Filter-based Adaptive Model Pruning for Efficient Incremental Learning on Edge Devices

Jing-Jia Hung*, Yi-Jung Chen†, Hsiang-Yun Cheng‡, Hsu Kao¶, Chia-Lin Yang*§
* Department of Computer Science and Information Engineering, National Taiwan University, Taipei, Taiwan
† Department of Computer Science and Information Engineering, National Chi Nan University, Nantou, Taiwan
‡ Research Center for Information Technology Innovation, Academia Sinica, Taipei, Taiwan
¶ Department of Engineering and System Science, National Tsing Hua University, Hsinchu, Taiwan
*{r10944050, yangc}@csie.ntu.edu.tw, † yjchen@ncnu.edu.tw, ‡ hycheng@citi.sinica.edu.tw, ¶ gosh@gapp.nthu.edu.tw

***Abstract*—Incremental Learning (IL) enhances Machine Learning (ML) models over time with new data, ideal for edge devices at the forefront of data collection. However, executing IL on edges faces challenges due to limited resources. Common methods involve IL followed by model pruning or specialized IL methods for edges. However, the former increases training time due to fine-tuning and compromises accuracy for past classes due to limited retained samples or features. Meanwhile, existing edge-specific IL methods utilize weight pruning, which requires specialized hardware or compilers to speed up and cannot reduce computations on general embedded platforms. In this paper, we propose Filter-based Adaptive Model Pruning (FAMP), the first pruning method designed specifically for IL. FAMP prunes the model before the IL process, allowing fine-tuning to occur concurrently with IL, thereby avoiding extended training time. To maintain high accuracy for both new and past data classes, FAMP adapts the compressed model based on observed data classes and retains filter settings from the previous IL iteration to mitigate forgetting. Across all tests, FAMP achieves the best average accuracy, with only a 2.78% accuracy drop over full ML models with IL. Moreover, unlike the common methods that prolong training time, FAMP takes 35% shorter training time on average than using the full ML models for IL.**

I. Introduction

Incremental Learning (IL) is a training paradigm that continues evolving Machine Learning (ML) models based on existing knowledge, adapting model settings when encountering new data classes. IL is particularly suitable for edge devices, often at the forefront of data reception. Since edges commonly have limited computing and memory resources, ML models need pruning or downsizing to perform IL on edges. However, naively pruning models after the IL process results in lengthy training latency and reduced accuracy.

In class-incremental learning, a widely discussed IL paradigm [22], [35], new data classes arrive periodically, prompting the IL process to enhance the ML model's knowledge. Each batch of arriving data classes forms an IL task. With class IL, computational requirements and memory footprint must remain bounded regardless of the number of classes encountered so far [26]. However, this constraint leads to the challenge of *catastrophic forgetting*, where past knowledge is easily lost while learning new information due to the lack of data from past classes. Catastrophic forgetting would be even worse with model pruning. When conducting model pruning, fine-tuning is essential to regain accuracy. Fine-tuning can be achieved by retraining the pruned model with the complete dataset [8], [13], [23] or employing knowledge distillation with partial data samples [14], [32], [33]. Given the data-limited nature of class IL, knowledge distillation is more suitable than retraining. However, our experiments reveal that, compared to performing IL on a full ML model, employing model pruning with knowledge distillation after the IL process still results in an average of 13.55% accuracy drop and 39% longer training time across our test cases. To solve this, SparCL [34] proposes adaptive weight masking for IL tasks. However, the unstructured weight reduction fails to effectively reduce computations compared to structured filter pruning [7], [13].

To understand the issues of naively applying pruning with class IL, we perform experiments with several scenarios, e.g. pruning after IL, and SparCL [34], to analyze their accuracy and performance. Our analysis identifies key properties crucial for designing a model compression method tailored for class IL, aiming for high accuracy without extending training time: (a) adapting the compressed model structure for new classes, (b) preserving learned knowledge from prior IL tasks to prevent catastrophic forgetting, (c) model pruning before IL, enabling concurrent fine-tuning with IL to expedite training, and (d) structured pruning, i.e. filter-based pruning, for efficient computation reduction. Details are presented in Sec. II-B.

In this paper, we propose Filter-based Adaptive Model Pruning (FAMP), the first structured model compression method tailored for class IL. To avoid prolonging training time and maintain accuracy comparable to IL without compression, for each IL task, FAMP selects the compressed model structure before IL training begins, allowing for fine-tuning during the IL process. To adapt the model structure to IL tasks observed so far, FAMP conducts class-aware model selection using new class data and the original full model. Subsequently, to preserve past-class knowledge, FAMP unites the compressed model with the one derived from the previous IL task. To manage potential post-union size increases, FAMP utilizes an accuracy recovery estimation method [32] to prune layers, accelerating the attainment of the target model size while optimizing recoverable accuracy drops. The key contributions of this paper include:

- We propose FAMP, the first structured pruning framework tailored for IL. Compared to using the full model for IL

§Corresponding author.

978-3-9826741-0-0/DATE25/© 2025 EDAA

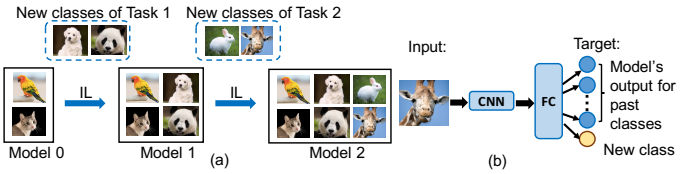

Fig. 1. IL overview: (a) IL paradigm and (b) ML model after an IL task.

tasks, FAMP achieves an average accuracy drop of just 2.78% while reducing training time by 35%.

- We thoroughly explore all possible scenarios of applying IL on edges, and provide a detailed analysis of their accuracy and performance in Sec. II-B.
- FAMP is not limited to and can work with existing IL methods, as demonstrated through compatibility analysis experiments in Sec. V-D.
- An ablation study is conducted to evaluate how various designs of FAMP's key components affect its overall effectiveness.
- We evaluate the computation time and memory requirement of various methods for deploying ML models with IL to edges. FAMP excels in accuracy and greatly reduces training time, well-suited for resource-limited devices.

II. Background and Motivation

A. Basics of IL and Model Pruning

Incremental Learning. In class-incremental learning, as depicted in Fig. 1(a), tasks arrive periodically, each consisting of one or more data classes, triggering the IL process. Following the IL process, new classification heads are appended to the model, as shown in Fig. 1(b). Catastrophic forgetting, discussed in Sec. I, poses a significant challenge, prompting the development of regularization-based, rehearsal-based, and bias correction methods to address it. Regularization-based methods use regularization terms with classification loss to retain past knowledge [18], [31], while rehearsal-based methods retain a subset of past class exemplars or features for subsequent IL tasks [26], [31]. Bias correction methods aim to rectify class bias that may arise after several IL tasks [1], [31]. A Modern trend in IL design synergizes these methods, e.g. Foster [31].

Model Pruning. Existing model pruning methods can be categorized into fine-grained weight pruning or unstructured pruning [2]–[4], [20], [30], and coarse-grained or structured pruning [8], [13], [21], [23], [29]. Unstructured pruning [3], [20] eliminates individual connections, achieving high compression rates with minimal accuracy loss. However, it requires specialized hardware or libraries to reduce computations [7], [13]. Structured pruning removes filters or layers, estimating filter importance via weight magnitudes [6], [13], [33] or gradients [23], [25]. Compared to unstructured pruning, structured pruning effectively reduces computations, making it more suitable for edges [7], [33]. After pruning, the compressed model should be fine-tuned by retraining with the entire dataset [8], [13], [23], or knowledge distillation with small datasets [14], [32], [33]. Considering the computational constraints of edge devices, SparCL [34] proposes mask-based weight-level unstructured pruning, training both weights and masks during IL. SparCL suggests Patdnn [24] for compiling

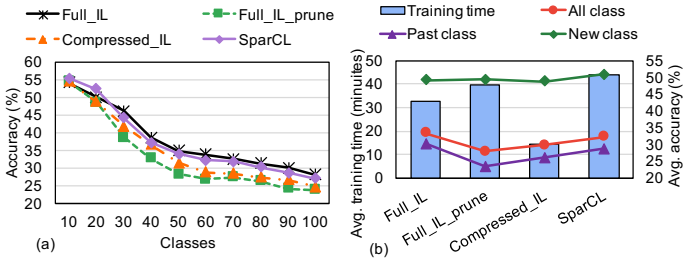

Fig. 2. Motivation experiments of ResNet18 on Tiny-ImageNet-100: (a) Accuracy curve across IL tasks and (b) average training time of each IL task and average accuracy of all, past, and new classes.

pruned models to accelerate, but lacks specific pruning patterns required by Patdnn, limiting its effectiveness on edge devices.

Execution Scenario. While each IL task handles less data than traditional ML training, the computational demands remain taxing for edge devices. One solution is to perform IL and model compression on a server and deploy the compressed model to edges for data collection and inference (Server-IL). Yet, this server-side approach risks the interception of data and model parameter. Due to security concerns, conducting IL on edges (Edge-IL) is preferred, but computations and memory usages must be constrained to suit the target edge device.

B. Motivation

To study the issues of existing solutions for deploying IL on edges, we implement the following methods.

- **Full_IL_prune**: For each IL task, the full model is first trained for the task, followed by downsizing and fine-tuning. While straightforward, performing IL and model compression on a server is necessary due to its high computational cost.
- **Compressed_IL**: Another direct method is to prune the full model before the first IL task and deploy the compressed model to the edge for subsequent tasks. Using only the compressed model, edge-based IL becomes feasible.
- **SparCL**: The method proposed in [34].
- **Full_IL**: The full model is used for IL without compression, representing the accuracy upper-bound.

For Full_IL_prune and Compressed_IL, we use Foster [31] as the IL method and apply gradient-based structured pruning with knowledge distillation [23], [28], [32] for model compression. In Fig. 2, we present accuracy results using ResNet18 [5] with the Tiny-ImageNet dataset [11], an 80% pruning ratio, and pre-training with 10 classes. Each IL task includes 10 classes. We also report the average training time of each IL task on a low-power edge platform, NVIDIA Jetson AGX Xavier. As depicted in Fig. 2(a), SparCL closely matches full_IL's accuracy, occasionally surpassing it during pre-training. According to [17], pruning can act as a regularizer during model training, improving performance compared to the original uncompressed models. However, SparCL requires an average of 35.59% more training time than Full_IL, as shown in Fig. 2(b), and an average of 1.33x more epochs to determine the mask settings.

Full_IL_prune and Compressed_IL exhibit average accuracy drops of 5.45% and 3.59%, respectively, compared to Full_IL. Notably, Full_IL_prune, which adjusts the compressed model for each IL iteration, performs 1.86% worse on average than

TABLE I
COMPARISON WITH OTHER METHODS.

	Full_IL_prune	Compressed_IL	SparCL	FAMP
Structured pruning	✓	✓		✓
Pruning before IL		✓		✓
Adaptively prune	✓		✓	✓
Retain past classes' value			✓	✓

Compressed_IL, where the model remains unchanged. Studies [15], [19] indicate that the selection of model structures affects accuracy. We compare adaptive and fixed compressed model structures, isolating the accuracy impact of the structure alone, without relying on previously learned values for new IL tasks. Results show adaptive structures achieve 0.2% to 2.68% higher accuracy. However, for Full_IL_prune, the benefit of the adaptive structure is not clear, as both compressed model selection and fine-tuning rely on the limited samples from past and new classes, potentially biasing the model toward newly arrived classes. Full_IL_prune achieves 2.51% lower accuracy in past classes compared to Compressed_IL, which accumulates knowledge based on an unchanged and compressed model.

Based on the above observations, an effective structured pruning method for IL should: (a) prune before IL to allow fine-tuning alongside IL, (b) adapt the compressed model structure to accommodate new classes, and (c) retain trained knowledge, such as weight values from previous IL iterations, to prevent catastrophic forgetting. As listed in Table I, none of the existing solutions meets all these properties.

III. Problem Modeling and Formulation

This section presents the modeling and formulation of the problem that FAMP solves. In class IL, a sequence of T tasks arrives periodically. Each task T_i has its dataset D_i, composed of one or more classes of data. We denote the set of task datasets as $D = \{D_{T_1}, D_{T_2}, \ldots, D_{T_{|T|}}\}$, where each task has distinct and non-overlapping classes. We use a rehearsal buffer E that keeps $|E|$ exemplars from past tasks, i.e., only $|E|/t$ exemplars can be kept for each of the t tasks seen so far. We use the exemplar selection method proposed in [26] and [31]. In a conventional IL paradigm, the full ML model at the i-th task, denoted as FM_{T_i}, is obtained by training the model from the previous task, $FM_{T_{i-1}}$, using a dataset composed of D_{T_i} and E.

An ML model M is composed of L layers, and each layer l_i, where $1 < i < |L|$, contains a set of filters $F_{l_i} = \{f_1^{l_i}, f_2^{l_i}, \ldots, f_{n_{l_i}}^{l_i}\}$, where n_{l_i} is the number of filters in layer l_i. When performing structured model pruning on the model M, the goal is to reduce the number of filters in the model while the loss, denoted by $\mathcal{L}$, is minimized.

FAMP aims to find a structured compressed model for each IL task such that the loss is minimized, and can be formulated as follows. For IL task T_i, FAMP takes D_{T_i}, E, the full ML model without any IL training, denoted as FM_0, and the compressed model after the previous IL task training, denoted as $CM_{T_{i-1}}$ as inputs to get a compressed model CM_{T_i}:

$$CM_{T_i} = FAMP(D_{T_i}; E; FM_0; CM_{T_{i-1}}) \quad (1)$$

The goal is to minimize the loss $\mathcal{L}$ while the target pruning ratio r is met:

$$\min_{W} \mathcal{L}(F; D_{T_i}; E) \text{ s.t. } \text{Card}(F) \leq 1 - r \quad (2)$$

where $\text{Card}(\cdot)$ calculates the cardinality of F.

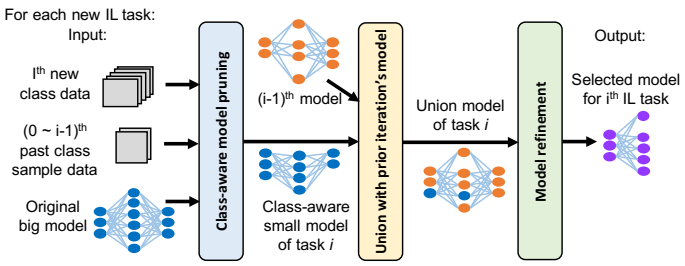

Fig. 3. Flow of the FAMP method for each new IL task.

IV. Filter-based Adaptive Model Pruning

We propose Filter-based Adaptive Model Pruning (FAMP), depicted in Fig. 3, to find a structured compressed model that achieves comparable accuracy and shorter training time compared to a full model with IL. FAMP satisfies all the properties listed in Table I. When an IL task arrives, FAMP initially performs *class-aware model pruning* using new-class data, past-class exemplars, and the full model without any IL, resulting in a compressed model suitable for both new and past classes. FAMP then *unites* this compressed model with the one from the previous IL task to retain learned knowledge. Since the model size may be larger than the target size after the union step, FAMP performs *model refinement*, downsizing the united model through layer pruning while maintaining accuracy. Finally, FAMP uses the structured and compressed model to preserve knowledge from previous tasks and adapt to new classes during the IL process for new-class training and fine-tuning. Algorithm 1 shows the pseudo-code of FAMP. For each IL task i, the training dataset is defined in line 2, which includes new-class data and past-class exemplars. Subsequently, class-aware pruning, union, and model refinement are executed in lines 3 to 5. The details of these three major steps are outlined in Section IV-A, Section IV-B, and Section IV-C, respectively.

Algorithm 1 FAMP

Require: FM_0, E, D, # of IL tasks T, pruning ratio r
Ensure: The compressed model CM_i
1: **for** $i = 1, ..., T$ **do**
2: $\quad \bar{D}_i \leftarrow E \cup D_i$
3: $\quad CM_i^{init} \leftarrow$ CLASS-AWARE PRUNING(FM_0, D_i, E, r)
4: $\quad CM_i^U \leftarrow$ UNION(CM_{i-1}, CM_i^{init})
5: $\quad CM_i \leftarrow$ MODEL REFINEMENT(CM_i^U, $\bar{D}_i$, r)
6: **end for**

A. Class-aware pruning

The goal of class-aware pruning is to select a compressed model structure suitable for both new and past classes by identifying filters important to both. Based on SNIP [12], filter importance is gauged by assessing each filter's contribution to the overall loss through weight and gradient multiplication. To mitigate bias towards new classes, given the disparity in past-class exemplars versus new-class data, we adjust the filter importance calculation proposed in [12]. The importance of filter f_i, denoted as I_{f_i}, is modeled as follows:

$$I_{f_i} = \alpha * g_{f_i}(D_i) * m_{f_i} + (1 - \alpha) * g_{f_i}(E) * m_{f_i}, \quad (3)$$

where m_{f_i} and g_{f_i} represent the weights and gradients of filter f_i, respectively, and α regulates the influence of both new-class data and past-class exemplars. Algorithm 2 outlines the pseudo-code of class-aware pruning. Using the full model without any

IL training, denoted as FM_0, along with new-class data D_i and past-class exemplars E, class-aware pruning computes the importance of each filter based on Eq. (3) (lines 2-4), and then prunes the filters with the least importance.

Algorithm 2 Class-Aware Pruning

Require: Full model without IL FM_0, exemplars of past classes E, new-class data D_i, pruning ratio r
Ensure: The compressed model $CM_{T_i}^{init}$
1: $CM_{T_i}^{init} \leftarrow FM_0$
2: $F \leftarrow$ number of filters in $CM_{T_i}^{init}$
3: **for** $f = 1, ..., F$ **do**
4: Compute I_f with D_{T_i} and E, Eq. (3)
5: **end for**
6: SortDescending(I)
7: $CM_{T_i}^{init} \leftarrow$ Prune the top $r * F$ filters with least I_f

Algorithm 3 Union

Require: The model CM_i^{init} and CM_{i-1}
Ensure: The union model CM_i^U
1: $CM_i^U \leftarrow CM_i^{init} \cup CM_{i-1}$
2: $F \leftarrow$ # of filters in $CM_{T_i}^U$
3: **for** $i = 1, ..., F$ **do** % set filter values
4: **if** $\exists w_i^{CM_{i-1}}$ **then**
5: $w_i^{CM_i^U} \leftarrow w_i^{CM_{i-1}}$
6: **else**
7: $w_i^{CM_i^U} \leftarrow w_i^{CM_i^{init}}$
8: **end if**
9: **end for**

Algorithm 4 Model Refinement

Require: Union model CM_i^U, training data set $\bar{D}_i$, pruning ratio r
Ensure: The compressed model CM_i
1: $L \leftarrow$ number of layers in CM_i^U
2: $k \leftarrow (1-r) * size(CM_i^U)$
3: $CM_i \leftarrow CM_i^U$
4: **for** $i = 1, ..., L$ **do**
5: Compute R_{l_i} with $\bar{D}_i$, Eq. (4)
6: $size_{prune} \leftarrow size(CM_i) - k$
7: $S_{l_i} \leftarrow {R_{l_i}}^{\alpha} * (size_{l_i} - size_{prune})^{\beta} / size_{l_i}$
8: **end for**
9: **while** $size(CM_i) \geq k$ **do**
10: SortDescending(S)
11: Drop the layer with the minimum S_l
12: $size_{prune} \leftarrow size(CM_i) - k$
13: **for** $i = 1, ..., L$ **do**
14: $S_{l_i} \leftarrow {R_{l_i}}^{\alpha} * (size_{l_i} - size_{prune})^{\beta} / size_{l_i}$
15: **end for**
16: **end while**

B. Union

To preserve knowledge from previous IL tasks, for a task n, as illustrated in Fig. 4, FAMP unites the compressed model from class-aware pruning with the one trained by the previous IL task $n-1$, yielding a united model $CM_{T_n}^U$. For filter values, shared filters between the compressed model of task $n-1$ and the current one are set with values from task $n-1$, while the remaining filters retain their values from the original model. The pseudo-code of the union step is outlined in Algorithm 3.

C. Model Refinement

Following the union step, the model size may exceed the target size. One approach is to reapply class-aware pruning, but the post-union model contains filters vital to both new and past knowledge. In this step, we need to determine which set of filters to prune that will cause the least accuracy drop for all classes observed so far. We adopt the "recoverability" concept from [32], which aims to optimize the latency-accuracy trade-off. Recoverability refers to the amount of accuracy drop

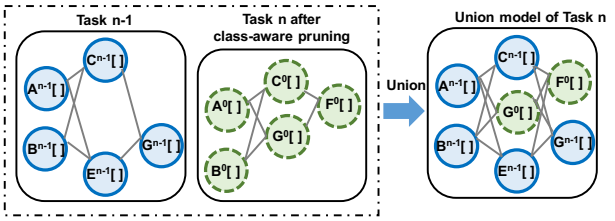

Fig. 4. Union method in FAMP.

after pruning and fine-tuning, estimated by training the model with removed candidate filters over a few batches. However, estimating recoverability for each filter is time-consuming. Thus, Wang et al. [32] propose layer-level pruning, inserting adaptors in adjacent layers to the dropped layer. These adaptors are briefly trained to measure accuracy recovery potential. While applying layer-pruning at the refinement step may be drastic, our ablation study in Section V-C shows that using layer-level refinement with recoverability analysis yields better results than using class-aware pruning.

The method proposed in [32] prioritizes model acceleration over size reduction, using a score function to assign high priority to layers exhibiting both good recoverability and acceleration results for pruning. However, this may lead to significant accuracy drops for our united model due to excessive pruning. Therefore, we redesign the score function to prioritize layers that achieves the target model size post-pruning while also demonstrating good recoverability. The recoverability R_{li} [32] and the score S_{l_i} of pruning layer l_i are modeled as follows.

$$R_{l_i} = min \ \|UM_n^{\theta} - UM_n^{\theta \setminus l_i}\|^2 \tag{4}$$

$$S_{l_i} = \frac{{R_{l_i}}^{\alpha} * (size_{l_i} - size_{prune})^{\beta}}{size_{l_i}} \tag{5}$$

UM_n^{θ} denotes the parameters θ in the united model UM_n of task n, and $\setminus l_i$ represents the exclusion of parameters in layer l_i. Coefficients α and β regulate the impact of recoverability and layer size. In Eq. (4) and Eq. (5), smaller values denote better recoverability and higher priority for pruning. The score function allows layers closer in size to the required pruning target to exhibit smaller differences, facilitating easier pruning.

Algorithm 4 presents the refinement step. We begin by computing recoverability and scores for each layer (line 4-7). Then, we prune the layer with the lowest score, updating the score after each layer is pruned until reaching the desired size (line 8-13). To minimize computation, we calculate recoverability once at the start. The ablation study in Section V-C shows that calculating recoverability at the start and recalculating for every layer removal yield comparable accuracy results.

V. Experiments

A. Experiment Setup

Datasets. We evaluate FAMP using two representative IL benchmarks, CIFAR-100 [10] and Tiny-ImageNet-100 [11]. Both benchmarks follow the widely used protocol of incrementally training 10 classes per task, with a rehearsal buffer of 2000 exemplars maintained via a herding-based method [26].
Models. We use three models commonly employed in IL and pruning studies [31]–[33]: ResNet18 [5], DenseNet121 [9], and MobileNetV2 [27]. These models vary in size and layer connection densities, with MobileNetV2 being the smallest and DenseNet121 featuring densely connected layers.

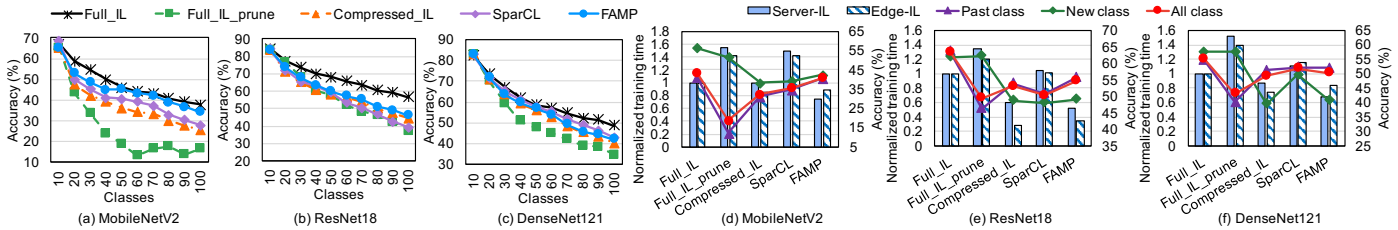

Fig. 5. Main results on CIFAR-100: classification accuracy curve for (a) MobileNetV2, (b) ResNet18, and (c) DenseNet121; normalized training time and average accuracy of all, past, and new classes for (d) MonbileNetV2, (e) ResNet18, and (f) DesnseNet121.

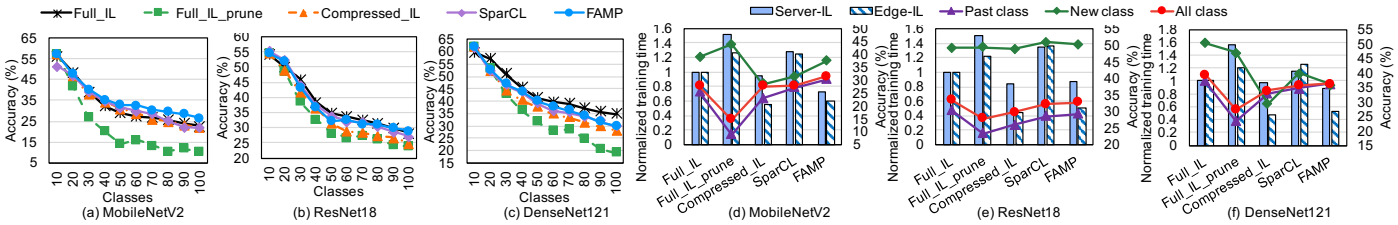

Fig. 6. Main results on Tiny-ImageNet-100: classification accuracy curve for (a) MobileNetV2, (b) ResNet18, and (c) DenseNet121; normalized training time and average accuracy of all, past, and new classes for (d) MonbileNetV2, (e) ResNet18, and (f) DesnseNet121.

TABLE II
AVERAGE ACCURACY DROP AND NORMALIZED TRAINING TIME OVER FULL_IL ACROSS ALL EVALUATED DATASETS AND MODELS.

	Full_IL	Full_IL_prune	Compressed_IL	SparCL	FAMP
Accuracy drop	0.00%	13.55%	6.12%	4.95%	2.78%
Normalized T_{train} (Server)	1	1.50	0.86	1.23	0.72
Normalized T_{train} (Edge)	1	1.28	0.52	1.24	0.58

Evaluation metrics and platforms. We present classification accuracy curves, showing the average accuracy of all learned classes, to assess IL quality, along with average accuracies for new, past, and all tasks. Performance is evaluated by measuring training time. For Edge-IL, training is run on a widely used low-power edge platform, NVIDIA Jetson AGX Xavier, while for Server-IL, training is done on an NVIDIA Tesla P40.

Comparison methods. We compare FAMP with two methods, Full_IL_prune and Compressed_IL, both introduced in Section II-B, as well as SparCL [34], a SOTA IL method designed for edge devices. Additionally, we evaluate Full_IL, which uses the full model for IL, serving as the accuracy upper bound.

Implementation details. We train all models using stochastic gradient descent with a momentum of 0.9 and a learning rate of 0.1. For CIFAR-100, MobileNetV2 is trained for 160 epochs, ResNet18 for 60 epochs, and DenseNet121 for 100 epochs. For Tiny-ImageNet-100, the same models are trained for 80, 60, and 80 epochs, respectively. The pruning ratio is set to the highest level that ensures less than a 2% accuracy drop compared to the full ML model without IL training. Specifically, 90%, 80%, and 90% of filters are pruned for MobileNetV2, ResNet18, and DenseNet121, respectively.

B. Main Results

Table II summarizes the average results across all evaluated cases, showing that FAMP achieves a superior balance between accuracy and training efficiency. It generally outperforms intuitive and prior methods, with only a 2.78% accuracy drop against Full_IL, while greatly reducing training time — requiring just 65% of Full_IL's training time on average across all IL execution scenarios (72% in Server-IL and 58% in Edge-IL).

Classification accuracy. Fig. 5 and Fig. 6 show the accuracy curves ((a)-(c)), as well as the average accuracy of all, past, and new classes ((d)-(f)). FAMP achieves an average accuracy drop of just 2.78% compared to Full_IL and even surpasses Full_IL by 3.69% for MobileNetV2 on Tiny-ImageNet-100. This stems from FAMP's effective model adaptation and past knowledge retention, ensuring that crucial filters are well-trained.

In most cases, FAMP surpasses intuitive methods like Full_IL_prune and Compressed_IL, while achieving accuracy on par with SparCL. FAMP achieves a significant all-class accuracy gain (avg. 10.78%) over Full_IL_prune, driven by a substantial increase in past-class accuracy (avg. 14.91%). This is attributed to FAMP's union and refinement steps, which effectively retain crucial filters and mitigate catastrophic forgetting. The gain is especially notable for MobileNetV2 (avg. 19.34%), where Full_IL_prune struggles to retain past knowledge within the smaller model. Compared to Compressed_IL, FAMP improves accuracy by 3.35% on average, driven by increased new class accuracy (avg. 3.74%). This is due to its class-aware pruning and model refinement, which enable adaptive model adjustments to better accommodat e new classes. While FAMP's accuracy is generally on par with SparCL, it surpasses SparCL on MobileNetV2 (avg. 4.39%) due to its model refinement technique, which benefits smaller models with limited knowledge retention and learning capabilities.

Training Time. Fig. 5(d)-(e) and Fig. 6(d)-(e) show training time for both Server-IL and Edge-IL. Full_IL_prune takes much longer than Full_IL due to post-compression fine-tuning. Compressed_IL, using a fixed compressed model to eliminate fine-tuning, reduces training time to 86% and 52% of Full_IL's for Server-IL and Edge-IL, but compromises new-class accuracy. SparCL improves new-class accuracy with adaptive weight masking but adds an average 23.5% latency over Full_IL due to extra epochs for mask generation and unstructured model training. FAMP avoids these extra epochs. By pruning before IL training and fine-tuning during the IL process, FAMP achieves comparable training time to Compressed_IL - 72% and 58% of Full_IL's for Server-IL and Edge-IL. Despite minor overhead from pruning and model refinement, FAMP's layer-based pruning reduces layers, leading to slightly shorter average training time than Compressed_IL in Server-IL.

TABLE III
ABLATION STUDY TO SHOW THE IMPACT ON THE AVERAGE ACCURACY OF PAST, NEW, AND ALL CLASSES FOR TINY-IMAGENET-100.

Methods	MobileNetV2			ResNet18			DenseNet121		
	past	new	all	past	new	all	past	new	all
Full_IL	25.38	38.87	27.63	30.20	49.27	33.37	37.33	50.47	39.52
FSP	20.20	37.53	23.10	27.44	34.38	28.59	26.97	37.93	28.91
CAP_only	19.11	40.07	22.59	26.60	36.07	27.88	27.70	38.80	29.55
CAP+Union	24.47	38.84	26.87	26.36	49.89	30.28	31.18	44.18	33.34
CAP+Union+CAP	24.78	37.73	26.98	26.19	48.47	29.90	31.86	35.11	32.40
FAMP	30.13	37.29	31.32	29.25	50.20	32.74	36.33	35.86	36.24

Memory Requirements. For Server-IL, FAMP, like Full_IL_prune and Compressed_IL, only needs to store the pruned model at the edge, requiring just 1.5, 12.9, and 3.2MB for MobileNetV2, ResNet18, and DenseNet121. However, the unstructurally pruned SparCL needs the full model and weight masks, leading to 3.75 to 11.27x more storage overhead. For Edge-IL, both Full_IL_prune and FAMP must store the full model alongside the pruned model, with memory usage under 56MB for all models, which is manageable for edge devices. SparCL, however, needs more memory due to the extra weight masks.

C. Ablation Study

Class-aware pruning. We compare models using class-aware pruning alone (CAP_only) with those using fixed structure pruning (FSP)[1]. FSP uses a fixed model structure chosen before IL tasks, based solely on pretrained data. Both methods uses pretrained parameters for each IL iteration, isolating the effect of model structure alone. Table III shows CAP_only improves new-class accuracy by an average of 1.70% over FSP, with MobileNetV2 seeing a 2.54% boost. CAP_only's dynamic adjustment helps this smaller model adapt better to new classes despite its limited learning capacity. However, CAP_only alone worsens catastrophic forgetting, as it does not retain past knowledge, necessitating a combination with methods that preserve past knowledge for better overall accuracy.

Union. To assess the impact of Union, we compare the method that combines CAP and Union (CAP+Union) against CAP_only. As shown in Table III, CAP+Union improves past-class accuracy by 2.87% and overall accuracy by 3.49% on average compared to CAP_only. However, it increases the model size by 1.16% to 3.05% per IL iteration, necessitating an extra pruning step for edge devices with limited memory.

Model refinement. To analyze the impact of model refinement, we compare FAMP, which uses recoverability-aware layer-wise pruning, with CAP+Union+CAP, which employs the same gradient-based filter-wise pruning from the first step for model refinement. Table III shows that FAMP surpasses CAP+Union+CAP in past-class accuracy, particularly with MobileNetV2, which struggles with knowledge retention due to its smaller size. This results in a 3.67% average improvement in overall accuracy. FAMP's advantage comes from its recoverability-aware method, which mitigates bias toward new classes in situations with limited past-class exemplars.

D. Further Analyses

Compatibility. To demonstrate FAMP's compatibility with different IL methods, we conduct experiments based on the widely used iCaRL [26]. Given iCaRL's potential overfitting issues [16], adapting the model structure is crucial. As shown in Fig. 7, Compressed_IL, which uses a fixed model structure, often achieves worse accuracy. By better balancing new and past class knowledge, FAMP generally outperforms other methods, with an average 1.04% accuracy improvement over Full_IL. Additionally, FAMP reduces training time, requiring only 54.04% of Full_IL's time for Edge-IL.

[1]Due to space limitation, we only show results of Section V-C and Section V-D for Tiny-ImageNet-100, while the trend is similar for CIFAR-100.

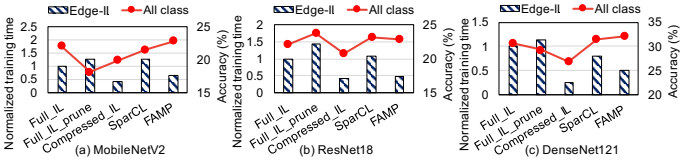

Fig. 7. Normalized Edge-IL training time and average accuracy of all classes on Tiny-ImageNet-100 when IL is based on iCaRL [26].

TABLE IV
AVERAGE CLASSIFICATION ACCURACY ON TINY-IMAGENET-100 WHEN PRUNING RATIO VARIES FROM 70% (PR-70) TO 90% (PR-90).

Methods	MobileNetV2			ResNet18			DenseNet121		
	pr-70	pr-80	pr-90	pr-70	pr-80	pr-90	pr-70	pr-80	pr-90
Full_IL	27.63	27.63	27.63	33.37	33.37	33.37	39.52	39.52	39.52
Full_IL_prune	20.72	17.96	14.70	27.92	26.83	23.93	35.34	32.34	27.42
Compressed_IL	24.23	25.28	27.45	29.78	26.21	22.96	32.31	31.94	33.87
SparCL	30.40	29.97	27.83	32.31	31.07	29.60	34.75	35.53	35.79
FAMP	29.34	29.83	31.32	32.74	30.79	31.12	32.46	34.08	36.24

Pruning ratio. Different edge devices may require varying pruning ratios based on available hardware resources. Table IV shows classification accuracy across different pruning ratios. Increasing the pruning ratio typically reduces accuracy, as the smaller model size limits knowledge retention and learning capacity. Full_IL_prune shows a significant 5.98% average accuracy drop when increasing the pruning ratio from 70% to 90%, as its ability to retain past knowledge diminishes. In contrast, FAMP's use of Union and model refinement effectively retains past knowledge, limiting the accuracy drop to under 1.95% with a 10% increase in pruning ratio. In some cases, FAMP's accuracy even improves with higher pruning ratios, as its recoverability-aware model refinement allows focused tuning on important kernels, demonstrating its suitability for various edge devices.

VI. CONCLUSION

We introduce Filter-based Adaptive Model Pruning (FAMP), a novel model compression method designed to work with class IL. FAMP combines past knowledge retention with adaptive structural adjustments, improving both accuracy and training speed over existing methods. This approach strikes an excellent balance between accuracy and efficiency, making it a promising solution for deploying IL on resource-constrained edge devices. Moreover, FAMP does not rely on any specific hardware or software for acceleration, ensuring broad usability.

ACKNOWLEDGMENT

We would like to thank Jun-Cheng Chen for his valuable suggestions on experimental analysis. This work was supported in part by research grants from the National Science and Technology Council of Taiwan (NSTC 111-2923-E-002-014-MY3, NSTC 112-2221-E-002-116-MY3, and NSTC 112-2221-E-001-002-MY3), and sponsored by Macronix Inc., Hsin-chu, Taiwan (113HT912007).

REFERENCES

[1] E. Belouadah and A. Popescu, "Il2M: Class Incremental Learning with Dual Memory," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 583–592.

[2] T. Gale, E. Elsen, and S. Hooker, "The State of Sparsity in Deep Neural Networks," *arXiv preprint arXiv:1902.09574*, 2019.

[3] S. Han, J. Pool, J. Tran, and W. J. Dally, "Learning both Weights and Connections for Efficient Neural Networks," in *Proceedings of the International Conference on Neural Information Processing Systems - Volume 1*, 2015, p. 1135–1143.

[4] B. Hassibi and D. G. Stork, "Second Order Derivatives for Network Pruning: Optimal Brain Surgeon," in *Proceedings of the International Conference on Neural Information Processing Systems*, 1992, p. 164–171.

[5] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2016, pp. 770–778.

[6] Y. He, G. Kang, X. Dong, Y. Fu, and Y. Yang, "Soft Filter Pruning for Accelerating Deep Convolutional neural networks," *arXiv preprint arXiv:1808.06866*, 2018.

[7] Y. He and L. Xiao, "Structured Pruning for Deep Convolutional Neural Networks: A Survey," *arXiv preprint arXiv:2303.00566*, 2023.

[8] Z. Hou, M. Qin, F. Sun, X. Ma, K. Yuan, Y. Xu, Y.-K. Chen, R. Jin, Y. Xie, and S.-Y. Kung, "CHEX: CHannel EXploration for CNN Model Compression," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 12 277–12 288.

[9] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely Connected Convolutional Networks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2017, pp. 4700–4708.

[10] A. Krizhevsky, G. Hinton *et al.*, "Learning Multiple Layers of Features from Tiny Images," *Master's thesis, Department of Computer Science, University of Toronto*, 2009.

[11] Y. Le and X. Yang, "Tiny ImageNet Visual Recognition Challenge," *CS 231N*, vol. 7, no. 7, p. 3, 2015.

[12] N. Lee, T. Ajanthan, and P. H. Torr, "SNIP: Single-shot Network Pruning based on Connection Sensitivity," *arXiv preprint arXiv:1810.02340*, 2018.

[13] H. Li, A. Kadav, I. Durdanovic, H. Samet, and H. P. Graf, "Pruning Filters for Efficient convNets," *arXiv preprint arXiv:1608.08710*, 2016.

[14] T. Li, J. Li, Z. Liu, and C. Zhang, "Few Sample Knowledge Distillation for Efficient Network Compression," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 14 627–14 635.

[15] Y. Li, K. Adamczewski, W. Li, S. Gu, R. Timofte, and L. Van Gool, "Revisiting Random Channel Pruning for Neural Network Compression," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 191–201.

[16] Y. Li, Z. Li, L. Ding, Y. Pan, C. Huang, Y. Hu, W. Chen, and X. Gao, "SupportNet: Solving Catastrophic Forgetting in Class Incremental Learning with Support Data," *arXiv preprint arXiv:1806.02942*, 2018.

[17] Z. Li, T. Chen, L. Li, B. Li, and Z. Wang, "Can Pruning Improve Certified Robustness of Neural Networks?" *arXiv preprint arXiv:2206.07311*, 2022.

[18] Z. Li and D. Hoiem, "Learning without Forgetting," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 12, pp. 2935–2947, 2017.

[19] Z. Liu, M. Sun, T. Zhou, G. Huang, and T. Darrell, "Rethinking the Value of Network Pruning," *arXiv preprint arXiv:1810.05270*, 2018.

[20] E. S. Lubana and R. P. Dick, "A Gradient Flow Framework for Analyzing Network Pruning," *arXiv preprint arXiv:2009.11839*, 2020.

[21] J.-H. Luo, J. Wu, and W. Lin, " ThiNet: A Filter Level Pruning Method for Deep Neural Network Compression ," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 5068–5076.

[22] M. Masana, X. Liu, B. Twardowski, M. Menta, A. D. Bagdanov, and J. Van De Weijer, "Class-Incremental Learning: Survey and Performance Evaluation on Image Classification," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 45, no. 5, pp. 5513–5533, 2022.

[23] P. Molchanov, A. Mallya, S. Tyree, I. Frosio, and J. Kautz, " Importance Estimation for Neural Network Pruning ," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 11 256–11 264.

[24] W. Niu, X. Ma, S. Lin, S. Wang, X. Qian, X. Lin, Y. Wang, and B. Ren, "PatDNN: Achieving Real-Time DNN Execution on Mobile Devices with Pattern-based Weight Pruning," in *Proceedings of the ACM International Conference on Architectural Support for Programming Languages and Operating Systems*, 2020, pp. 907–922.

[25] M. Nonnenmacher, T. Pfeil, I. Steinwart, and D. Reeb, "SOSP: Efficiently Capturing Global Correlations by Second-Order Structured Pruning," *arXiv preprint arXiv:2110.11395*, 2021.

[26] S.-A. Rebuffi, A. Kolesnikov, G. Sperl, and C. H. Lampert, " iCaRL: Incremental Classifier and Representation Learning ," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2017, pp. 5533–5542.

[27] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, " MobileNetV2: Inverted Residuals and Linear Bottlenecks ," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018, pp. 4510–4520.

[28] C. Shen, X. Wang, Y. Yin, J. Song, S. Luo, and M. Song, "Progressive Network Grafting for Few-Shot Knowledge Distillation," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 3, 2021, pp. 2541–2549.

[29] M. Shen, H. Yin, P. Molchanov, L. Mao, J. Liu, and J. M. Alvarez, "Structural Pruning via Latency-Saliency Knapsack," *Proceedings of the Advances in Neural Information Processing Systems*, vol. 35, pp. 12 894–12 908, 2022.

[30] S. P. Singh and D. Alistarh, "WoodFisher: Efficient Second-Order Approximation for Neural Network Compression," in *Proceedings of the International Conference on Neural Information Processing Systems*, 2020.

[31] F.-Y. Wang, D.-W. Zhou, H.-J. Ye, and D.-C. Zhan, "FOSTER: Feature Boosting and Compression for Class-Incremental Learning," in *Proceedings of the European Conference on Computer Vision*, 2022, p. 398–414.

[32] G.-H. Wang and J. Wu, " Practical Network Acceleration with Tiny Sets ," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 20 331–20 340.

[33] H. Wang, J. Liu, X. Ma, Y. Yong, Z. Chai, and J. Wu, " Compressing Models with Few Samples: Mimicking then Replacing ," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 691–700.

[34] Z. Wang, Z. Zhan, Y. Gong, G. Yuan, W. Niu, T. Jian, B. Ren, S. Ioannidis, Y. Wang, and J. Dy, "SparCL: Sparse Continual Learning on the Edge," in *Proceedings of the International Conference on Neural Information Processing Systems*, 2024.

[35] J. Zhang, J. Zhang, S. Ghosh, D. Li, S. Tasci, L. Heck, H. Zhang, and C.-C. Jay Kuo, " Class-Incremental Learning via Deep Model Consolidation ," in *Proceedings of the IEEE Winter Conference on Applications of Computer Vision*, 2020, pp. 1120–1129.