12 mJ per Class On-Device Online Few-Shot Class-Incremental Learning

Yoga Esa Wibowo‡, Cristian Cioflan*†, Thorir Mar Ingolfsson†, Michael Hersche†¶,
Leo Zhao‡, Abbas Rahimi¶, Luca Benini†§
‡*D-ITET, ETH Zurich;* †*Integrated Systems Laboratory, ETH Zurich;* ¶*IBM Research-Zurich;* §*DEI, University of Bologna*
{ywibowo, lezhao}@ethz.ch, {cioflanc, thoriri, lbenini}@iis.ee.ethz.ch, {her, abr}@zurich.ibm.com

***Abstract*—Few-Shot Class-Incremental Learning (FSCIL) enables machine learning systems to expand their inference capabilities to new classes using only a few labeled examples, without forgetting the previously learned classes. Classical backpropagation-based learning and its variants are often unsuitable for battery-powered, memory-constrained systems at the extreme edge. In this work, we introduce Online Few-Shot Class-Incremental Learning (O-FSCIL), based on a lightweight model consisting of a pretrained and metalearned feature extractor and an expandable explicit memory storing the class prototypes. The architecture is pretrained with a novel feature orthogonality regularization and metalearned with a multi-margin loss. For learning a new class, our approach extends the explicit memory with novel class prototypes, while the remaining architecture is kept frozen. This allows learning previously unseen classes based on only a few examples with one single pass (hence online). O-FSCIL obtains an average accuracy of 68.62% on the FSCIL CIFAR100 benchmark, achieving state-of-the-art results. Tailored for ultra-low-power platforms, we implement O-FSCIL on the 60 mW GAP9 microcontroller, demonstrating online learning capabilities within just 12 mJ per new class.**

***Index Terms*—Continual learning, on-device learning, deep neural networks, microcontrollers.**

I. Introduction

Classical Machine Learning (ML) solutions often use large datasets to train a highly complex yet fixed model, which cannot adapt to the needs and requirements of the end user. Such systems are nonetheless exposed to dynamic, ever-changing environments; thus, adaptability is a crucial requirement for an intelligent system. Moreover, while in a server-based, offline learning paradigm, curated and labeled data is widely available, that is seldom the case when a pretrained model must adapt to a particular user. Few-Shot Class-Incremental Learning (FSCIL) evaluates models on two aforementioned problems, requiring a previously trained model to expand its classification domain while providing very few labeled training samples. The key challenge is to prevent catastrophic forgetting (i.e., forgetting prior knowledge during knowledge expansion) and avoid overfitting to the few novel samples. These goals can be achieved by balancing stability and plasticity [1]. Several FSCIL solutions have tackled sample-scarce class learning over several incremental sessions [2], [3] in a transfer learning fashion, where a model is pretrained to classify a set of predefined base classes, followed by freezing its backbone and only training its classification head to learn novel classes.

Although such methods achieve remarkable results on previously unseen classes, they rely on the computationally expensive backpropagation algorithm. Alternative solutions, such as Neural Collapse Few-Shot Class-Incremental Learning (NC-FSCIL) [4], Semantic-Aware Virtual Contrastive (SAVC) model [5], or Constrained Few-Shot Class-Incremental Learning (C-FSCIL) [6], [7], use large ResNet backbones [8]. However, adapting to new requirements and learning novel classes should happen directly on the user device, thus addressing privacy and security concerns [9], while also avoiding energy-hungry data streaming to the cloud. Therefore, when employed at the extreme edge, FSCIL algorithms should operate under the TinyML constraints [10]; hence, suitably small and computationally affordable backbones are required.

On-device training frameworks targeting Microcontroller Units (MCUs) have been proposed [11]–[13], which encouraged the development of domain adaptation [14] and class continual learning [15] strategies tailored for MCUs. However, such iterative learning approaches require storing training samples during the update process. Our proposed online training methodology addresses the storage constraints by enabling novel class learning with a single pass over the available samples. This also decreases the learning time and, thus, the energy consumption of our system. Moreover, as opposed to FSCIL strategies deployed on Tensor Processing Units (TPUs) [16], Field Programmable Gate Arrays (FPGAs) [17], neuromorphic chips [18], or in-memory processors [7], our methodology is suitable for off-the-shelf, widely available MCUs, with a conservative power profile.

This work introduces Online Few-Shot Class-Incremental Learning (O-FSCIL)[1], a lightweight FSCIL learning methodology aimed at resource-constrained edge devices. Using orthogonal regularization and data augmentation strategies such as Mixup [19] and Cutmix [20], we pretrain the MobileNetV2 [21] backbone, which requires only 2.5 million parameters and 149.2 million Multiply-Accumulate (MAC) operations. During the server-side metalearning phase, we employ a multi-margin loss to prevent overfitting to the few labeled samples. We further demonstrate the on-device learning capabilities by deploying and evaluating O-FSCIL on ultra-low-power, commercially available GAP9 [22].

The main contributions of the paper are as follows:

* Corresponding author

[1]The code will be open-sourced at: https://github.com/pulp-platform/fscil

979-8-3503-4859-0/DATE24/© 2024 EDAA

- We introduce O-FSCIL, a class-incremental learning method achieving a new state-of-the-art average accuracy of 68.62% on the FSCIL CIFAR100 [23] benchmark.
- For each novel class, our pretrained and metalearned backbone generates orthogonal feature vectors quantized to 3-bit integers, which are stored in the Explicit Memory (EM), resulting in memory requirements of only $9.6\,\mathrm{kB}$ for 100 classes.
- We demonstrate few-shot online (i.e., single-pass) learning capabilities on a $50\,\mathrm{mW}$, requiring as little as $12\,\mathrm{mJ}$ to learn a new class.

II. Related Work

A. Few-shot Class-incremental Learning

Different approaches have been proposed to tackle the FSCIL scenario, in which models presented with out-of-distribution data learn new classes from few labeled data. TOPIC [1] introduces the FSCIL problem and employs a neural gas technique to maintain the topology in the embedding space. By updating the backbone of the neural network [1], [24], [25], one can successfully learn to recognize previously unseen classes. To learn new classes and avoid catastrophic forgetting without maintaining a large reservoir memory, we freeze the backbone and only store class prototypes in the explicit memory.

To reduce the costs of retraining the backbone, other works [2], [3] proposed to freeze the backbone and operate on the classifier and its additional components. Works such as [2], [3], [26] rely on an episodic memory, where class-specific information is stored and compared with the query image during inference. Zhou et al. [26] introduce a forward compatibility methodology, where a provident model minimizes the negative compression effects novel classes have on the embedding space. ALICE [27] proposes the angular penalty loss to achieve compact clustering and feature diversity, achieving better generalization for unseen classes. Conversely, SAVC [5] enhance the cluster separation with virtual classes created through predefined transformations, which diversify the semantic information. However, these methods do not generate meaningful feature representations in space between class clusters, thus impeding clustering in that vacant space.

NC-FSCIL [4] tackles the clustering problem by creating a placeholder for all class prototypes with fixed, predetermined vectors, thus addressing the prototype readjustment issue when adding a new class cluster. As opposed to these works, our methodology improves features' representations and expressiveness already during the pretraining and metalearning stages, through feature interpolation, as well as orthogonality and multi-margin loss-based network update. This allows us to achieve higher accuracy levels with inexpensive few-shot class-incremental learning, suitable for on-device deployment.

B. On-device continual learning

While deployment frameworks enabling Deep Neural Network (DNN) inference are already well established [28], on-device training frameworks [11], [13] are yet to generalize to multple Deep Learning (DL) topologies and generally target single-platform families. Notably, Nadalini et al. [12] propose a framework aimed at both single- and multi-core platforms, accounting for the particular memory hierarchy of each target device. To improve accessibility, in [29] the authors introduce a continual learning framework for smartphones. Conversely, Chen et al. [17] introduce a reconfigurable array architecture to accelerate backpropagation through uniform memory access patterns, with $410\,\mathrm{mW}$ power consumption on FPGA. Nonetheless, such works address backpropagation-based training, which is generally unfit for extreme edge devices due to memory and computational requirements. Furthermore, backpropagation-based fine-tuning requires large amounts of labeled data, often unavailable in real-world settings.

In order to solve the extreme edge learning challenges, several on-device (few-shot) class-incremental learning implementations have been proposed. Hacene et al. [30] introduce a feature extractor whose features are compared with class anchor points, their architecture implemented in a $22\,\mathrm{W}$ FPGA. Lungu et al. [16] use Siamese Networks to compute the similarity measure between a query and class prototypes, with their FPGA implementation enabling class learning within $35\,\mathrm{ms}$. Neuromorphic chips have also been employed for continual [18] and few-shot [31] on-device learning, with energy requirements of $167\,\mathrm{mJ}$ per class in a data-scarce context. Instead, we propose a methodology enabling real-time inference and learning, with energy requirements as low as $12\,\mathrm{mJ}$.

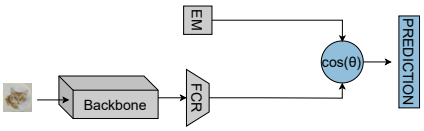
III. FSCIL task description

In FSCIL, a learner is progressively presented with new classes over a number of training sessions. The data stream is denoted as $D = \{D^t\}_{t=0}^{T}$, where t indicates the session index. The class labels $y_n^t \in C^t$ does not intersect across sessions, i.e., $\forall i \neq j, \mathcal{C}^i \cap \mathcal{C}^j = \emptyset$. During the base session, or session 0, the model is pretrained and metalearned to gain representation knowledge of the input sequence. Subsequently, in the online stage, the incremental sessions introduce N new classes, each with S samples per class, referred to as N-way, S-shot FSCIL. The model is evaluated on samples from all previous classes $\widetilde{C^t} := C^0 \cup C^1 \cdots \cup C^t$, ensuring that new classes are learned without forgetting previous ones.

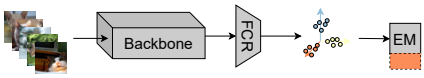
IV. Online Few-shot Class-incremental Learning

This section presents the first main contribution of the paper: we introduce O-FSCIL, which is comprised of a backbone, a Fully Connected Reductor (FCR), and an Explicit Memory (EM), shown in Fig. 1. The backbone ($f(\cdot)$) maps an input image (x) to an intermediate representation ($\theta_a \in \mathbb{R}^{d_a}$), and the FCR projects the representation to a lower-dimensional feature ($\theta_p \in \mathbb{R}^{d_p}$ where $d_p < d_a$). During inference, the feature vector (θ_p) is compared against all prototypes stored in the EM; the prototype with the highest cosine similarity indicates the final prediction (see Fig. 1a).

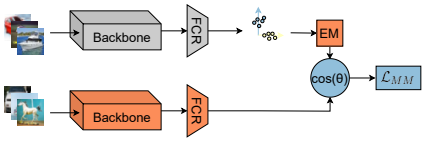
When learning a new class i, the prototype $(\bar{\theta}_p, i)$ is added to the EM through averaging all θ_p feature of class i samples, while the backbone and FCR remains frozen. This allows for online updates by passing labeled images through the model only once, without requiring expensive iterative (batched)



(a) Classifying a query in inference mode by comparing its features with class prototypes stored in EM.



(b) O-FSCIL update of EM with the class average of FCR-generated features. The backbone and the FCR are frozen.



(c) Server-side metalearning the model on the base session. The loss is used to update the backbone and the FCR.

Fig. 1: Inference (a), on-device learning a new class (b), and server-side metalearning (c) modes of O-FSCIL. Modules colored in orange are updated, while grey ones are frozen. During pretraining, we replaced the prototype computation and EM update from (b) with an FCR-like FCC classifier, with all three sections jointly trained.

TABLE I: Proposed backbones. Here we represent the convolutional stride per inverted residual block in MobileNetV2, as well as the dimensionality of the FCR features d_a and prototypical features d_p.

	MobileNetV2			**Resnet12**
		_x2	**_x4**	
CNN stride	1,2,2,2,1,2,1	1,2,2,2,1,1,1	1,2,2,1,1,1,1	-
d_a	1280	1280	1280	640
d_p	256	256	256	512
Params. [M]	2.5	2.5	2.5	12.9
MACs [M]	25.9	45.4	149.2	525.3

gradient updates. This online learning capability is acquired via pretaining and metalearning, which are performed before deploying the model using data from the base session.

A. Light-weight backbone

To enable FSCIL at the extreme edge, we replace the ResNet-12 backbone [5], [6] with a lightweight MobileNetV2 [21]. To handle low-resolution inputs (i.e., 32×32 in CIFAR100 [23]), we reduce the stride across the convolutional blocks of the seven inverted residual blocks, obtaining three models with different complexity. The proposed networks and their hardware costs are presented in Table I, together with the dimensions of the FCR features (d_a) and prototypical features (d_p).

B. Pretraining

The pretraining guides the backbone and FCR towards meaningful representations by solving a supervised classification problem on the base session (with $|C^0|$ classes). To this end, we modify the topology shown in Fig. 1b by replacing the EM with a Fully Connected Classifier (FCC), which computes the class probability of an input image. The backbone, the FCR, and the FCC are mutually trained to minimize the Cross-Entropy (CE) loss. To improve the accuracy of our system, we implement data augmentation methods enabling feature interpolation and a novel feature orthogonality regularization.

Feature interpolation: Apart from traditional data augmentation techniques (i.e., blur, horizontal flip, crop, and resize), we use inter-class feature interpolation using Mixup [19] and CutMix [20]. Thus, instead of generating multiple inputs from a single image, we combine two input images and create intermediate class labels. The two methods are employed exclusively, with a probability of 0.4.

Feature orthogonality: The pretraining of the architecture with the CE loss can reduce the representationality of the prototype features. The FCC layer reduces the dimension of θ_p from $\mathbb{R}^{d_p}$ to $\mathbb{R}^{|C^0|}$, where $|C^0|$ is the number of the base classes and $d_p > |C^0|$. The classification boundary of FCC only lives in the smaller base class hyperplane with dimension equal to $|C^0|$, ignoring perpendicular features. This impedes the generation of new feature clusters on orthogonal planes, hindering the model's ability to learn new classes.

We propose feature orthogonality regularization to address the dimensionality reduction problem. Instead of applying weight matrix orthogonal regularization [32], we orthogonalize the feature vectors and study its generalization capability to novel classes in the FSCIL scenario. Equation 1 shows the formulation of our orthogonality regularization:

$$\mathcal{L}_{ortho} = (\theta_{pb}^t \times \theta_{pb} - I_{d_f})^2, \tag{1}$$

where θ_{pb} is a batch of θ_p bundled as matrix with dimension $\mathbb{R}^{B \times d_p}$, and B is the batch size. The proposed pretraining loss includes the classification CE loss and the orthogonality loss, weighted by the λ_{ortho} regularization strength:

$$\mathcal{L}_{pre} = \mathcal{L}_{ce} + \lambda_{ortho} \cdot \mathcal{L}_{ortho}, \tag{2}$$

C. Metalearning

After pretraining, we perform offline metalearning to enhance feature clustering by emulating the learning and inference processes on the base session [33], as shown in Fig. 1c. We train the backbone and FCR for multiple iterations, re-computing the class prototypes from meta-samples, N randomly selected images per class. After generating the class prototypes, we compute the cosine similarity (cossim) between a query sample (x) and a class prototype ($\bar{\theta}_{p,i}$):

$$l_i = \mathrm{ReLU}(\mathrm{cossim}(FCR(f(x)), \bar{\theta}_{p,i}). \tag{3}$$

Note that the backbone-extracted features are employed for x, whereas the prototypes, generated by clustering the features extracted using the same backbone, are stored in the EM, as illustrated in Fig. 1c. Inspired by MANN [34], we use the ReLU sharpening function to induce quasi-orthogonality.

Although CE shows good performance in previous studies [5], [6], it draws the features towards the cluster's centroid without accounting for the confidence level. This can lead to overfitting regions, where high confidence points are prioritized. To this end, we employ a multi-margin loss defined as:

$$\mathcal{L}_{MM}(l, y) = \frac{\sum_i \max(0, (m - l_{gt} + l_i))^2}{|C^0|}, \tag{4}$$

where y is the ground truth label and m denotes margin value, which set to $m = 0.1$ after grid search. Multi-margin loss spreads the features near the classification frontier, improving the accuracy on distant points while maintaining its performance on those near the prototype.

V. Hardware deployment

A. Deployment

We deploy and evaluate our model on the GAP9 MCU, a multi-core Single Instruction Multiple Data (SIMD) processor suitable for vector and matrix computations, present in neural networks. GAP9 comprises two processing components, a fabric controller, handling control and communications, and a 9-core cluster designed to efficiently execute parallelized algorithms. All cores share access to an L1 memory and instruction cache, an L2 memory is shared by the compute processors, and the system also supports an external $8\,\mathrm{MB}$ L3 memory. Furthermore, Direct Memory Access (DMA) units enable asynchronous L1$\leftrightarrow$L2 and L2$\leftrightarrow$L3 memory transfers.

To deploy our model on GAP9, we first quantize the weights and activations of our metalearned Floating-Point (FP) model to 8-bit integers using Training Quantization Thresholds (TQT) algorithm in Quantlib [35], with additional quantization-aware pretraining and metalearning epochs. The network is then deployed using Dory [28], whereas an additional IO interface layer is implemented to support the evaluation of our O-FSCIL.

B. On-Device learning

During the deployment, the MCU performs online novel class learning, computing the corresponding class prototype and storing it to EM. For inference, a query image is classified according to the class prototype with the highest cosine similarity with the query θ_p feature. To increase the number of classes that can be learned on the device, we reduce the memory requirements of EM. We study precision reductions through bit-shift divisions in Section V-A.

Similar to Mode 2 in C-FSCIL [6], we implement an optional FCR finetuning, while freezing the backbone. To minimize the training effort, we store average class activations $\overline{\theta}_{a,i}$ in an activation memory. We then iteratively update the FCR by maximizing the similarity between the FCR mapping of $\overline{\theta}_{a,i}$ and the bipolarized class prototype through batched gradient descent over B iterations. To minimize the memory accesses, we develop a sub-batching mechanism that creates three input matrices from N pairs of $\overline{\theta}_{a,i}$, $\mathrm{FCR}(\overline{\theta}_{a,i})$, and $\overline{\theta}_{p,i}$. This allows for computing the accumulated gradient of N samples, reducing the number of memory accesses to B/N per batch.

VI. Experimental results

A. CIFAR100 benchmark

We use the CIFAR100 [23] dataset to evaluate our architecture, split into three sets: base session (50 images/class for 60 classes), class-incremental learning sessions (eight 5-way, 5-shot sessions) and 100 images per class for the test set.

As shown in Table II, O-FSCIL achieves state-of-the-art accuracy with ResNet-12 as a backbone, outperforming other works by more than one percentage point averaged over eight learning sessions. Notably, our pretraining and metalearning lead to an accuracy gain of 1.5% in the base session, paving the way for robust incremental learning in later stages. This further enables us to employ O-FSCIL on lightweight MobileNetv2 backbones, achieving average test accuracy levels of up to 66.54% for MobileNetV2_x4, outperformed only by 1% by the larger NC-FSCIL. Notably, the minimal accuracy reduction comes with a $5.7\times$ reduction in computational effort and a $5.2\times$ decrease in storage requirements compared to the ResNet12 backbone, as shown in Table I.

Remarkably, our metalearning strategy allows us to generate robust, separable sample projections without expensive retraining of the FCR. Nevertheless, if backpropagation-based finetuning is employed on MobileNetV2_x4, an additional 0.2% is to be gained, yet this would incur an adaptation cost of up to 6.6G MACs/session, compared to 0.7G MACs/session on MobileNetV2_x4 without any fine-tuning. The computational effort reduction of $8.8\times$ further motivates the extreme edge potential of our architecture.

B. Ablation study

This section investigates the benefits of our proposed pretraining and metalearning. As a baseline, we consider the ResNet-12-pretrained O-FSCIL architecture. Firstly, we notice that data augmentation generates accuracy gains of 2.15% on the base session and 2.4% on the eighth one. Second, adding orthogonal regularization in the pipeline significantly boosts performance, particularly for the new classes, leading to accuracy increments between 1.65% and 2.87%. This confirms that orthogonalization encourages neural networks to learn useful features beyond those of the base classes. Interestingly, CE metalearning incurs performance degradation. Corroborating this with CE loss reductions noticed during training without the accuracy classification gains, we conclude that CE discourages feature robustness and generalization.

C. Deployment

We quantize O-FSCIL using TQT [36], with three pretraining epochs and ten metalearning iterations following the quantization. As shown in Table II, similar accuracy levels are measured when comparing `int8`-quantized models with `fp32` networks. Interestingly, quantized networks achieve higher accuracy for the latter sessions, as reducing the precision acts as a regularizer, improving the features' separability.

We furthermore measured the latency, power, and energy consumption for our quantized and deployed models, presented in Table IV. We deploy our models on the GAP9 MCU, operating at $650\,\mathrm{mV}$, $240\,\mathrm{MHz}$ as this is the most energy-efficient operating point for the MCU. Notably, we can perform both inference and training in real-time, as O-FSCIL learns a new class only $256\,\mathrm{ms}$. Moreover, we remain within the $50\,\mathrm{mW}$ power envelope also for backpropagation-based FCR. By performing both EM update and last layer finetuning on MobileNetV2_x4, GAP9 draws up to $320\,\mathrm{mJ}$ per new class. Without the finetuning, it can efficiently learn new classes consuming $12\,\mathrm{mJ}$ with

TABLE II: O-FSCIL accuracy on CIFAR100. FP32 models were evaluated on NVIDIA GeForce GTX 1080 Ti, INT8 models were evaluated on GAP9 MCU. FT represents optional iterative FCR fine-tuning.

				Session accuracy [%]									
Method	Backbone	Prec.	Size [MB]	0	1	2	3	4	5	6	7	8	Avg.
MetaFSCIL [33]	ResNet20	FP32	1.08	74.50	70.10	66.84	62.77	59.48	56.52	54.36	52.56	49.97	60.79
C-FSCIL [6]	ResNet12	FP32	51.6	77.47	72.40	67.47	63.25	59.84	56.95	54.42	52.47	50.47	61.64
LIMIT [26]	ResNet20	FP32	1.08	73.81	72.09	67.87	63.89	60.70	57.77	55.67	53.52	51.23	61.84
SAVC [5]	ResNet12	FP32	51.6	78.47	72.86	68.31	64.00	60.96	58.28	56.17	53.91	51.63	62.73
ALICE [27]	ResNet18	FP32	66.8	79.00	70.50	67.10	63.40	61.20	59.20	58.10	56.30	54.10	63.21
NC-FSCIL [4]	ResNet12	FP32	51.6	82.52	76.82	73.34	69.68	66.19	62.85	60.96	59.02	56.11	67.50
O-FSCIL	ResNet12	FP32	51.6	**84.05**	**79.10**	74.23	69.96	66.92	63.89	61.67	59.51	57.10	68.52
O-FSCIL + FT		FP32	51.6	84.02	79.08	**74.34**	**70.11**	**66.95**	**64.00**	**61.86**	**59.72**	**57.50**	**68.62**
O-FSCIL	MobileNetV2	FP32	10.0	77.51	72.82	68.41	64.25	61.24	57.98	55.32	53.03	50.73	62.37
		INT8	2.5	76.97	72.46	68.24	64.19	61.07	58.14	56.01	53.55	51.37	62.44
O-FSCIL	MobileNetV2_x2	FP32	10.0	78.37	73.47	69.20	64.94	61.20	58.24	55.33	53.49	51.51	62.86
		INT8	2.5	78.13	73.54	69.27	65.13	61.85	58.73	56.21	54.04	51.81	63.19
O-FSCIL	MobileNetV2_x4	FP32	10.0	81.79	77.37	72.73	68.42	64.87	61.76	59.76	57.21	54.95	66.54
		INT8	2.5	81.56	76.81	72.56	68.32	64.95	61.94	59.65	57.47	55.33	66.51
O-FSCIL + FT	MobileNetV2_x4	FP32	10.0	81.90	77.31	72.90	68.48	65.09	61.91	59.73	57.72	55.45	66.75
		INT8	2.5	81.53	76.82	72.54	68.44	64.81	61.76	59.65	57.78	55.72	66.56

TABLE III: Ablation study of the proposed methods for the accuracy[%] on CIFAR100. AG: augmentation, OR: orthogonal regularization, MM: multi-margin-based metalearning, CE: cross-entropy-based metalearning, FT: incremental fine-tuning. The experiments were conducted with ResNet12 backbone.

AG	OR	MM	CE	FT	Session		Avg
					0	8	
					79.72	51.47	62.94
✓					81.87	53.85	64.77
✓	✓				83.52	56.72	67.88
✓		✓			83.65	56.25	67.56
✓	✓	✓			84.05	57.10	68.52
✓	✓		✓		83.02	51.54	64.56
✓	✓	✓		✓	84.02	57.50	68.62

TABLE IV: The execution time, power, and energy consumption on GAP9 for O-FSCIL – EM update, emphasized – and for FCR finetuning, added for comparison. The results are reported per class, for a five-shot learning setting. Finetuning is performed for 100 epochs. BB denotes the backbone.

Operation	BB	Time [ms]	Power [mW]	Energy [mJ]
FCR	∀	$3.23^{\pm 0.73}$	$47.75^{\pm 0.34}$	$0.15^{\pm 0.01}$
BB inference	M	$48.10^{\pm 5.14}$	$43.96^{\pm 0.98}$	$2.12^{\pm 0.23}$
	M2	$52.51^{\pm 5.27}$	$45.12^{\pm 0.24}$	$2.40^{\pm 0.24}$
	M4	$99.50^{\pm 2.41}$	$44.19^{\pm 0.64}$	$4.40^{\pm 0.12}$
EM update	M	$256.65^{\pm 11.6}$	$44.22^{\pm 0.84}$	$11.35^{\pm 0.24}$
	M2	$278.70^{\pm 11.9}$	$45.75^{\pm 0.26}$	$12.75^{\pm 0.25}$
	M4	$513.65^{\pm 5.63}$	$44.29^{\pm 0.59}$	$22.75^{\pm 0.12}$
FCR finetune	M	$6171.7^{\pm 29.8}$	$50.29^{\pm 0.55}$	$310.35^{\pm 0.72}$
	M2	$6193.7^{\pm 29.9}$	$50.33^{\pm 0.52}$	$311.75^{\pm 0.74}$
	M4	$6428.7^{\pm 28.0}$	$50.05^{\pm 0.54}$	$321.75^{\pm 0.58}$

minor degradation, demonstrating the feasibility of O-FSCIL for battery-operated devices.

Fig. 2 illustrates the impact of the multi-core architecture of GAP9 on O-FSCIL. Highly parallelizable given the presence of convolutional layers, we can achieve up to 6.5 MACs/cycle for our largest MobileNetV2 backbone with 8 cores. The parallelization potential reduces as we increase the number of strided convolutions in the feature extractor. The performance

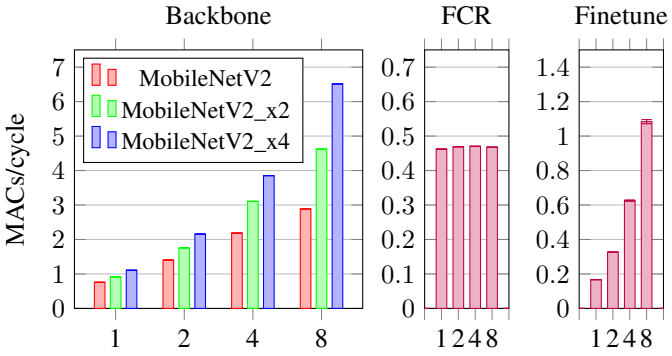


Fig. 2: Average number of operations per cycle given the number of active cores, for backbone inference (left), FCR inference (centre), and FCR backpropagation update (right).

gains of parallelizing the FCR layer on a multi-core architecture are not as pronounced as those for other layers, primarily due to the data transfer overheads. Specifically, transferring approximately 328 kB of data from L3 to the L1 caches incurs a latency of roughly 3 ms. In contrast, when parallelized, the actual computation consumes only about 0.25 ms, showcasing a significant speedup of nearly 5 times on the multi-core system.

To reduce the memory requirements, we analyze the impact of the memory precision $\overline{\theta}_{p,i}$ on the accuracy, shown in Fig. 3. A 17-bit integer is sufficient to represent the class prototype without overflow for MobileNetV2_x4. We can further reduce the $\overline{\theta}_{p,j}$ bit length to an 8-bit integer by performing a 9-bit right shift (i.e., vector division), reducing the norm while maintaining the general $\overline{\theta}_{p,j}$ vector direction and preserving the accuracy. Further reductions down to 3-bit class prototypes can be achieved without accuracy drops, thus enabling us to store 100 class prototypes with only 9.6 kB.

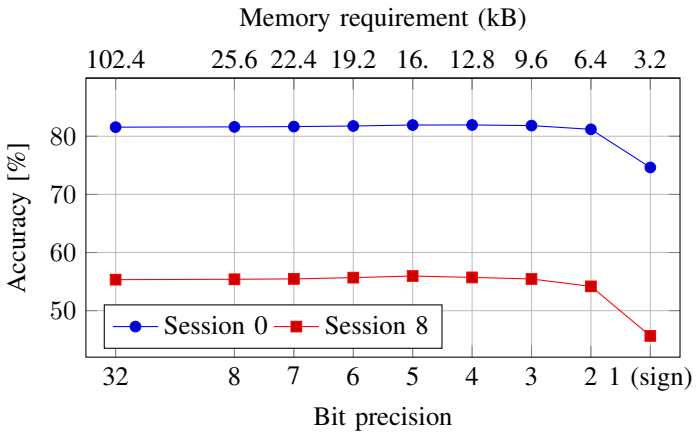


Fig. 3: The representation precision in the episodic memory impacts the accuracy of MobileNetV2_x4-based model, considering 100 class prototypes stored in the memory.

VII. Conclusion

We proposed O-FSCIL, a novel FSCIL methodology using orthogonal regularization and multi-margin-based metalearning to improve feature separability in incoming classes. We achieved state-of-the-art accuracy on the CIFAR100 dataset using ResNet12 and comparable results with NC-FSCIL when using the $5\times$ smaller and $3\times$ more computationally efficient MobileNetV2_x4. We moreover designed, deployed, and evaluated O-FSCIL on GAP9 MCU, with energy requirements to learn a new class as low as $12\,\mathrm{m\,J}$, making it suitable for battery-operated extreme edge devices.

VIII. Acknowledgements

This work was partly supported by the Swiss National Science Foundation under grant No 207913: TinyTrainer: On-chip Training for TinyML devices.

References

[1] X. Tao *et al.*, "Few-shot class-incremental learning," in *Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 12 183–12 192.

[2] C. Zhang *et al.*, "Few-shot incremental learning with continually evolved classifiers," in *Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021, pp. 12 455–12 464.

[3] K. Zhu *et al.*, "Self-promoted prototype refinement for few-shot class-incremental learning," in *Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021, pp. 6801–6810.

[4] Y. Yang *et al.*, "Neural collapse inspired feature-classifier alignment for few-shot class-incremental learning," in *The Eleventh International Conference on Learning Representations (ICLR)*, 2023.

[5] Z. Song *et al.*, "Learning with fantasy: Semantic-aware virtual contrastive constraint for few-shot class-incremental learning," in *Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023, pp. 24 183–24 192.

[6] M. Hersche *et al.*, "Constrained few-shot class-incremental learning," in *Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022, pp. 9057–9067.

[7] G. Karunaratne *et al.*, "In-memory realization of in-situ few-shot continual learning with a dynamically evolving explicit memory," in *IEEE 48th European Solid State Circuits Conference (ESSCIRC)*, 2022, pp. 105–108.

[8] K. He *et al.*, "Deep residual learning for image recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770–778.

[9] S. Kumar *et al.*, "Internet of Things is a revolutionary approach for future technology enhancement: a review," *Journal of Big Data*, vol. 6, no. 1, p. 111, 2019.

[10] C. R. Banbury *et al.*, "Benchmarking tinyml systems: Challenges and direction," *arXiv preprint arXiv:2003.04821*, 2020.

[11] H. Ren *et al.*, "Tinyol: Tinyml with online-learning on microcontrollers," in *2021 International Joint Conference on Neural Networks (IJCNN)*, 2021, pp. 1–8.

[12] D. Nadalini *et al.*, "PULP-TrainLib: Enabling On-Device Training For RISC-V Multi-Core MCUs Through Performance-Driven Autotuning," in *Embedded Computer Systems: Architectures, Modeling, and Simulation*. Springer International Publishing, 2022, p. 200–216.

[13] J. Lin *et al.*, "On-device training under 256kb memory," in *Annual Conference on Neural Information Processing Systems (NeurIPS)*, 2022.

[14] C. Cioflan *et al.*, "Towards on-device domain adaptation for noise-robust keyword spotting," in *2022 IEEE 4th International Conference on Artificial Intelligence Circuits and Systems (AICAS)*, 2022, pp. 82–85.

[15] L. Ravaglia *et al.*, "A tinyml platform for on-device continual learning with quantized latent replays," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems (JETCAS)*, vol. 11, no. 4, pp. 789–802, 2021.

[16] I. A. Lungu *et al.*, "Siamese networks for few-shot learning on edge embedded devices," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems (JETCAS)*, vol. 10, no. 4, pp. 488–497, 2020.

[17] X. Chen *et al.*, "Eile: Efficient incremental learning on the edge," in *2021 IEEE 3rd International Conference on Artificial Intelligence Circuits and Systems (AICAS)*, 2021, pp. 1–4.

[18] E. Hajizada *et al.*, "Interactive continual learning for robots: A neuromorphic approach," in *Proc. of the International Conference on Neuromorphic Systems (ICONS)*. New York, NY, USA: ACM, 2022.

[19] H. Zhang *et al.*, "mixup: Beyond Empirical Risk Minimization," in *International Conference on Learning Representations (ICLR)*, 2018.

[20] S. Yun *et al.*, "Cutmix: Regularization strategy to train strong classifiers with localizable features," in *Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 6023–6032.

[21] M. Sandler *et al.*, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 4510–4520.

[22] "Gap9 product brief," https://greenwaves-technologies.com/wp-content/uploads/2023/02/GAP9-Product-Brief-V1_14_non_NDA.pdf, accessed: 2023-08-10.

[23] A. Krizhevsky, "Learning multiple layers of features from tiny images," *University of Toronto*, 2009.

[24] S. Dong *et al.*, "Few-shot class-incremental learning via relation knowledge distillation," in *Proc. of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 2, 2021, pp. 1255–1263.

[25] H. Zhao *et al.*, "Mgsvf: Multi-grained slow vs. fast framework for few-shot class-incremental learning," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021.

[26] D.-W. Zhou *et al.*, "Few-shot class-incremental learning by sampling multi-phase tasks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022.

[27] C. Peng *et al.*, "Few-shot class-incremental learning from an open-set perspective," in *European Conference on Computer Vision (ECCV)*. Springer, 2022, pp. 382–397.

[28] A. Burrello *et al.*, "Dory: Automatic end-to-end deployment of real-world dnns on low-cost iot mcus," *IEEE Transactions on Computers*, 2021.

[29] L. Pellegrini *et al.*, "Continual learning at the edge: Real-time training on smartphone devices," *arXiv preprint arXiv:2105.13127*, 2021.

[30] G. B. Hacene *et al.*, "Incremental learning on chip," in *2017 IEEE Global Conference on Signal and Information Processing (GlobalSIP)*, 2017, pp. 789–792.

[31] K. Stewart *et al.*, "Online few-shot gesture learning on a neuromorphic processor," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 10, pp. 512–521, 2020.

[32] K. Ranasinghe *et al.*, "Orthogonal projection loss," in *Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021, pp. 12 333–12 343.

[33] Z. Chi *et al.*, "Metafscil: A meta-learning approach for few-shot class incremental learning," in *Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022, pp. 14 166–14 175.

[34] G. Karunaratne *et al.*, "Robust high-dimensional memory-augmented neural networks," *Nature communications*, vol. 12, no. 1, p. 2468, 2021.

[35] M. Spallanzani *et al.*, "Quantlab: a modular framework for training and deploying mixed-precision nns," March 2022.

[36] S. Jain *et al.*, "Trained quantization thresholds for accurate and efficient fixed-point inference of deep neural networks," *Proc. of Machine Learning and Systems*, vol. 2, pp. 112–128, 2020.