14945–14974. PMLR, 2023.

[Kukreja *et al.*, 2019] Navjot Kukreja, Alena Shilova, Olivier Beaumont, Jan Huckelheim, Nicola Ferrier, Paul Hovland, and Gerard Gorman. Training on the edge: The why and the how. In *2019 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, pages 899–903. IEEE, 2019.

[Kwon *et al.*, 2021] Young D Kwon, Jagmohan Chauhan, Abhishek Kumar, Pan Hui Hkust, and Cecilia Mascolo. Exploring system performance of continual learning for mobile and embedded sensing applications. In *2021 IEEE/ACM Symposium on Edge Computing (SEC)*, pages 319–332. IEEE, 2021.

[Lee *et al.*, 2022] Soobee Lee, Minindu Weerakoon, Jonghyun Choi, Minjia Zhang, Di Wang, and Myeongjae Jeon. Carm: Hierarchical episodic memory for continual learning. In *Proceedings of the 59th ACM/IEEE Design Automation Conference*, pages 1147–1152, 2022.

[Li and Hoiem, 2017] Zhizhong Li and Derek Hoiem. Learning without forgetting. *IEEE transactions on pattern analysis and machine intelligence*, 40(12):2935–2947, 2017.

[Li *et al.*, 2021] Min Li, Yu Li, Ye Tian, Li Jiang, and Qiang Xu. Appealnet: An efficient and highly-accurate edge/cloud collaborative architecture for dnn inference. In *2021 58th ACM/IEEE Design Automation Conference (DAC)*, pages 409–414. IEEE, 2021.

[Li *et al.*, 2024] Hui Li, Xiuhua Li, Qilin Fan, Qiang He, Xiaofei Wang, and Victor CM Leung. Distributed dnn inference with fine-grained model partitioning in mobile edge computing networks. *IEEE Transactions on Mobile Computing*, 2024.

[Long *et al.*, 2021] Yinghan Long, Indranil Chakraborty, Gopalakrishnan Srinivasan, and Kaushik Roy. Complexity-aware adaptive training and inference for edge-cloud distributed ai systems. In *2021 IEEE 41st International Conference on Distributed Computing Systems (ICDCS)*, pages 573–583. IEEE, 2021.

[Lv *et al.*, 2024] Changze Lv, Yansen Wang, Dongqi Han, Xiaoqing Zheng, Xuanjing Huang, and Dongsheng Li. Efficient and effective time-series forecasting with spiking neural networks. In *Forty-first International Conference on Machine Learning*, 2024.

[Ma *et al.*, 2023] 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*, pages 1–15, 2023.

[Ma *et al.*, 2024] De Ma, Xiaofei Jin, Shichun Sun, Yitao Li, Xundong Wu, Youneng Hu, Fangchao Yang, Huajin Tang, Xiaolei Zhu, Peng Lin, et al. Darwin3: a large-scale neuromorphic chip with a novel isa and on-chip learning. *National Science Review*, 11(5):nwae102, 2024.

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

[Masana *et al.*, 2022] Marc Masana, Xialei Liu, Bartłomiej Twardowski, Mikel Menta, Andrew D Bagdanov, and Joost Van De Weijer. Class-incremental learning: survey and performance evaluation on image classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(5):5513–5533, 2022.

[Paissan *et al.*, 2024] Francesco Paissan, Davide Nadalini, Manuele Rusci, Alberto Ancilotto, Francesco Conti, Luca Benini, and Elisabetta Farella. Structured sparse back-propagation for lightweight on-device continual learning on microcontroller units. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2172–2181, 2024.

[Qiu *et al.*, 2024a] Haonan Qiu, Munan Ning, Zeyin Song, Wei Fang, Yanqi Chen, Tao Sun, Zhengyu Ma, Li Yuan, and Yonghong Tian. Self-architectural knowledge distillation for spiking neural networks. *Neural Networks*, page 106475, 2024.

[Qiu *et al.*, 2024b] Xuerui Qiu, Rui-Jie Zhu, Yuhong Chou, Zhaorui Wang, Liang-jian Deng, and Guoqi Li. Gated attention coding for training high-performance and efficient

spiking neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 601–610, 2024.

[Shao *et al.*, 2024] Minglai Shao, Dong Li, Chen Zhao, Xintao Wu, Yujie Lin, and Qin Tian. Supervised algorithmic fairness in distribution shifts: A survey. In Kate Larson, editor, *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence, IJCAI-24*, pages 8225–8233, 2024.

[Shi *et al.*, 2024] Xinyu Shi, Zecheng Hao, and Zhaofei Yu. Spikingresformer: Bridging resnet and vision transformer in spiking neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5610–5619, 2024.

[Souza *et al.*, 2020] Vinicius MA Souza, Denis M dos Reis, Andre G Maletzke, and Gustavo EAPA Batista. Challenges in benchmarking stream learning algorithms with real-world data. *Data Mining and Knowledge Discovery*, 34(6):1805–1858, 2020.

[Su *et al.*, 2023] Junwei Su, Difan Zou, Zijun Zhang, and Chuan Wu. Towards robust graph incremental learning on evolving graphs. In *International Conference on Machine Learning*, pages 32728–32748. PMLR, 2023.

[Wang and Sun, 2022] Zifeng Wang and Jimeng Sun. Transtab: Learning transferable tabular transformers across tables. *Advances in Neural Information Processing Systems*, 35:2902–2915, 2022.

[Wang *et al.*, 2023] Bingsen Wang, Jian Cao, Jue Chen, Shuo Feng, and Yuan Wang. A new ann-snn conversion method with high accuracy, low latency and good robustness. In *IJCAI*, pages 3067–3075, 2023.

[Wang *et al.*, 2024] Yingchao Wang, Chen Yang, Shulin Lan, Liehuang Zhu, and Yan Zhang. End-edge-cloud collaborative computing for deep learning: A comprehensive survey. *IEEE Communications Surveys & Tutorials*, 2024.

[Wu *et al.*, 2019] Yujie Wu, Lei Deng, Guoqi Li, Jun Zhu, Yuan Xie, and Luping Shi. Direct training for spiking neural networks: Faster, larger, better. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 1311–1318, 2019.

[Xiao *et al.*, 2024] Mingqing Xiao, Qingyan Meng, Zongpeng Zhang, Di He, and Zhouchen Lin. Hebbian learning based orthogonal projection for continual learning of spiking neural networks. In *The Twelfth International Conference on Learning Representations*, 2024.

[Xu *et al.*, 2023] Qi Xu, Yaxin Li, Jiangrong Shen, Jian K Liu, Huajin Tang, and Gang Pan. Constructing deep spiking neural networks from artificial neural networks with knowledge distillation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7886–7895, 2023.

[Yao *et al.*, 2024a] 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.

[Yao *et al.*, 2024b] Yao Yao, Zuchao Li, and Hai Zhao. Gkt: A novel guidance-based knowledge transfer framework for efficient cloud-edge collaboration llm deployment. *arXiv preprint arXiv:2405.19635*, 2024.

[Yu *et al.*, 2024] 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, IJCAI-24*, pages 5389–5397, 2024.

[Yuan *et al.*, 2024] Mengwen Yuan, Chengjun Zhang, Ziming Wang, Huixiang Liu, Gang Pan, and Huajin Tang. Trainable spiking-yolo for low-latency and high-performance object detection. *Neural Networks*, 172:106092, 2024.

[Zhang *et al.*, 2024] Ziyang Zhang, Yang Zhao, Huan Li, Changyao Lin, and Jie Liu. Dvfo: Learning-based dvfs for energy-efficient edge-cloud collaborative inference. *IEEE Transactions on Mobile Computing*, 2024.

[Zhou *et al.*, 2022] Da-Wei Zhou, Qi-Wei Wang, Han-Jia Ye, and De-Chuan Zhan. A model or 603 exemplars: Towards memory-efficient class-incremental learning. *CoRR*, abs/2205.13218, 2022.

[Zhou *et al.*, 2024a] Da-Wei Zhou, Qi-Wei Wang, Zhi-Hong Qi, Han-Jia Ye, De-Chuan Zhan, and Ziwei Liu. Class-incremental learning: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024.

[Zhou *et al.*, 2024b] Siyuan Zhou, Duc Van Le, and Rui Tan. Ecseg: Edge-cloud switched image segmentation for autonomous vehicles. In *2024 21th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON)*. IEEE, 2024.

[Zhu *et al.*, 2024] Rui-Jie Zhu, Ziqing Wang, Leilani H. Gilpin, and Jason Eshraghian. Autonomous driving with spiking neural networks. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.