AdaptCL: Adaptive Continual Learning for Tackling Heterogeneity in Sequential Datasets

Yuqing Zhao, Divya Saxena, *Member, IEEE*, and Jiannong Cao, *Fellow, IEEE*

***Abstract*— Managing heterogeneous datasets that vary in complexity, size, and similarity in continual learning presents a significant challenge. Task-agnostic continual learning is necessary to address this challenge, as datasets with varying similarity pose difficulties in distinguishing task boundaries. Conventional task-agnostic continual learning practices typically rely on rehearsal or regularization techniques. However, rehearsal methods may struggle with varying dataset sizes and regulating the importance of old and new data due to rigid buffer sizes. Meanwhile, regularization methods apply generic constraints to promote generalization but can hinder performance when dealing with dissimilar datasets lacking shared features, necessitating a more adaptive approach. In this article, we propose a novel adaptive continual learning (AdaptCL) method to tackle heterogeneity in sequential datasets. AdaptCL employs fine-grained data-driven pruning to adapt to variations in data complexity and dataset size. It also utilizes task-agnostic parameter isolation to mitigate the impact of varying degrees of catastrophic forgetting caused by differences in data similarity. Through a two-pronged case study approach, we evaluate AdaptCL on both datasets of MNIST variants and DomainNet, as well as datasets from different domains. The latter include both large-scale, diverse binary-class datasets and few-shot, multiclass datasets. Across all these scenarios, AdaptCL consistently exhibits robust performance, demonstrating its flexibility and general applicability in handling heterogeneous datasets.**

***Index Terms*— Adaptive continual learning (AdaptCL), data-driven pruning, heterogeneous datasets, parameter isolation, task-agnostic continual learning.**

I. Introduction

THE past decade has witnessed a surge in data generation, facilitated by sensor-equipped devices and rapid digitization, across diverse domains, such as healthcare, smart manufacturing, transportation, food safety, and so on. However, datasets associated with these domains often originate from multiple sources or at different times, contributing to their inherent heterogeneity, which encompasses variations in dataset size, complexity, and similarity. This heterogeneity presents unique challenges, particularly in implementing continual learning algorithms.

Manuscript received 2 August 2022; revised 13 July 2023 and 6 November 2023; accepted 1 December 2023. This work was supported in part by the Shenzhen–Hong Kong–Macau Technology Research Program under Grant SGDX20201103095203029; in part by the HK RGC Research Impact Fund under Grant R5034-18; in part by the HK RGC General Research Fund PolyU under Grant 15204921; and in part by the Research Institute for Artificial Intelligence of Things, The Hong Kong Polytechnic University. *(Corresponding author: Yuqing Zhao.)*

The authors are with the Department of Computing, The Hong Kong Polytechnic University, Hong Kong (e-mail: csyzhao1@comp.polyu.edu.hk; divsaxen@comp.polyu.edu.hk; csjcao@comp.polyu.edu.hk).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TNNLS.2023.3341841.

Digital Object Identifier 10.1109/TNNLS.2023.3341841

As machine learning models, particularly continual learning models, gain prominence in these domains, it becomes evident that they must be robust and flexible enough to accommodate the inherent heterogeneity of datasets. This heterogeneity often manifests in several ways: the size of the dataset can range from few-shot examples to large-scale samples; the complexity of data can differ based on the range and intricacy of features, and the similarity of data can vary, which can create difficulties in distinguishing task boundaries. Conventional continual learning methods for these scenarios [1], [2], [3] are typically task-agnostic and depend on either rehearsal or regularization techniques, and they have limitations when dealing with such datasets. The rehearsals often struggle with size variability due to a rigid buffer size that makes the importance regulation between old and new data challenging, while regularization techniques may hinder performance when dealing with dissimilar datasets that lack shared features. These challenges underscore the need for a more adaptive approach to handling heterogeneous datasets.

On the other hand, structure-based methods, such as parameter isolation, show great promise in handling both similar and dissimilar domains. These methods segment the network into distinct modules that do not interfere with each other during inference (Fig. 1). However, they are primarily suitable for task-specific continual learning, where the manual selection of the parameter module, based on the task category during inference, is feasible. In the case of task-agnostic continual learning, using all parameters for integrated inference can lead to significant interference and a drop in accuracy [4]. Therefore, the direct application of parameter isolation to task-agnostic continual learning is unsuitable without an appropriate adaptation mechanism.

Building on our previous work [5], we propose adaptive continual learning (AdaptCL). AdaptCL enables adaptive learning through fine-grained data-driven pruning, effectively responding to variations in data complexity and dataset size. It also employs task-agnostic parameter isolation to ensure optimal model performance across datasets with varying similarity levels, all without the need for manual module selection. AdaptCL draws inspiration from the human brain's adaptive nervous system, a complex neural network that dynamically prunes redundant synapses [6], [7] and reuses neural circuits for different tasks without compromising the original functions during development [8].

AdaptCL is a pioneering task-agnostic continual learning approach designed specifically to tackle heterogeneity in sequential datasets. The key contributions of this research can be summarized as follows.

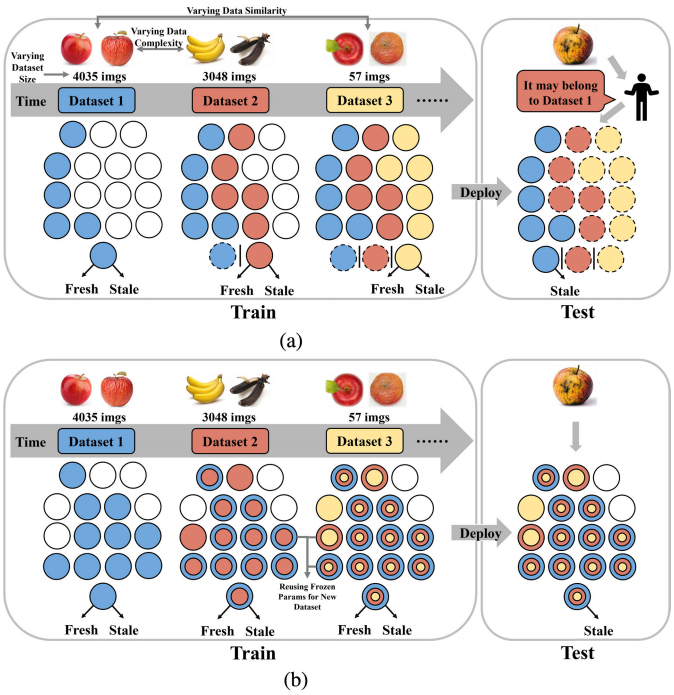

Fig. 1. (a) Traditional parameter isolation methods divide the network into noninterfering modules during inference. However, these methods are limited to task-specific continual learning (aka task incremental learning). They require manual selection of output layers and parameters, resulting in limited generalization and higher parameter usage. (b) AdaptCL achieves task-agnostic parameter isolation by fine-grained data-driven parameter partitioning, enabling high accuracy on heterogeneous datasets without module selection, while also optimizing parameter reuse and saving resources.

1) We conduct the first comprehensive investigation into adaptive continual learning for managing heterogeneous datasets, irrespective of their complexity, size, and similarity. This innovative approach does not require retraining or different models for varying batches of data, marking a significant leap in continual learning techniques.
2) Our method, AdaptCL, uniquely employs a combination of fine-grained data-driven pruning and task-agnostic parameter isolation to address the problems of catastrophic forgetting and variations in data complexity and dataset size. These adaptive mechanisms enable the model to respond effectively to different scenarios, increasing both flexibility and robustness.
3) Extensive experiments conducted on several datasets, including MNIST variants, DomainNet, and large-scale diverse and few-shot, multiclass food quality datasets, demonstrate the general applicability and resilience of AdaptCL. The method consistently outperformed existing solutions, providing higher average accuracy (AAC) and versatility across different networks and applications.

II. Related Works

Continual learning methods are crucial tools in the field of machine learning, aiding in the effective handling of tasks that evolve over time. The existing methods primarily fall into three categories: rehearsal-based, regularization-based, and structure-based. This section provides a detailed overview of these methods, highlighting their strengths and limitations, particularly when applied to task-agnostic continual learning and the management of heterogeneous datasets.

A. Task-Agnostic Continual Learning

1) Rehearsal-Based: These techniques seek to overcome catastrophic forgetting, a significant challenge in continual learning, by replaying previous training data periodically. Early methods, such as GEM and A-GEM [9], [10], relied on storing a portion of past training data and reusing it in future training phases. This approach has been further refined by Peng et al. [11] and Ho et al. [12] with the incorporation of generative networks to create synthetic data distributions for pseudo-rehearsals. To enhance memory efficiency in rehearsal techniques, Zhao et al. [2] adopted auxiliary low-fidelity exemplar samples.

LwF [13] introduces knowledge distillation that utilizes a teacher network to distill knowledge and soft targets to a student network while training on new tasks, enabling retention of knowledge from previous tasks. Some combine replay with knowledge distillation like Rosasco et al. [14] keep a very small buffer for highly informative samples and combine with distillation playback, and Sun et al. [15] distill knowledge and replay experience from previous tasks when fitting on a new task. ICaRL [16] adopts a combination of rehearsal and regularization through learning a compact and discriminative feature representation to enable class-incremental learning. Similarly, He and Zhu [3] adopt a combination of rehearsal and regularization that uses the nearest class mean (NCM) classifier on food image classification dataset Food1k-100; the class mean of all data seen so far is estimated by the online mean update standard during the training phase. PRE-DFKD [17] further refines these strategies and proposes to rehearse the model using the data-free knowledge distillation through the distribution of the previously observed synthetic samples from a variational autoencoder.

Despite these advancements, rehearsing techniques face limitations when managing datasets of varying sizes and maintaining the balance between old and new data. However, with AdaptCL, the model allocates parameters based on the accuracy in a data-driven way, allowing it to retain knowledge as parameter-level representations, independent of the data volume.

2) Regularization-Based: These methods incorporate regularization techniques, such as weight decay or dropout, to prevent catastrophic forgetting in neural networks when learning multiple tasks sequentially. Inspired by Bayesian learning, elastic weight consolidation (EWC) [18], [19] mitigates catastrophic forgetting by tracking changes using the Fisher information matrix.

He and Zhu [1] adopt knowledge distillation on augmented exemplars in a class-incremental setting on food image classification.

Liu et al. [20] introduce an instance neighborhood-preserving loss and a label priority-preserving loss to maintain the relative relationships of model responses within a set of instances and the relative relationships of model outputs with

respect to an instance, contrasting with traditional knowledge distillation methods that retain absolute responses of isolated instances.

P&C [21] compress learned knowledge and distill it into the knowledge base, and preserve knowledge with EWC while using the active column to progress new data. Using a Bayesian neural network, CBLN [22] preserves distinctive parameters for different datasets for retaining performance. Similarly, Park et al. [23] introduced developmental memory (DM) into a CNN, continually growing submemory networks to preserve important features of learned tasks while allowing faster learning. Each submemory can store task-specific knowledge by using a memory loss function and preserve it during continual adaptations. HAT [24] learns an attention mask over important parameters. By aligning local representations, P-TNCN [25] replaces the backpropagation method that descent steepest, punishing parameter updates to a more generalized result, therefore mitigating catastrophic forgetting.

Despite the potential of regularization-based methods, they can face challenges when handling heterogeneous datasets, especially those that are dissimilar and have few shared features. While through parameter isolation in a data-driven manner, AdaptCL can effectively adapt to datasets with varying levels of similarity, including dissimilar ones.

B. Task-Specific Continual Learning

1) Structure-Based: Structure-based methods are primarily employed in task-specific scenarios, and these methods use parameter isolation to handle both similar and dissimilar domains effectively. They divide the network into separate modules to mitigate interference during inference. While these techniques excel in managing catastrophic forgetting, they present difficulties when directly applied to task-agnostic scenarios.

One approach, exemplified by progressive neural nets (PNNs) [26], involves a static growth of the architecture with equal-sized modules, allowing for forward knowledge transfer (FWT) between them. However, this method lacks a data-driven approach and requires task-specific settings for subsequent tasks, limiting its flexibility. Another approach, represented by SILF [27], addresses parameter isolation by pruning unimportant parameters, isolating the important ones to mitigate forgetting. However, SILF relies on manual pruning ratio setting instead of leveraging a data-driven approach. Reinforced continual learning (RCL) [28] expands each layer using reinforcement learning and enables parameter sharing. Nevertheless, this method necessitates task labels as additional inputs during inference to determine the parameters to use. To strike a balance between knowledge transfer and catastrophic forgetting, CLAW [29] identifies which parts of the network should be shared or preserved for specific tasks. PathNet [30] and RPS-Net [31] adopt a modularized network with multiple possible paths from input to output. They choose specific paths based on tasks or dataset labels. Additionally, RPS-Net includes a distillation loss and retrospection replay to further minimize forgetting. CAT [32] masks used parameters and blocks gradient flow through unused units for dissimilar tasks. Task masks are stored according to task ID or label and need to be retained during testing. Other methods, such as DAM [33], CLNP [34], and PackNet [35], leverage pruning to strike a balance between model sparsity and performance. DAM assigns learning of each domain to a fraction of the network, typically with the same percentage (e.g., 13%). CLNP and PackNet prune parameters based on specific percentages.

Notably, the power of structure-based parameter isolation methods, such as PackNet, has been demonstrated through recent advancements [36] that have shown superior performance compared to other continual learning methods [9], [13], [16], [18], [24], [37].

However, challenges persist, particularly when dealing with heterogeneous datasets in task-agnostic settings, which calls for more adaptive approaches. Our previous work [5] priorly applied the structure-based parameter isolation method to the task-agnostic scenario. However, its coarse-grained pruning resulted in limited adaptability to the heterogeneity of dataset size and similarity, leading to suboptimal accuracy. Additionally, more adequate validation is needed on heterogeneous datasets.

III. Problem Setting and Objective

We are given a sequence of non-IID datasets $D_1, D_2, \ldots, D_n$ for a fixed task. Each dataset consists of a group of labeled data $(X, Y) \in D$, where X and Y are input variables and the corresponding output variables, respectively. A task-agnostic continual learning setting aims to optimize

$$\max_{\theta} E_{t\sim D}\left[E_{(X,Y)\sim D_t}\left[\log p_\theta(Y|X)\right]\right] \tag{1}$$

where θ identifies the parametrization of the network. Such a maximization problem is subject to continual learning constraints: when accessing the current dataset D at time t, it is impractical or impossible to access any previous or future dataset. We aim to develop a task-agnostic continual learning method that can effectively handle a sequence of heterogeneous datasets.

Here, the absence of known task or dataset labels prevents task-aware inference in the model. The task-agnostic setting requires merging the output units into a single-headed classifier, with more serious task interference between data from different domains, which leads to more severe forgetting [4].

IV. Adaptive Continual Learning

AdaptCL (Fig. 2) employs adaptive learning that utilizes fine-grained data-driven pruning to adapt to variations in data complexity and dataset size. It also employs a form of task-agnostic parameter isolation to mitigate the impact of varying degrees of catastrophic forgetting caused by differences in data similarity.

A. Fine-Grained Data-Driven Pruning

In continuous learning with heterogeneous datasets, effectively managing model complexity becomes crucial within a limited computational budget. Fine-grained pruning goes

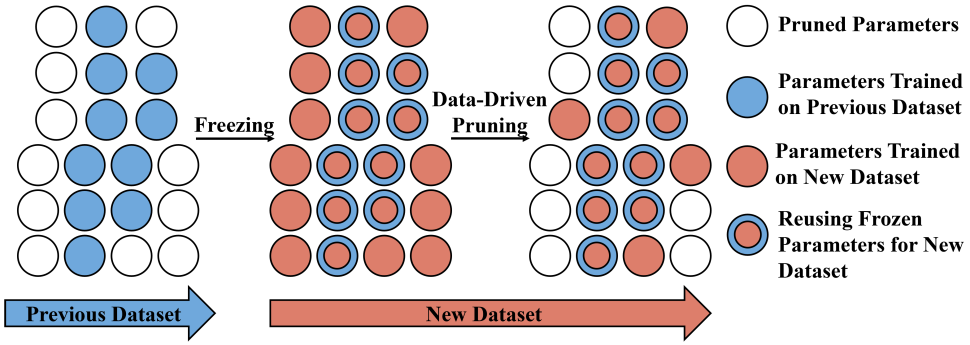

Fig. 2. AdaptCL training flow. It facilitates adaptive learning via fine-grained data-driven pruning to respond effectively to variations in data complexity and dataset size. Additionally, it enables task-agnostic parameter isolation to ensure optimal model performance on datasets ranging in similarity without requiring the manual selection of modules.

beyond traditional pruning approaches by compressing the model while maintaining or even increasing accuracy. This data-driven pruning method aims to strike a balance between network accuracy and sparsity, facilitating better parameter reuse among similar datasets and improved fitting accuracy for complex or dissimilar datasets. Let us consider a neural network with a parameter set $\{W_i : 1 \leq i \leq C\}$, where W_i represents the parameter matrix at layer i and C denotes the number of layers. For fully connected layers, the corresponding parameter is $W_i \in R^{c_o \times c_i}$, where c_o is the output dimension and c_i is the input dimension. For convolutional layers, a convolution kernel $K_i \in R^{c_o \times c_i \times w \times h}$ exists, where c_o represents the number of output channels and c_i, w, and h denote the number of input channels, width, and height, respectively. Pruning involves applying a binary mask M^p to each parameter W, setting unimportant parameters to 0. To determine the masks, a trainable pruning threshold vector t is introduced. The magnitude of parameters is compared to the corresponding threshold values using a unit step function $S(x)$, as shown in (3)

$$S(x) = \begin{cases} 0, & x < 0 \\ 1, & x \geq 0 \end{cases} \tag{2}$$

$$M^p{}_{ij} = S\big(|W_{ij}| - t_i\big), \quad 1 \leq i \leq c_o, \ 1 \leq j \leq c_i. \tag{3}$$

The corresponding element in pruning mask $M^p{}_{ij}$ will be set to 0 if W_{ij} needs to be pruned.

Unlike traditional methods that use a fixed threshold value, achieving fine-grained pruning requires a high-dimensional threshold, denoted as t, in order to ensure more precise pruning. For a fully connected layer or recurrent layer with a parameter size of $W \in R^{c_o \times c_i}$, our threshold tensor size is $t \in R^{c_o}$. Each weight W_{ij} will have a neuronwise threshold, denoted as t_i, where W_{ij} represents the jth weight associated with the ith output neuron. Similarly, for convolutional layers, the thresholds are filterwise. Consequently, each neural network layer will be pruned based on high-dimensional thresholds, where each row of the tensor has its unique threshold. This approach ensures a more fine-grained pruning, avoiding the removal of potentially important parameters. For fully connected and recurrent layers, instead of using the dense parameter W, the sparse product $W \circ M^p$ is used in the batched matrix multiplication, where $\circ$ represents the Hadamard product operator. As for convolutional layers, each convolution kernel is flattened to obtain W, following a process similar to that of fully connected layers.

Inspired by the dynamic sparse training [38], we separate important and unimportant parameters by learning a threshold for each fully connected and convolutional neural network layer during training on one dataset. This threshold is a trainable parameter that is updated along with the backpropagation of the neural network to achieve a stepwise update. In order to make the binary step function $S(x)$ in threshold vector t trainable via backpropagation, a derivative estimation is needed. A long-tailed higher order estimator $H(x)$ proposed by Xu and Cheung [39] is adopted for a balance of tight approximation and smooth backpropagation

$$\frac{d}{dx} S(x) \approx H(x) = \begin{cases} 2 - 4|x| & -0.4, \leq x \leq 0.4 \\ 0.4, & 0.4 < |x| \leq 1 \\ 0, & \text{otherwise.} \end{cases} . \tag{4}$$

To get the pruning masks M^p with high sparsity, higher pruning thresholds are needed. To achieve this, a sparse regularization term L_s is added to the training loss that penalizes the low threshold value. For each trainable masked layer with threshold t, the corresponding regularization term is $R = \sum_{i=1}^{c_o} \exp(-t_i)$. Thus, the sparse regularization term L_s for a neural network with C trainable masked layers is

$$L_s = \sum_{i=1}^{C} R_i. \tag{5}$$

$\exp(-x)$ is used as the regularization function, since it is asymptotical to zero as x increases. Consequently, it penalizes low thresholds without encouraging them to become extremely large. Given the training dataset D, a sparse neural network can be trained directly with backpropagation algorithm by adding the sparse regularization term L_s to the loss function as follows:

$$W^*, t^* = \text{argmin}[L(D; W) + \alpha \ L_s] \tag{6}$$

where $L(\cdot)$ is the loss function, e.g., cross-entropy loss for classification, and α is the scaling coefficient for the sparse regularization term, which can control the percentage of parameters remaining. It is calculated according to the total number of iterations of one dataset. For a new dataset, the

pruning threshold t is reinitialized, and a new round of fine-grained data-driven pruning is restarted and be applied to neural network parameters not occupied by previous datasets.

B. Task-Agnostic Parameter Isolation

To address the issue of catastrophic forgetting with varying similarity datasets, we enhance the technique of parameter isolation. Traditional methods freeze learned parameters during both training and inference, preventing them from being updated and masking task-specific parameters. In contrast, our data-driven approach progressively learns from frozen parameters while utilizing all parameters during inference. This allows effective handling of heterogeneity in data similarity during continuous learning. Parameter freezing in neural networks involves preventing specific parameters from being updated during training. In our approach, we introduce a binary freeze mask, denoted as M, of the same shape as the parameters. This mask has a value of 1 for parameters that are allowed to be updated and 0 for frozen parameters. We obtain the frozen parameters, θ_f, by elementwise multiplying the original parameters by this mask

$$\theta_f = \theta \odot M.$$

During training, the gradients computed with respect to the loss function are applied only to the nonfrozen parameters, updating them according to the optimization algorithm. The frozen parameters remain unchanged throughout the training process. At the end of training on each dataset, we calculate a freeze mask M^f, which is the result of the union between the existing pruning mask and the freeze mask from the previous round. This mask is used to freeze the learned parameters during the next dataset training. The freeze mask M^f is calculated as follows:

$$M^f{}_{ij} = S\big(|M^f{}_{ij} + M^p{}_{ij}|\ \big), \quad 1 \le i \le c_o,\ 1 \le j \le c_i \quad (7)$$

where S is the sign function, c_o is the number of output channels, c_i is the number of input channels, and $M^p{}_{ij}$ is the pruning mask obtained after pruning.

In order to ensure that the corresponding gradient of the parameters in the freeze mask $M^f{}_{ij}$ is set to 0 when W_{ij} needs to be frozen, we use the following equation:

$$W^*, t^* = \text{argmin}[L(D; W, t) + \alpha L_s] \circ \big(1 - M^f\big). \quad (8)$$

Here, $L(D; W, t)$ denotes the loss function on the current dataset, L_s is the penalty for changes in learned parameters, α is the learning rate, W^* and t^* denote the optimal value of the weight and threshold, respectively, and $\circ$ denotes elementwise multiplication between the matrices. During inference, all the parameters, including the frozen ones, are used to make predictions, as the model has already learned useful representations from them. By applying parameter freezing, a neural network can retain knowledge from previous tasks while allowing for further learning without catastrophic forgetting. For a new dataset, adaptive continual learning initiates a fresh iteration while preserving important frozen parameters. Pruning is only applied to other free neural network parameters.

Algorithm 1 Training Flow of AdaptCL

```
Require: weight of parameter W, threshold vector t is initialized with zero tensor.
for dataset d = 0, 1, 2, ..., do
    for layer in model do
        Reset threshold t ← 0
    end for
    for epoch do
        for step do
            update pruning mask M^p_ij = S(|W_ij| − t_i )
            update pruned weight W = W ∘ M^p
            for layer in model do
                update the loss L(·) = L(D; W) + αL_s
            end for
            if d == 0 then
                gradient decent W*, t* = argminL(·)
            else
                gradient decent with frozen parameters W*, t* = argminL(·) ∘ (1 − M^f)
            end if
        end for
    end for
    update freeze mask M^f_ij = S(|M^f_ij + M^p_ij| )
end for
```

C. Adaptive Continual Learning Training Flow

Referring to the algorithm flow of our proposed method, depicted in Algorithm 1, at the start of each new round of dataset training, threshold parameters are initialized. During training, these threshold parameters are calculated and updated at each step of backpropagation, leading to the refinement of the pruning mask

$$M^p{}_{ij} = S\big(|W_{ij}| - t_i\ \big).$$

The refinement process, being fine-grained, data-driven, and stepwise, allows AdaptCL to adapt to variations in data complexity and dataset size. Throughout the training process, AdaptCL freezes the gradient descent at each step based on the freeze mask $(1-M^f)$. This protects the current parameters from further modification, preserving the knowledge learned in previous training rounds and mitigating the impact of catastrophic forgetting caused by variations in data similarity. At the end of each round of training, AdaptCL generates an updated freeze mask to protect the current set of parameters for future training. This allows AdaptCL to continue learning from new data while retaining the knowledge gained from previous training rounds. Overall, the adaptive learning with fine-grained data-driven pruning approach, coupled with task-agnostic parameter isolation, enables AdaptCL to effectively adapt to variations in data complexity and dataset size while mitigating the impact of variation of data similarity during the training process.

V. Experiments

Our method is evaluated on a range of benchmark datasets with heterogeneous characteristics, encompassing various

domains and tasks. To assess the performance of our method, we apply the method to the widely recognized ResNet-18, LeNet-5, and VGG-16 architectures. To establish a solid benchmark for comparison, we implement several other baseline algorithms in the domain incremental setting. These algorithms include SGD as the naive setting, as well as EWC, LwF, PRE-DFKD, PackNet*, and separated models for learning (SML). Particularly, PackNet* represents an extension of PackNet specifically designed for our task-agnostic evaluation.

By conducting experiments on these benchmark datasets and comparing our method against these baseline algorithms, we aim to gain insights into the performance of our proposed approach and to assess its effectiveness in addressing the challenges of tackling data heterogeneity in continual learning. In particular, we aim to answer the following research questions.

1) *Q1*: How does AdaptCL compare to other baseline continual learning methods in terms of AAC and parameter efficiency?
2) *Q2*: What is the effectiveness of AdaptCL in managing heterogeneity in sequential datasets from different application domains, such as food quality and DomainNet?
3) *Q3*: What is the impact of AdaptCL's fine-grained data-driven pruning technique on adapting to differences in data complexity and dataset size?
4) *Q4*: How does AdaptCL's task-agnostic parameter isolation approach mitigate catastrophic forgetting in the presence of varying degrees of data similarity?

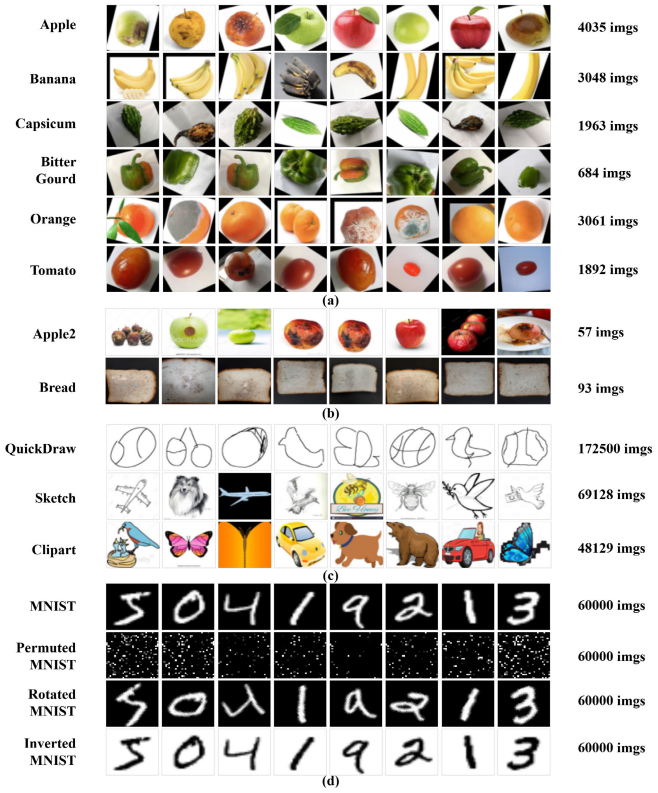

Fig. 3. Examples of input images used in the experiments. (a) Large-scale, diverse binary-class food quality dataset. (b) Few-shot, multiclass food quality dataset. (c) DomainNet comprises datasets with heterogeneous complexity and size. (d) MNIST variants with heterogeneous similarity.

A. Datasets

We choose the following datasets to evaluate our method.

1) Large-Scale, Diverse Binary-Class Food Quality Dataset: The dataset comprises a total of 14 683 images of six different types of fruits and vegetables, as shown in Fig. 3(a), including apples, bananas, bitter gourds, capsicums, oranges, and tomatoes. Each image in the dataset is classified as either fresh or stale. The datasets vary in size, and the images are obtained from various sources, such as online image repositories, self-captured images, or artificially generated images through data augmentation techniques, resulting in different levels of complexity and similarity among the datasets. The datasets are designed to be heterogeneous and challenging to evaluate the robustness and generalization of machine learning models. All the images in the dataset have been preprocessed to ensure a uniform size and aspect ratio of 64 × 64 pixels. The total size of the dataset is approximately 2 GB.

2) Few-Shot, Multiclass Food Quality Dataset: The dataset is used as a real-life application case to verify our solution with a small dataset size, which poses a more challenging scenario compared to the previous binary classification dataset. This dataset comprises images of apples and bread, each associated with a freshness score label. The freshness scores range from 0 to 4, where 0 represents total corruption and 4 indicates total freshness. The apple dataset consists of a total of 57 images, while the bread dataset contains 93 images, as illustrated in Fig. 3(b). This dataset aims to evaluate the model's performance in adapting to very few samples and the ability to transfer knowledge to solve underfitting.

3) DomainNet With Heterogeneous Complexity and Size: The DomainNet [40] dataset consists of image data from six domains, each with a different amount of data, including real photographs, painting, clipart, infograph, quickdraw, and sketch. There are 48–172k images (600k in total) categorized into 345 classes per domain. DomainNet comprises datasets with heterogeneous complexity and size. For instance, the quickdraw dataset holds 172 500 images but requires only 439 MB of storage, while the sketch dataset includes 69 128 images but occupies 2.5 GB of storage. The sketch, quickdraw, and clipart domains are selected as datasets, as shown in Fig. 3(c), to evaluate models' performance on datasets with different complexities.

4) MNIST and Variants With Heterogeneous Similarity: To provide additional validation for our model, we choose to utilize MNIST dataset with its variants, which include the MNIST, Permuted MNIST, Inverted MNIST, and Rotated MNIST datasets. These datasets are organized into sequences, reflecting a different level of similarity, and in both domain-incremental and class-incremental settings.

The datasets each consist of 70 000 images of handwritten digits from 0 to 9 of size 32 × 32. In each dataset, 60 000 images are used for training and 10 000 images for the test, as listed in Fig. 3(d).

1) *Permuted MNIST:* It is an MNIST variant that applies a fixed random permutation of the pixels of the MNIST digits. It also includes the same number of images of

handwritten digits. Permuted MNIST bears no resemblance to MNIST at all.

2) *Inverted MNIST:* It is another variant of the MNIST dataset inverting the color of MNIST images from black to white. The Inverted MNIST and MNIST are the exact opposite in the color of the input data and the same in the output.
3) *Rotated MNIST:* It is also a variant of the MNIST dataset. It rotates MNIST data randomly by 0°–45°. There is some overlap of data between the MNIST and the Rotated MNIST, making the two datasets similar to each other.

B. Networks Used

To evaluate the applicability of our proposed technique on networks of different sizes and structures, we conducted our experiments using three popular network architectures: LeNet-5, ResNet-18, and VGG-16.

LeNet-5 is a relatively simple architecture with 61 706 parameters and a compact size of 0.24 MB. It was primarily designed for digit recognition in checks and consists of seven convolutional layers. However, due to its limited number of convolutional layers, LeNet-5 may face resource constraints when processing sequential datasets.

ResNet-18, on the other hand, is a more complex architecture with 11 172 810 parameters and a larger size of 42.62 MB. This network incorporates a greater number of convolutional layers, making it better equipped to handle complex image recognition tasks. The increased number of parameters allows for a larger network capacity, which is advantageous for continuous learning scenarios involving sequential datasets.

Finally, we utilized the VGG-16 architecture, which is more parameter-rich, with 14 986 570 parameters and a size of 57.17 MB. This architecture offers a high degree of expressiveness due to its numerous convolutional and fully connected layers.

C. Evaluation Metrics

For a principled evaluation, we adopt the following evaluation metrics [9].

1) $\text{AAC} = (1/T)\sum_{i=1}^{T} R_{T,i}$.
2) $\text{BWT} = (1/T-1)\sum_{i=1}^{T-1}\left(R_{T,i} - R_{i,i}\right)$.
3) $\text{FWT} = (1/T-1)\sum_{i=2}^{T-1} R_{i-1,i} - \bar{b}_i$.

We consider access to a testing dataset for each of the D datasets. After the model finishes learning about the domain t_i, we evaluate its test performance on all T datasets. By doing so, we construct the matrix $R \in R^{t\times t}$, where $R_{i,j}$ is the test classification accuracy of the model on the dataset t_j after observing the last sample from dataset t_i. Letting $\bar{b}$ be the vector of test accuracy for each task at random initialization. For comparison, our primary criterion for evaluating performance is the AAC metric, where higher values indicate better performance. Additionally, we consider the metrics of BWT and FWT efficiency, with higher values being preferred. Furthermore, we calculate parameters (Params) to assess parameter efficiency. To gain a deeper understanding of model performance across datasets, we also compare test accuracy for each dataset.

D. Baselines

To validate the effectiveness of our method in continual learning with heterogeneous datasets, we compare our model with baseline algorithms. We implement all of the following described baselines in our code base.

1) *SML:* Separate models are trained for every task, achieving the highest possible accuracy by dedicating all the network resources to that single dataset. In this case, there is no knowledge transfer or catastrophic forgetting. It requires manual selection of the model during inference.
2) *Stochastic Gradient Descent (SGD) [41]:* A naïve model trained with direct SGD.
3) *EWC [18]:* A regularization technique in continual learning that uses diagonal elements of Fisher information matrix to constrain the weights of the neural network and avoid catastrophic forgetting.
4) *LwF [13]:* A rehearsal-based method that uses knowledge distillation to preserve previously learned knowledge along with training on new tasks.
5) *PRE-DFKD [17]:* A recently proposed rehearsal strategy that rehearses the model using the data-free knowledge distillation through the distribution of the previously observed synthetic samples from a variational autoencoder.
6) *PackNet [35]:* A structure-based parameter isolation method that prunes a specific ratio of the network during training to sequentially "pack" multiple tasks into a single network. It requires knowing the number of datasets ahead to calculate the pruning ratio. Also, it needs to select masks to indicate network modules to perform during inference. We implement it in the task-agnostic setting referred to as PackNet* later in this article.

E. Implementation Details

We use PyTorch and Torchvision libraries to implement neural networks. All of the training images are scaled and normalized before training as preprocessing. Identical processes are applied to the test images. The optimizer is SGD, with a 0.001 learning rate, 0.9 as the momentum value, and Nesterov accelerated gradient for regularization. To guarantee completely reproducible results, we set seed value as 5 for the random function of Numpy, Python Random, PyTorch, PyTorch Cuda, and set PyTorch Backends Cudnn benchmark as false, with deterministic as true, configuring PyTorch to avoid using nondeterministic algorithms for some operations, so that multiple calls to those operations, given the same inputs, will produce the same result. Algorithm 1 shows the learning procedure of AdaptCL. We keep all the settings the same for our method and the baselines.

Considering the Fisher matrix of EWC, we use EWC λ as 1. Regarding PackNet, we implement it in a domain-incremental setting, which we refer to as PackNet* in our

paper. Instead of using a pretrained model, we train it for the same number of epochs as other methods, selecting ten epochs of sparse training following pruning, as discussed in PackNet's paper. To ensure each dataset received equal attention, we prune the network to assign the same ratio of $1/T$ parameters per dataset, where T is the number of datasets. For PRE-DFKD, we follow the default setting and use Kullback–Leibler divergence (KLD) loss with a hyperparameter of 10^{-5}. Regarding LwF, we set the hyperparameters alpha and temperature to 1 and 3, respectively. For the naïve settings with SGD, we simply fine-tune the network on each new dataset without making any network modifications. For SML, we use one network for training on every single dataset and do not fine-tune it on other datasets.

To facilitate the reproducibility of our experiments, we have made the source **code** available.

VI. Results

A. Performance on Datasets With Varied Sizes (Q1–Q3)

1) Large-Scale, Diverse Binary-Class Food Quality Dataset: As shown in Table I, our method achieves an AAC of 78.20% on the six food datasets, surpassing baseline methods by 4.32%. It outperforms other approaches in terms of final accuracy and has the lowest parameter count (1.091×10^7) while effectively overcoming catastrophic forgetting. Despite having fewer parameters, our model successfully fits up to six datasets, with some minor accuracy gaps compared to SML due to data complexity and the need for increased capacity. To unlock its full potential, we recommend scaling up the model for improved accuracy in continual learning.

Analyzing Fig. 4, we observe the varying impact of learning across datasets due to their heterogeneity. In most cases, our model maintains the highest accuracy on the learned datasets. Comparing it to PackNet*, which also uses pruning methods, we notice a notable accuracy increase on unlearned datasets during pruned epochs, indicating the efficacy of pruning for enhancing generalization in continual learning with relevant datasets. Our model struggles with the orange dataset following training on capsicum and bitter gourd datasets due to conflicting features, mainly caused by the low data resolution of 64×64 pixels, which led the model to primarily rely on color and shape to differentiate images. This issue, observed in all baseline models, can be resolved by increasing data resolution.

2) Few-Shot, Multiclass Food Quality Dataset: We test our method on this dataset to evaluate its ability to generalize on small datasets. Similar to the human brain, which excels at few-shot learning and generalizing from limited examples, our AdaptCL method, inspired by the neural reuse principle, improves efficiency and performance on small sample datasets. As shown in Table II, when evaluating the few-shot, multiclass food quality dataset, AdaptCL achieves 99.50% accuracy using 10% fewer parameters than baseline methods, even producing a rare positive BWT of **1.08%**, meaning the positive consequence of inductive knowledge transfer is more significant than catastrophic forgetting. Since the first dataset is small, the model is not fully trained, and easy to overfit;

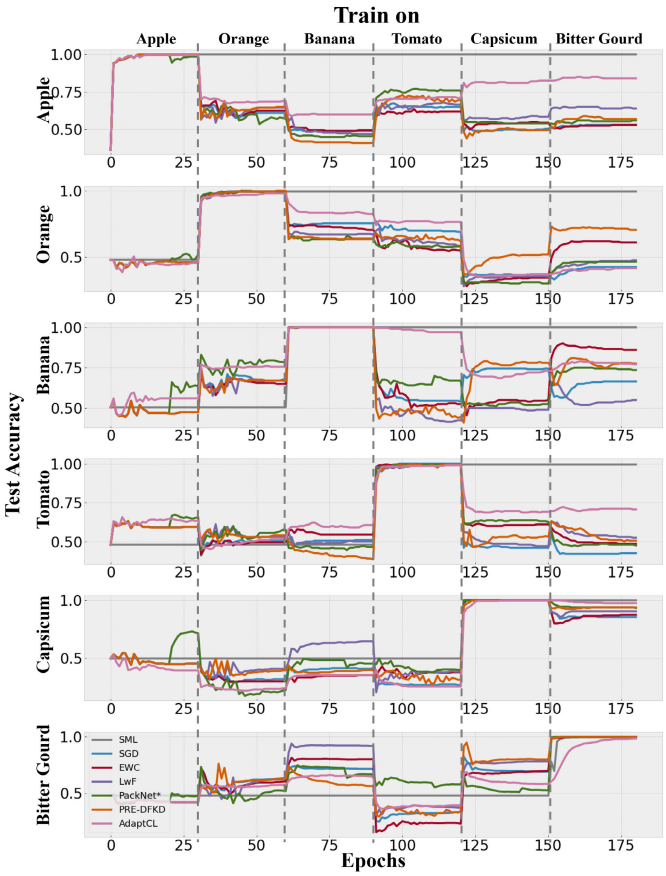

Fig. 4. Test accuracy comparison of continual learning methods on the large-scale, diverse binary-class food quality dataset. Our proposed method, AdaptCL, achieves higher AAC while consistently preventing catastrophic forgetting in real-world applications with heterogeneous data, outperforming other methods (best viewed in color).

the new dataset can make the network more robust to have higher accuracy during inference on the test dataset. Our model outperforms the baselines' AAC by 11.20% and is superior to using an SML on the few-shot, multiclass food quality dataset sequence with only 45% of SML's parameters. These results demonstrate the potential advantages of our model when encountering a continuous stream of smaller datasets.

a) Effect of training orders: To investigate the impact of training orders, we conduct experiments using different dataset sequences. Our method, AdaptCL, consistently achieves the best results in both forward and reverse training orders, as shown in Table II. Unlike baseline methods, the AAC of AdaptCL remains unaffected by the training sequence, while the final accuracy of methods, such as SGD, LwF, and EWC, is heavily influenced by the order of training. This can be attributed to AdaptCL's fine-grained pruning and task-agnostic parameter isolation, which minimize catastrophic forgetting and promote model generalization, enabling adaptation to new datasets regardless of their presentation order. These findings demonstrate the significant impact of training order on the performance of traditional methods, likely due to the tendency to overfit early datasets during training. The robustness of AdaptCL to training order positions it as a preferred method for domains requiring frequent learning and adaptation to new

TABLE I

PERFORMANCE EVALUATION OF CONTINUAL LEARNING METHODS IN TERMS OF AAC, BWT, FWT, AND NUMBER OF USED PARAMETERS ON THE LARGE-SCALE, DIVERSE BINARY-CLASS FOOD QUALITY DATASET

	AAC↑	BWT ↑	FWT ↑	Params ($\times 10^7$) ↓	Test Accuracy↑					
					Apple	Orange	Banana	Tomato	Gourd	Capsicum
SML	0.998	-	-	6.704	1.000	0.995	1.000	0.995	1.000	1.000
SGD	0.650	-0.419	0.033	1.117	0.530	0.425	0.665	0.425	0.855	1.000
LwF	0.683	-0.379	0.072	1.117	0.640	0.475	0.550	0.525	0.905	1.000
EWC	0.727	-0.326	0.058	1.117	0.530	0.610	0.860	0.485	0.875	1.000
PackNet*	0.695	-0.361	0.049	1.117	0.560	0.465	0.735	0.475	0.935	1.000
PRE-DFKD	0.749	-0.300	0.085	1.117	0.570	0.705	0.775	0.505	0.940	1.000
AdaptCL	**0.782**	**-0.252**	0.041	**1.091**	**0.840**	0.415	**0.770**	**0.705**	**0.975**	0.984

TABLE II

COMPARISON OF AAC, BWT, FWT, AND THE NUMBER OF USED PARAMETERS OF VARIOUS CONTINUAL LEARNING METHODS ON THE FEW-SHOT, MULTICLASS FOOD QUALITY DATASET WITH DIFFERENT TRAINING ORDERS

	Bread → Apple2						**Apple2 → Bread**					
	AAC↑	BWT↑	FWT↑	Params($\times 10^7$)↓	Test Accuracy↑		AAC↑	BWT↑	FWT↑	Params($\times 10^7$)↓	Test Accuracy↑	
					Bread	Apple					Apple	Bread
SML	0.989	-	-	2.235	0.978	1.000	0.989	-	-	2.235	1.000	0.978
SGD	0.774	-0.430	0.368	1.117	0.548	1.000	0.849	-0.281	0.452	1.117	0.719	0.978
LwF	0.782	-0.398	0.368	1.117	0.581	0.982	0.849	-0.281	0.452	1.117	0.719	0.978
EWC	0.763	-0.452	0.368	1.117	0.527	1.000	0.831	-0.316	0.452	1.117	0.684	0.978
PackNet*	0.894	-0.172	0.281	1.117	0.806	0.982	0.893	-0.193	0.398	1.117	0.807	0.978
PRE-DFKD	0.859	-0.086	0.404	1.117	0.892	0.825	0.867	-0.158	0.409	1.117	0.842	0.892
AdaptCL	**0.995**	**0.011**	0.316	**1.014**	**0.989**	**1.000**	**0.980**	**-0.018**	0.441	**1.005**	**0.982**	**0.978**

datasets, as it effectively avoids the limitations associated with traditional methods.

B. Performance on Datasets With Varied Complexities (Q1–Q3)

We evaluate the performance of AdaptCL on the DomainNet sequence, which is a heterogeneous classification dataset made up of images from different domains, each of varying size and complexity [40]. On the DomainNet dataset, rehearsal-based methods, such as LwF and PRE-DFKD, perform poorly, especially LwF, even lower than SGD without any continual learning method assistance. This is likely due to the large and disparate sizes of the three subsets in DomainNet, making it challenging to adjust simple knowledge distillation methods based on dataset size. Additionally, comparing SML with other methods on the last subset, clipart, we observe that learning models on sequential datasets can facilitate faster learning and FWT, resulting in higher test accuracy compared to SML. The AdaptCL does not achieve higher accuracy than SML on this subset due to pruning, which makes the model more parameter efficient, but simultaneously slows down the learning of new data because of insufficient model capacity. This issue can be solved by network expansion. As shown in Table III, AdaptCL outperforms the baselines, improving network performance by 18.24% in AAC and 44.79% in BWT compared to SGD, while beating the baselines CL methods by 9.70% in AAC and 30.69% in BWT, by using only 92.65% of their parameters. Despite the impressive results, gaps in accuracy persisted compared to using separate models for learning, primarily due to the complex

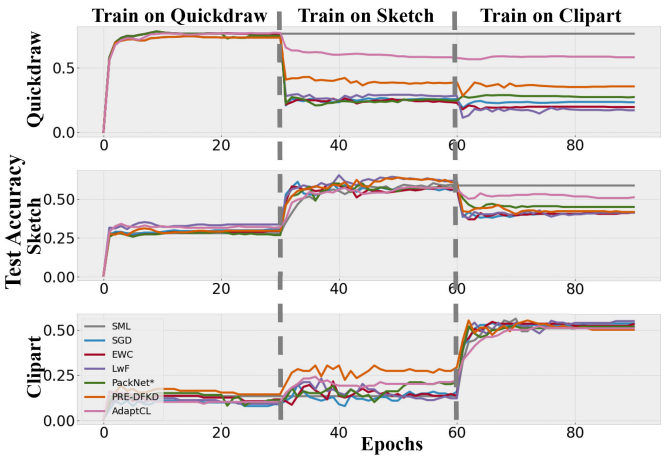

Fig. 5. Results of continual learning methods on the DomainNet that comprises datasets with heterogeneous complexity and size. AdaptCL achieves the best AAC and is the most robust to datasets with varied complexities and sizes (best viewed in color).

nature of the DomainNet data that demand increased model capacity to handle more complex information with significant distribution shifts within the dataset. From Fig. 5, we can see that even with significant differences in data, clipart, sketch, and quickdraw can rely on FWT to achieve faster learning. In the context of continual learning, old datasets can improve the accuracy of new datasets, making CL methods more accurate than using SML to learn new data. Among the methods evaluated, rehearsal is the most effective in promoting faster learning, while AdaptCL excelled in accuracy retention.

TABLE III

RESULTS OF DIFFERENT CONTINUAL LEARNING METHODS ON THE DOMAINNET DATASET, INCLUDING THEIR AAC, BWT, FWT, AND THE NUMBER OF USED PARAMETERS. OUR PROPOSED METHOD, ADAPTCL, DEMONSTRATES THE BEST AAC AND BWT, INDICATING ITS ABILITY TO HANDLE DATASETS WITH HETEROGENEOUS DATASET SIZE AND COMPLEXITY

	AAC ↑	BWT ↑	FWT ↑	Params($\times 10^7$) ↓	Test Accuracy↑		
					Quickdraw	Sketch	Clipart
SML	0.624	-	-	3.404	0.774	0.573	0.524
SGD	0.449	-0.220	0.209	1.135	0.394	0.414	0.539
LwF	0.448	-0.237	0.103	1.135	0.381	0.411	0.552
EWC	0.457	-0.203	0.103	1.135	0.421	0.414	0.536
PackNet*	0.484	-0.180	0.098	1.135	0.483	0.448	0.521
PRE-DFKD	0.461	-0.206	0.109	1.135	0.464	0.416	0.504
AdaptCL	**0.531**	**-0.131**	0.106	**1.051**	**0.570**	**0.510**	0.512

C. Performance on Datasets With Varied Similarities (Q1 and Q4)

1) Dissimilar MNIST Variants: We screen two sets of MNIST variants to compose similar and dissimilar sequences, with the dissimilar sequence as MNIST, Permuted MNIST, and Inverted MNIST. Regarding the dissimilar MNIST variant sequence, AdaptCL significantly improves the network's AAC by 28.14% and alleviates forgetting (BWT) by 94.50% (Table IV). It outperforms baselines by 15.03% in AAC and 91.30% in BWT on this sequence. Compared to SML where separate models are trained for each task, AdaptCL achieves comparable AAC while utilizing only 31.20% of SML's parameters. Our method's ability to minimize forgetting while learning dissimilar datasets, its parameter efficiency, and generalization contributes to its effectiveness.

Rehearsal and regularization-based methods, such as EWC, LwF, and PRE-DFKD, perform poorly on the dissimilar MNIST variants dataset due to the vast data amount and dissimilarity between datasets. Parameter isolation-based methods, such as PackNet*, and our method, AdaptCL, demonstrate significant advantages on this dataset. Training with plain SGD leads to catastrophic forgetting and a performance decline of at least 50% (Fig. 6). EWC slows down the performance decline initially, but it deteriorates over time. While PackNet* shows some improvement through pruning during learning on Dataset A, it is not as effective as AdaptCL in inhibiting catastrophic forgetting. AdaptCL, with a fixed neural network, adapts to new datasets while maintaining high performance on previous datasets without significant forgetting.

2) More Similar MNIST Variants: In a similar MNIST Variant sequence consisting of MNIST, Permuted MNIST, and Rotated MNIST, our method maintains the highest accuracy, outperforming baselines by 21.80% while using fewer parameters (Table V). AdaptCL achieves significant FWT on similar datasets with minimal catastrophic forgetting. Although it may not achieve the best accuracy on particularly similar datasets, such as MNIST and Rotated MNIST, our model consistently performs well on datasets with heterogeneous similarity, avoiding overfitting to similar datasets. As dataset similarity increases, EWC suppresses catastrophic forgetting and achieves higher AAC compared to SGD, which differs from results on dissimilar datasets. LwF and PRE-DFKD

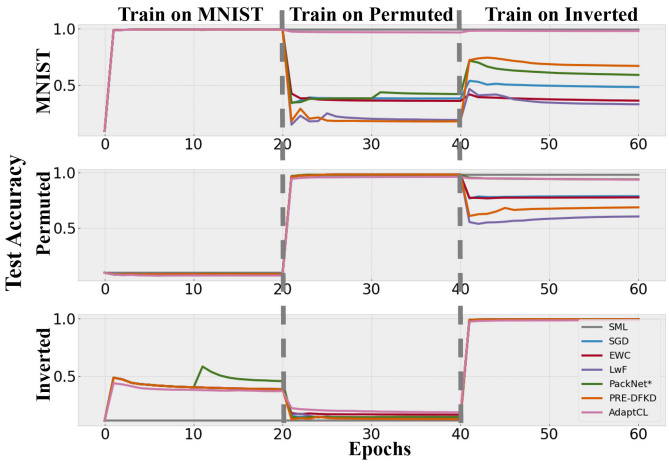

Fig. 6. Test accuracy comparison of continual learning methods on three dissimilar MNIST variant datasets. Compared with using SML, AdaptCL achieved comparable AAC results while using only 31.20% of SML's parameters. AdaptCL's ability to achieve minimal forgetting while learning dissimilar datasets, coupled with its parameter efficiency, establishes the effectiveness of our approach (best viewed in color).

struggle to balance dataset importance, leading to significantly lower accuracy on Permuted MNIST, indicating their unsuitability for large datasets with varying similarities. Our method demonstrates robust performance in such scenarios.

In Fig. 7, all continual learning methods' inference accuracies on Rotated MNIST initially rise due to its similarity to MNIST. AdaptCL minimizes catastrophic forgetting and maintains high accuracy and generalization. Other CL methods, compared to SML, enhance model generalization, but still experience catastrophic forgetting while learning on Permuted MNIST. Notably, rehearsal-based methods, such as LwF and PRE-DFKD, exacerbate catastrophic forgetting on Permuted MNIST when learning Rotated MNIST due to their similarity to the initial MNIST dataset.

D. Performance in Class-Incremental Setting (Q1, Q3, and Q4)

The methods are evaluated in challenging class-incremental learning settings, with heterogeneous classes, as shown in Table VI. These datasets included class overlap and variations in the number of classes, introducing differences in data similarity and dataset size. This setting poses a significant

TABLE IV
COMPARISON OF AAC, BWT, FWT, AND THE NUMBER OF USED PARAMETERS OF DIFFERENT CONTINUAL LEARNING METHODS ON THE DISSIMILAR MNIST VARIANTS DATASET

	AAC↑	BWT↑	FWT↑	Params ($\times 10^7$)↓	Test Accuracy↑		
					MNIST	Permuted MNIST	Inverted MNIST
SML	0.989	-	-	3.352	0.993	0.980	0.993
SGD	0.755	-0.351	0.006	1.117	0.483	0.787	0.994
LwF	0.643	-0.521	-0.002	1.117	0.331	0.604	0.994
EWC	0.753	-0.354	0.009	1.117	0.363	0.776	0.993
PackNet*	0.841	-0.222	0.009	1.117	0.591	0.939	0.993
PRE-DFKD	0.784	-0.274	0.011	1.117	0.723	0.686	0.943
AdaptCL	**0.967**	**-0.019**	**0.023**	**1.046**	**0.980**	**0.936**	**0.986**

TABLE V
COMPARISON OF AAC, BWT, FWT, AND THE NUMBER OF USED PARAMETERS OF DIFFERENT CONTINUAL LEARNING METHODS ON MORE SIMILAR MNIST VARIANT DATASETS

	AAC↑	BWT↑	FWT↑	Params ($\times 10^7$)↓	Test Accuracy↑		
					MNIST	Permuted MNIST	Rotated MNIST
SML	0.989	-	-	3.352	0.993	0.980	0.992
SGD	0.884	-0.156	0.077	1.117	0.994	0.666	0.993
LwF	0.729	-0.391	0.023	1.117	0.993	0.204	0.991
EWC	0.889	-0.149	0.088	1.117	0.994	0.681	0.992
PackNet*	0.938	-0.076	0.101	1.117	0.994	0.830	0.991
PRE-DFKD	0.822	-0.240	0.179	1.117	0.991	0.488	0.988
AdaptCL	**0.958**	**-0.032**	**0.339**	**1.044**	0.990	**0.900**	0.985

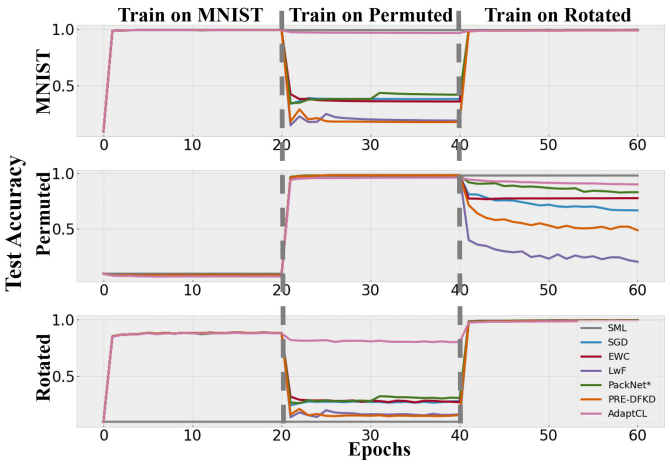

Fig. 7. Visualization of each method's test accuracy training on more similar MNIST variant datasets. Still, AdaptCL retains the best performance compared with other CL methods. Compared with using SML, AdaptCL can increase the accuracy on similar unseen datasets via FWT (best viewed in color).

challenge for continual learning, as similar inputs leading to distinct classes that are assigned to different output layers, often causing rapid forgetting of previously learned classes within a single epoch.

Due to high intensity of catastrophic forgetting and large dataset sizes, techniques, such as rehearsal and regularization methods, struggle to maintain accuracy effectively, as demonstrated in Table VI. Their performance is only marginally superior to SGD. Notably, when employing equal-proportional pruning, PackNet* exhibits poorer results in class-incremental settings in comparison to domain-incremental ones. This discrepancy is attributed to the significant impact of the pruning ratio on accuracy retention in class-incremental settings. In this context, catastrophic forgetting is prevalent across all methods and notably higher than in the domain-incremental setting, even with a reduced number of training epochs. AdaptCL, on the other hand, stands out by significantly preserving accuracy, achieving an impressive 37.27% higher AAC and reducing forgetting by 34.96% when compared to SGD. This strong performance underscores AdaptCL's potential in mitigating catastrophic forgetting in task-agnostic settings.

E. Performance on Different Networks (Q1)

AdaptCL can be applied to neural networks with fully connected, recurrent, or convolutional layers. We also apply our method to the shallow LeNet-5 network and more complex VGG-16 to test its performance over different networks. For LeNet-5 (Table VII), we observed a significant reduction in catastrophic forgetting compared to ResNet-18 when using AdaptCL. It achieved the second-highest AAC and the best BWT, trailing only PRE-DFKD, likely due to its lower parameter usage. Even with limited network capacity, AdaptCL outperformed other methods in mitigating catastrophic forgetting. Rehearsal-based approaches also showed promise in cases of insufficient model parameters and capacity.

On VGG-16 (Table VII), AdaptCL exhibited its effectiveness on MNIST variants datasets, surpassing baseline methods with the best AAC for AAC and BWT. Notably, most continual learning methods yielded negative FWT when applied to VGG-16. In the case of Permuted MNIST, VGG-16's performance did not improve after learning the MNIST dataset, possibly due to dissimilar image structures and VGG-16's

TABLE VI

COMPARISON OF AAC, BWT, FWT, AND THE NUMBER OF USED PARAMETERS OF DIFFERENT CONTINUAL LEARNING METHODS IN THE CLASS-INCREMENTAL SETTING

	AAC↑	BWT↑	FWT↑	Params ($\times 10^7$)↓	Test Accuracy↑		
					Class 0,1,2,3	Class 2,3,4,5,6	Class 0,4,7,8,9
SML	0.996	-	-	3.352	0.998	0.995	0.994
SGD	0.475	-0.781	0.208	1.117	0.231	0.199	0.995
LwF	0.476	-0.781	0.208	1.117	0.234	0.199	0.995
EWC	0.477	-0.780	0.208	1.117	0.208	0.208	0.996
PackNet*	0.475	-0.781	0.207	1.117	0.231	0.200	0.995
PRE-DFKD	0.476	-0.780	0.198	1.117	0.234	0.199	0.996
AdaptCL	**0.652**	**-0.508**	0.205	**1.089**	**0.429**	**0.543**	0.983

TABLE VII

RESULTS OF DIFFERENT CONTINUAL LEARNING METHODS APPLIED ON LENET-5 AND VGG-16, INCLUDING THEIR AAC, BWT, FWT, AND THE NUMBER OF USED PARAMETERS ON DATASETS ($D1$) MNIST, ($D2$) PERMUTED MNIST, AND ($D3$) INVERTED MNIST

	LeNet-5							**VGG-16**						
	AAC↑	BWT↑	FWT↑	Params	Test Accuracy↑			AAC↑	BWT↑	FWT↑	Params	Test Accuracy↑		
				($\times 10^4$)↓	D1	D2	D3				($\times 10^7$)↓	D1	D2	D3
SML	0.968	-	-	18.51	0.948	0.978	0.977	0.990	-	-	4.496	0.995	0.980	0.995
SGD	0.479	-0.740	0.049	6.171	0.233	0.220	0.983	0.730	-0.390	-0.004	1.499	0.441	0.753	0.994
LwF	0.542	-0.646	0.045	6.171	0.243	0.398	0.984	0.763	-0.342	-0.342	1.499	0.532	0.762	0.995
EWC	0.098	-0.839	0.003	6.171	0.098	0.098	0.098	0.747	-0.363	0.021	1.499	0.455	0.792	0.994
PackNet*	0.533	-0.644	-0.018	6.171	0.141	0.482	0.975	0.802	-0.281	-0.015	1.499	0.527	0.885	0.885
PRE-DFKD	0.579	-0.479	0.027	6.171	0.421	0.332	0.985	0.865	-0.182	0.010	1.499	0.841	0.767	0.989
AdaptCL	0.567	**-0.457**	0.041	**6.162**	0.220	**0.740**	0.742	**0.963**	**-0.027**	-0.003	**1.396**	**0.979**	**0.922**	0.987

inability to recognize permutations, whereas ResNet-18 performed slightly better in this scenario.

F. Ablation Study (Q3 and Q4)

1) Parameter Isolation Ratio: To validate our intuition, we analyze and visualize the pruning ratio of the model in different datasets. Fig. 8(a) displays the epochwise accuracy changes of the sparse network compared to fine-grained data-driven pruning and the dense network without pruning, along with the corresponding model remaining ratio during training. Our method dynamically and adaptively learns the model remaining ratios during training on each dataset, rather than manually setting fixed ratios as in other pruning methods. Additionally, fine-grained data-driven pruning enables a highly sparse pruned network to achieve the same accuracy as a dense network. Furthermore, Fig. 8(b) illustrates the change in the remaining ratios of the ResNet-18 model for each dataset of MNIST variants at each epoch. Fig. 8(b) demonstrates that it is possible to fit the new dataset without sacrificing the accuracy of the old dataset by adding only a few parameters, even when there are significant differences in data distribution between the old and new datasets. Consequently, manually assigning the same parameter ratio to all datasets is not reasonable.

2) Parameter Execution Pattern: To analyze the parameter reuse in AdaptCL, we visualize the pattern of parameter execution during training and the proportion of each layer in the neural network occupied progressively by different datasets. Fig. 9(a) displays the pattern of parametric ignition of the first Conv2d layer (flattened) on the MNIST variants. Fig. 9(b) presents the proportion of parameters occupied during adaptive

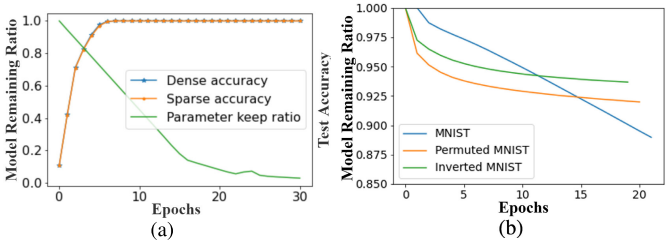

Fig. 8. (a) Model on ResNet-18 remaining ratio and sparse accuracy compared with dense accuracy, using $\alpha = 10^{-4}$. (b) Change of model remaining ratio during training on MNIST variants (best viewed in color).

learning in each convolution and fully connected layer of ResNet-18. From these figures, we observe that the parameters activated during previous task training remain unchanged when learning a new dataset. Additionally, we notice that some previously unused parameters (black) become activated during the new dataset training, contributing to the overall generalization of the network.

3) Hyperparameters: The hyperparameter α controls the intensity of pruning in the loss function during training, and it plays a crucial role in determining the final balance of model sparsity. We explore the effect of different α values, ranging from 10^{-3} to 10^{-7}, in our experiments. The value of α is mainly determined based on the data amount and training epochs, as pruning is primarily performed in each iteration of the training step. We aim to ensure that the product of the total number of iterations (e.g., image numbers multiplied by epochs) and α is approximately 1, allowing for efficient and effective pruning. To investigate the influence of

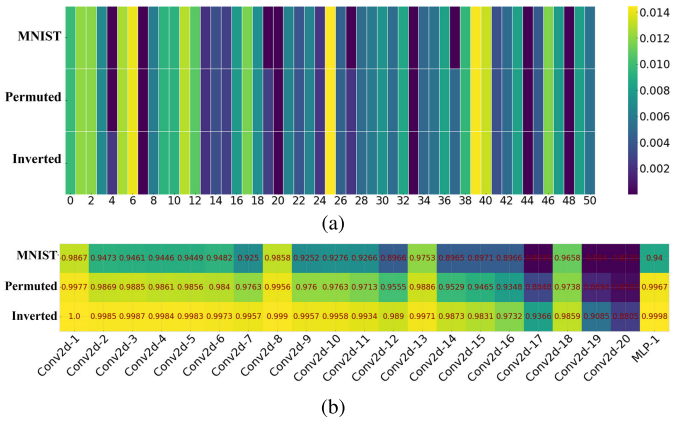

Fig. 9. Illustration of the parameter execution patterns for continually trained models on MNIST variants. (a) Heatmap showcases the firing probability of the xth parameters within the first Conv2d layer demonstrated in the x-axis; it illustrates that subsequent datasets maintain and reuse the previous parameters and generalize by adding new connections to them. (b) Heatmap shows the utilization ratio of different layers.

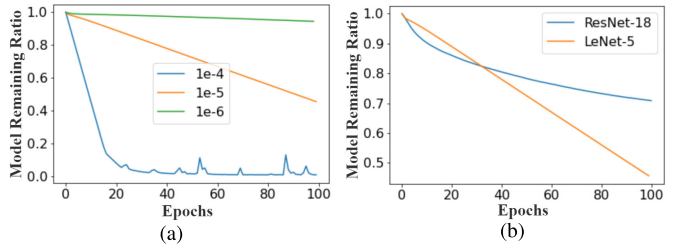

Fig. 10. (a) Pruning effect in ResNet-18 for different values of hyperparameter α. (b) Change of model remaining ratio with ResNet-18 and LeNet-5 for different values of α (best viewed in color).

different α values on the model's pruning intensity, we conduct experiments using different models, such as ResNet-18 and LeNet-5. Additionally, we examine the change in the model's remaining ratio for various α values. The results are shown in Fig. 10, providing insights into the relationship between α, pruning intensity, and the model's remaining parameters for each architecture.

VII. Conclusion

In this study, we aimed to tackle the challenge of managing heterogeneous datasets in continual learning. We observed unstable performance of rehearsal, regularization, and non-adaptive parameter isolation-based methods when dealing with multiple heterogeneous datasets in experiments. Inspired by the neural-reuse principle of human brains, we presented AdaptCL, a novel continual learning algorithm. Our proposed method effectively addresses the challenge of managing heterogeneous datasets in continual learning, outperforming existing approaches in terms of robustness and achieving higher AAC. Additionally, AdaptCL proves to be a proficient few-shot learner, exhibiting the capability to make generalizations based on limited examples similar to human cognitive abilities. By introducing fine-grained data-driven pruning and task-agnostic parameter isolation, we address catastrophic forgetting and demonstrate the effectiveness of AdaptCL across heterogeneous datasets in diverse applications. Our work contributes to the field by providing a novel algorithm that improves performance in heterogeneous dataset scenarios. While our approach is computationally efficient, we acknowledge the limitation of reduced learning efficiency with insufficient model capacity. To address this, future work will focus on introducing network expansion techniques to enhance scalability on a growing number of heterogeneous datasets.

References

[1] J. He and F. Zhu, "Online continual learning for visual food classification," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. Workshops (ICCVW)*, Oct. 2021, pp. 2337–2346.

[2] H. Zhao, H. Wang, Y. Fu, F. Wu, and X. Li, "Memory-efficient class-incremental learning for image classification," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 33, no. 10, pp. 5966–5977, Oct. 2022.

[3] J. He and F. Zhu, "Exemplar-free online continual learning," in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, Oct. 2022, pp. 541–545.

[4] D. Abati, J. Tomczak, T. Blankevoort, S. Calderara, R. Cucchiara, and B. E. Bejnordi, "Conditional channel gated networks for task-aware continual learning," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 3930–3939.

[5] Y. Zhao, D. Saxena, and J. Cao, "Memory-efficient domain incremental learning for Internet of Things," in *Proc. 20th ACM Conf. Embedded Networked Sensor Syst.*, Nov. 2022, pp. 1175–1181.

[6] A. Pascual-Leone, A. Amedi, F. Fregni, and L. B. Merabet, "The plastic human brain cortex," *Annu. Rev. Neurosci.*, vol. 28, no. 1, pp. 377–401, Jul. 2005.

[7] M. V. Johnston, "Plasticity in the developing brain: Implications for rehabilitation," *Develop. Disabilities Res. Rev.*, vol. 15, no. 2, pp. 94–101, Jan. 2009.

[8] M. L. Anderson, "Neural reuse: A fundamental organizational principle of the brain," *Behav. Brain Sci.*, vol. 33, no. 4, pp. 245–266, Aug. 2010.

[9] D. Lopez-Paz and M. Ranzato, "Gradient episodic memory for continual learning," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 30, 2017, pp. 1–10.

[10] A. Chaudhry, M. Ranzato, M. Rohrbach, and M. Elhoseiny, "Efficient lifelong learning with A-GEM," 2018, *arXiv:1812.00420*.

[11] J. Peng, D. Ye, B. Tang, Y. Lei, Y. Liu, and H. Li, "Lifelong learning with cycle memory networks," *IEEE Trans. Neural Netw. Learn. Syst.*, early access, pp. 1–14, Jul. 2023, doi: 10.1109/TNNLS.2023.3294495.

[12] S. Ho, M. Liu, L. Du, L. Gao, and Y. Xiang, "Prototype-guided memory replay for continual learning," *IEEE Trans. Neural Netw. Learn. Syst.*, pp. 1–11, Mar. 2023, Art. no. 10058177, doi: 10.1109/TNNLS.2023.3246049.

[13] Z. Li and D. Hoiem, "Learning without forgetting," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 12, pp. 2935–2947, Dec. 2018.

[14] A. Rosasco, A. Carta, A. Cossu, V. Lomonaco, and D. Bacciu, "Distilled replay: Overcoming forgetting through synthetic samples," in *Proc. 1st Int. Workshop Continual Semi-Supervised Learn.* Cham, Switzerland: Springer, 2022, pp. 104–117.

[15] J. Sun, S. Wang, J. Zhang, and C. Zong, "Distill and replay for continual language learning," in *Proc. 28th Int. Conf. Comput. Linguistics*, 2020, pp. 3569–3579.

[16] S.-A. Rebuffi, A. Kolesnikov, G. Sperl, and C. H. Lampert, "ICaRL: Incremental classifier and representation learning," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 5533–5542.

[17] K. Binici, S. Aggarwal, N. T. Pham, K. Leman, and T. Mitra, "Robust and resource-efficient data-free knowledge distillation by generative pseudo replay," in *Proc. AAAI Conf. Artif. Intell.*, vol. 36, no. 6, 2022, pp. 6089–6096.

[18] K. James et al., "Overcoming catastrophic forgetting in neural networks," *Proc. Nat. Acad. Sci. USA*, vol. 114, no. 13, pp. 3521–3526, Mar. 2017.

[19] F. Huszár, "On quadratic penalties in elastic weight consolidation," 2017, *arXiv:1712.03847*.

[20] Y. Liu, X. Hong, X. Tao, S. Dong, J. Shi, and Y. Gong, "Model behavior preserving for class-incremental learning," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 34, no. 10, pp. 7529–7540, Feb. 2022.

[21] J. Schwarz et al., "Progress & compress: A scalable framework for continual learning," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 4528–4537.

[22] H. Li, P. Barnaghi, S. Enshaeifar, and F. Ganz, "Continual learning using Bayesian neural networks," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 9, pp. 4243–4252, Sep. 2021.

[23] G.-M. Park, S.-M. Yoo, and J.-H. Kim, "Convolutional neural network with developmental memory for continual learning," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 6, pp. 2691–2705, Jun. 2021.
[24] J. Serra, D. Suris, M. Miron, and A. Karatzoglou, "Overcoming catastrophic forgetting with hard attention to the task," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 4548–4557.
[25] A. Ororbia, A. Mali, C. L. Giles, and D. Kifer, "Continual learning of recurrent neural networks by locally aligning distributed representations," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 31, no. 10, pp. 4267–4278, Oct. 2020.
[26] A. A. Rusu et al., "Progressive neural networks," 2016, *arXiv:1606.04671*.
[27] R. Ma, Q. Wu, K. N. Ngan, H. Li, F. Meng, and L. Xu, "Forgetting to remember: A scalable incremental learning framework for cross-task blind image quality assessment," *IEEE Trans. Multimedia*, vol. 25, pp. 8817–8827, Feb. 2023, doi: 10.1109/TMM.2023.3242143.
[28] J. Xu and Z. Zhu, "Reinforced continual learning," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 31, 2018, pp. 1–10.
[29] T. Adel, H. Zhao, and R. E. Turner, "Continual learning with adaptive weights (CLAW)," 2019, *arXiv:1911.09514*.
[30] C. Fernando et al., "PathNet: Evolution channels gradient descent in super neural networks," 2017, *arXiv:1701.08734*.
[31] J. Rajasegaran, M. Hayat, S. Khan, F. S. Khan, and L. Shao, "Random path selection for continual learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019, pp. 1–11.
[32] Z. Ke, B. Liu, and X. Huang, "Continual learning of a mixed sequence of similar and dissimilar tasks," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 33, 2020, pp. 18493–18504.
[33] A. Rosenfeld and J. K. Tsotsos, "Incremental learning through deep adaptation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 42, no. 3, pp. 651–663, Mar. 2020.
[34] S. Golkar, M. Kagan, and K. Cho, "Continual learning via neural pruning," 2019, *arXiv:1903.04476*.
[35] A. Mallya and S. Lazebnik, "PackNet: Adding multiple tasks to a single network by iterative pruning," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 7765–7773.
[36] M. De Lange et al., "A continual learning survey: Defying forgetting in classification tasks," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 44, no. 7, pp. 3366–3385, Jul. 2022.
[37] R. Aljundi, F. Babiloni, M. Elhoseiny, M. Rohrbach, and T. Tuytelaars, "Memory aware synapses: Learning what (not) to forget," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, Sep. 2018, pp. 139–154.
[38] J. Liu, Z. Xu, R. Shi, R. C. C. Cheung, and H. K. H. So, "Dynamic sparse training: Find efficient sparse network from scratch with trainable masked layers," 2020, *arXiv:2005.06870*.
[39] Z. Xu and R. C. C. Cheung, "Accurate and compact convolutional neural networks with trained binarization," 2019, *arXiv:1909.11366*.
[40] X. Peng, Q. Bai, X. Xia, Z. Huang, K. Saenko, and B. Wang, "Moment matching for multi-source domain adaptation," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 1406–1415.
[41] L. Bottou, "Stochastic gradient learning in neural networks," *Proc. Neuro-Nimes*, vol. 91, no. 8, p. 12, Nov. 1991.

Yuqing Zhao received the B.E. degrees in software engineering and in life sciences from Sichuan University, Chengdu, China, in 2020. She is currently pursuing the Ph.D. degree in computing with The Hong Kong Polytechnic University (PolyU), Hong Kong.

Her main research interests include continual learning, computer vision, and automated machine learning.

Divya Saxena (Member, IEEE) received the Ph.D. degree in computer science and engineering (CSE) from the Indian Institute of Technology (IIT) Roorkee, Roorkee, India, in 2017.

She worked as a Research Fellow and a Post-Doctoral Fellow with the Department of Computing and UBDA, The Hong Kong Polytechnic University (PolyU), Hong Kong, where she is currently a Research Assistant Professor with the Department of Computing. Her research interests include generative artificial intelligence (AI), continual learning, spatiotemporal data mining, as well as mobile sensing and computing.

Jiannong Cao (Fellow, IEEE) received the Ph.D. degree in computer science from Washington State University, Pullman, WA, USA, in 1990.

He is currently the Otto Poon Charitable Foundation Professor of data science and the Chair Professor of distributed and mobile computing with the Department of Computing, The Hong Kong Polytechnic University (PolyU), Hong Kong, where he is also the Dean of the Graduate School, the Director of the Research Institute for Artificial Intelligence of Things, and the Director of the Internet and Mobile Computing Laboratory. His research interests include distributed systems and blockchain, big data and machine learning, wireless sensing and networking, and mobile cloud and edge computing.