IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 8 August 2023, accepted 13 September 2023, date of publication 18 September 2023,
date of current version 27 September 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3316255

RESEARCH ARTICLE

CLiCK: Continual Learning Exploiting Intermediate Network Models With A Slack Class

HYEJIN KIM[1], SEUNGHYUN YOON[2], (Member, IEEE), AND HYUK LIM[2], (Member, IEEE)
[1]Gwangju Institute of Science and Technology (GIST), Gwangju 61005, Republic of Korea
[2]Korea Institute of Energy Technology (KENTECH), Naju 58217, Republic of Korea

Corresponding author: Hyuk Lim (hlim@kentech.ac.kr)

This work was supported in part by the Institute of Information and Communications Technology Planning and Evaluation (IITP) Grant funded by the Korea Government [Ministry of Science and ICT (MSIT)] through the Privacy Risk Analysis and Response Technology Development for AI Systems under Grant 2021-0-00379, and in part by the Korea Institute of Energy Technology (KENTECH) Research under Grant KRG 2022-01-013.

ABSTRACT We propose a continual learning (CL) method (called CLiCK), a hybrid of an architecture-based approach that increments a model when it detects that the dataset characteristics have changed significantly, and a rehearsal-based approach that exploits an episodic memory to store past dataset samples. The proposed CLiCK makes the final decision by taking the ensemble of the inference results for the current and a series of past models. A novelty of CLiCK is to introduce a concept for a slack class, which is an auxiliary class to represent unseen or undetermined classes that do not belong to the current dataset. Because the models trained with a slack class have the capability to differentiate between the classes that they were trained on and unseen classes, the inference results of the models that do not have knowledge about input can be automatically neglected in the final decision. Our experiments show that the proposed CLiCK achieves performance comparable to joint learning, which uses the entire dataset for each task, in domain-incremental scenarios on the MNIST dataset. In class-incremental scenarios on the MNIST and CIFAR-100 datasets, CLiCK outperforms other existing CL methods significantly.

INDEX TERMS Continual learning, episodic memory, intermediate models, slack class.

I. INTRODUCTION

One of the primary challenges in artificial intelligence is to create models that are adaptable and flexible enough to handle new tasks and datasets in a given domain as they grow over time. In traditional offline learning, creating a model that performs well on a specific task requires training the model on the task's dataset. If a new task is introduced, the training process is reperformed with the new dataset. However, when the learning model starts to be trained with only the new task's dataset, its knowledge is gradually skewed towards the new tasks leading to a decrease in performance on previous tasks. This phenomenon is known as catastrophic forgetting (CF) [1]. To avoid CF, the training process must be repeated using a comprehensive dataset that includes data from all previous and current tasks to maintain the model's performance on all previous and current tasks. Unfortunately, this training process using the entire dataset can be computationally expensive and might not be feasible or allowable in data-intensive applications.

Continual learning (CL) aims to improve the learning capacity of a model by allowing it to accumulate and expand its knowledge in a dynamic and ever-changing data environment without forgetting knowledge about previously encountered dataset [1], [2]. To prevent the model from forgetting information about previous tasks as it learns new information, several methods have been proposed in the literature, such as rehearsal strategy, dynamic model architecture, parameter updating regularization, and multiple single-class classifications. One popular approach among these is the rehearsal-based or experience replay method, which stores a subset of the previous training data in a memory buffer for use during the training of new tasks. CL in the continuously-changing dataset stream environments is much more challenging because it is difficult to define a distinct task from the data streams. In these general scenarios, CL models

The associate editor coordinating the review of this manuscript and approving it for publication was Yiqi Liu.

must effectively adapt to not only new classes but also varying data distributions of existing classes. In addition, the models are required to perform the learning for the dataset changes appropriately even without explicit task boundaries.

We propose a novel approach called CLiCK, a **C**ontinual **L**earning framework for exploiting **i**ntermediate network models with a Sla**CK** class. The proposed CLiCK mitigates the CF issues by using the integration of a series of previous models (called **intermediate models**) that have been already generated using past datasets and a new model for the current dataset. This approach of integrating all the models is based on the assumption that the knowledge of past datasets is already retained well in the previous intermediate models. In the general CL scenarios, if the error rate of inference result increases, it implies that the distribution of the data stream is changing and becomes different from what the previous models were trained for. When the error rate exceeds a certain threshold, the CLiCK generates a new model and trains it using the currently available dataset. The inference of CLiCK is performed by integrating the newly-generated model and all the previous intermediate models.

The proposed CLiCK is a hybrid approach of architecture-based CL [3], [4], [5], [6] and rehearsal-based CL [7], [8], [9], [10], [11], [12], [13]. **The architecture-based CL approaches** attempt to append a new set of network model parameters [3] or to reuse a subset of existing model parameters (e.g., subnetwork) [4], [5], [6] for a new task. These architecture-based approaches can mitigate CF effectively by freezing/reusing the model parameters for the previous tasks. However, their inference requires the knowledge/inference of the task identification that the model/subnetwork was trained for. Our CLiCK increments a model when it detects that the dataset characteristics have changed significantly, but it does not require any task identification for inference. Instead, CLiCK simply integrates the inference results of all the intermediate models without exploiting model-task mapping information, and thus the CLiCK is suitable for these general CL scenarios. **The rehearsal-based CL approaches** take a strategy to maintain episodic memory for mitigating CF issues. They exploit a portion of data belonging to past tasks stored in the episodic memory to update the CL model for new tasks. It is highly desirable to fill the episodic memory with the most representative samples belonging to the past tasks [7], [13]. For a new task, the model is trained to minimize a weighted sum of a loss function for new data samples and for past data samples stored in the episodic memory [8], a loss function with a constraint to regulate the weight updating [9], [10], a modified loss function with additional terms to regulate the weight updating [11]. These approaches attempt to balance the current and past learning performance, and thus they have difficulty avoiding the tradeoff relationship of CF. In general, rehearsal-based approaches tend to experience a decrease in learning performance as the number of tasks in memory increases, due to the decrease in the number of samples for each task. Under CLiCK, the training of each intermediate model focuses on the learning performance for the current training data stream rather than making a balance of learning performance between the current and past data streams. Furthermore, as all the samples in episodic memory are mapped into a single synthetic class called slack class, CLiCK does not require a large-sized memory buffer. In a nutshell, CLiCK is an architecture-based CL method that appends the models gradually and a rehearsal-based method that exploits a memory. However, the memory of CLiCK is used for the discrimination purpose of deciding whether an input belongs to one that has been unseen during the training process.

A novelty of CLiCK is the introduction of a concept for a **slack class**. Under CLiCK, as there are multiple intermediate models, we have a question for CL, ''which model is to be used as the best one for the final decision?'' or ''how can the models be combined for the decision?''. If we know which model is the best one, it would be possible to nullify the decision results of the others. Alternatively, if each model can identify whether or not it can handle an input, such a model that has no knowledge of the input can nullify its decision by itself during the integration of the final decision. To this end, each model is trained with its own dataset and synthetic data samples with an auxiliary class (i.e., slack class). The slack class of a task represents unseen or undetermined datasets that do not belong to the current task. The data samples of the slack class are readily generated from the memory that stores the data samples of past tasks. Once a model is trained together with a slack class, it has the capability to differentiate between the datasets that it was trained on and unseen/undetermined classes for input data. If a classification result of an intermediate model falls into a slack class, it implies that the model does not have any knowledge about the input, and the model can be excluded. As a result, the classification of input data can be made by the models that have knowledge of the input data in CL. The key contributions of this paper are summarized as follows:

- The proposed CLiCK exploits the existing past models for the classification of past tasks and enables the training of the current dataset to focus on a new dataset.
- The concept of the slack class is introduced to make the intermediate models ready to be used with new tasks in the future.
- The generation of slack class samples requires the retention of only a small subset of non-relevant data for the current task, reducing the need for a large memory buffer size.
- CLiCK can be readily incorporated with any type of network model. The performance of the proposed method has been evaluated using VGG-11, ResNet-18, and GoogLeNet in the challenging CL scenarios with the MNIST and CIFAR-100 datasets.

II. RELATED WORK

A. REHEARSAL-BASED CLs

Rebuffi et al. [7] proposed a class incremental learning method called iCaRL. When new classes are added, iCaRL stores representative samples per class in memory and

determines the classes of the newly added data as one with the nearest feature mean value among the stored data samples. Shin et al. [8] proposed incorporating a generative replay (GR) model into CL to alleviate the CF problem. GR produces samples from previous tasks and combines them with the input samples of the current task. Lopez-Paz and Ranzato [9] proposed a memory-based CL method, (called gradient episodic memory, GEM) for continual learning under the assumption that the input of the model is the stream of data with the non-independent identically distributed (non-IID) distribution. Chaudhry et al. [10] proposed a lightweight version of GEM called Averaged GEM (A-GEM), which mitigates the computational complexity of GEM by replacing the constraints for individual past tasks with that for the average loss over past tasks. Buzzega et al. [11] proposed the Dark Experience Replay (DER) for CL in a boundary-free environment, which relies on knowledge distillation and regularization exploiting rehearsal memory buffer to approximate the entire training trajectory. They also proposed DER++, which additionally leverages the ground truth labels, for a case where the distribution of the input stream suddenly changes. Caccia et al. [12] proposed the Experience Replay with Asymmetric Metric Learning (ER-AML), which uses different similarity-based loss functions for the newly-incoming data stream and buffered data in rehearsal memory for past tasks. Prabhu et al. [14] proposed a simple and greedy episodic memory-based method (called GDumb), which achieves a good performance in the class-incremental scenarios of the CL environment when the memory buffer is sufficiently large. Compared to the existing methods, the proposed CLiCK uses memory to store past datasets but uses it differently in that all the samples in the memory are used for generating synthetic samples belonging to the slack class.

B. ARCHITECTURE-BASED CLs

Rusu et al. [3] proposed a progressive neural network (PNN), which initializes a new column network for each new task. The other is model pruning, which temporarily disables some portions of model parameters. Mallya and Lazebnik [4] proposed a dynamic model expansion (called PackNet), which allocates multiple tasks to a single network by iterative pruning. PackNet sequentially prunes the model parameters not contributing to the performance of the current task while keeping them available for future tasks. Wortsman et al. [5] proposed a flexible continual learning algorithm called SupSup, designed for scenarios with a large number of tasks. SupSup trains a separate subnetwork for each task and utilizes a linear superposition of the subnetworks to identify the task, performing inference with the selected subnetwork. Kang et al. [6] proposed a method called Winning SubNetworks (WSN), which was inspired by the Lottery Ticket Hypothesis. WSN sequentially learns and selects optimal subnetworks for each task by jointly learning model weights and task-adaptive binary masks.

C. REGULARIZATION-BASED CLs

Elastic Weight Consolidation (EWC) leverages Fisher information matrix to restrict changes to the model parameters, allowing the model to find a good solution for both previous and current tasks [15]. Zenke et al. [16] proposed a simple structural regularize using synaptic intelligence (SI), which measures the importance of each model parameter and penalizes the change to important parameters to prevent them from being overwritten. Li and Hoiem [17] proposed learning without forgetting (LwF), which does not use data from past tasks. LwF adds new task-specific parameters and trains it such that the output values of the past model for the new data do not change significantly.

D. MULTIPLE SINGLE-CLASS CLASSIFIERS

Ye and Zhu [18] proposed an incremental learning algorithm that uses both support vector data description (SVDD) and convolutional neural networks (CNN) as one-class classifiers. They retained the hyperspheres of SVDD for the classes from past tasks and generated a new hypersphere for each new class to avoid CF. Wiewel et al. [19] proposed to transform the multi-class classification problem into multiple one-class classification problems in CL scenarios. Hu et al. [20] proposed a single-class loss function along with a regularization method to solve one-class classification problems and applied the one-class classification method to CL problems. The proposed CLiCK differs from multiple single-class classification methods by not creating a separate neural model for each class. Instead, it utilizes intermediate models generated for previous tasks, as they have already acquired sufficient knowledge for those specific tasks.

III. PROPOSED METHOD

A. CL WITH A SLACK CLASS

We propose CLiCK which performs the classification in CL using the intermediate network models trained for the past datasets and a current model trained with a new dataset. If the characteristics of the dataset change over time in a general CL scenario, the inference error of CLiCK may increase. It implies that the current task is over and a new task begins. Therefore, when CLiCK experiences an increase in the error rate exceeding a threshold, it triggers **a task of model training** with a set of training samples at the moment. It is worth noting that the tasks of model training are not predetermined with a set of distinct classes in a general CL scenario. However, the model training tasks will be referred to as tasks for the sake of simplicity in this paper.

For the CL framework, we define a sequence of tasks denoted by $\mathcal{T} = \{t_1, \cdots, t_N\}$, where N is the total number of tasks. A task $t_k \in \mathcal{T}$ comprises a set of classes $\mathcal{C}_k \subseteq \mathcal{C} = \{1, \cdots, c\}$ and a set of training data samples (x_k, y_k)'s drawn from data distribution of D_k, where x_k is the input data and $y_k \in \mathcal{C}_k$. Suppose t_k is the current task, which is the last task available in $\mathcal{T}$. Then, the task t_k aims to optimize a neural model that minimizes an objective function on the mixture

distribution of entire classes belonging to the past and current tasks.

$$\min_{\theta_k} \mathbb{E}_{(x,y)\sim f(x,y;D_1,\cdots,D_k)}[l(\Theta(x;\theta_k),y)], \tag{1}$$

where $f(x, y; D_1, \cdots, D_k)$ is a mixture distribution, Θ is the classifier for the input x with the parameter of θ, and l is a loss function. The above optimization corresponds to the conventional joint training that requires the entire data distributions of $(D_1, \cdots, D_k)$ up to the current task.

Unlike the joint training in (1), the CL trains the model only using the distribution of the current task. Given θ_{k-1} and D_k, the optimization is given by

$$\min_{\tilde{\theta}_k} \mathbb{E}_{(x,y)\sim D_k}[l(\Theta(x;\tilde{\theta}_k),y)], \tag{2}$$

where θ_{k-1} is the network model of task t_{k-1} and used as the initial weight of $\tilde{\theta}_k$ in the training process. As the training is performed using D_k, the classification performance of finetuning learning in (2) degrades compared with (1).

Here, we introduce a novel concept of a **slack class**, which is a representative class that refers to unseen/undetermined classes that do not belong to the current task. The data samples with the slack class are sampled from the episodic memory $\mathcal{M}_b$ and are used for the training of a model along with its own data samples. The training with a slack class enables each model to be exploited for the classification of future tasks in the CL environment. A model trained with a slack class can identify the samples unseen during its training process and avoid inferring that such samples fall into one of the classes that it was trained for. The classes belonging to the future tasks and unseen during the training of a model will be classified as the slack class of the model.

For example, let us consider a simple class-incremental CL scenario, where $\mathcal{C}_1 = \{0, 1\}$, $\mathcal{C}_2 = \{2, 3\}$, and $\mathcal{C} = \{0, 1, 2, 3\}$. Let s_k denote the slack class for the task t_k. If an input of class 2 is classified using $\tilde{\theta}_1$, the result cannot be avoided from being either 0 or 1. This is why the past task models cannot be exploited for the current task with another set of new classes in CL even though each one can achieve high classification accuracy for its own classes. In contrast, suppose that a model $\hat{\theta}_k$ is trained with s_k. For an input x of class $c \notin \mathcal{C}_k$, the model $\hat{\theta}_k$ can classify the input as s_k rather than one of the classes in $\mathcal{C}_k$. In the example, $\hat{\theta}_1$ and $\hat{\theta}_2$ are trained with $\mathcal{C}_1 \cup \{s_1\}$ and $\mathcal{C}_2 \cup \{s_2\}$, respectively. When an input of class 2 is classified using $\hat{\theta}_1$, the result would be s_1 rather than either 0 or 1. The model $\hat{\theta}_1$ can identify that the input does not belong to the classes of task t_1. If a model classifies an input as a slack class, the inference result of the model is neglected because the model has not seen such an input during its training process.

Namely, for an input x_k of class $c \in \mathcal{C}_k$, the neural model $\tilde{\theta}_k$ for task t_k is supposed to give a right classification decision with a high probability, i.e., $\Theta(x_k; \tilde{\theta}_k) = c$ with high probability if the training is performed properly. However, the preceding models of $\tilde{\theta}_1, \cdots, \tilde{\theta}_{k-1}$ give a wrong classification for the input x_k belonging to t_k because the classes $c \in \mathcal{C}_k$ were not seen when they were trained in the CL. When an input x_k with class $c \notin \mathcal{C}_{k-1}$ and $c \in \mathcal{C}_k$ is classified using the past intermediate model $\tilde{\theta}_{k-1}$, the model $\tilde{\theta}_{k-1}$ would classify the input x_k into one of the classes in $\mathcal{C}_{k-1}$ because of the equality constraint, i.e., $\sum_{c\in\mathcal{C}_i} \mathrm{P}(\Theta(x_k;\tilde{\theta}_i) = c) = 1$ for $i \in \{1, \cdots, k-1\}$. In contrast, the introduction of the slack class converts the equality condition into an inequality condition as follows:

$$\sum_{c\in\mathcal{C}_i} \mathrm{P}(\Theta(x_k;\hat{\theta}_i) = c) \le 1 \quad \text{for } i \in \{1, \cdots, k-1\}. \tag{3}$$

Note that $\sum_{c\in\mathcal{C}_i} \mathrm{P}(\Theta(x_k;\hat{\theta}_i) = c) + \mathrm{P}(\Theta(x_k;\hat{\theta}_i) = s_k) = 1$ for $i \in \{1, \cdots, k-1\}$. The inequality above implies that a model trained with the slack class has a degree of flexibility, allowing an input sample not to be mapped to any of the classes the model is trained on.

B. MODEL TRAINING OF CLiCK

For each task, the model is trained with both its own dataset and the data samples of the slack class. The data samples of the slack class are generated using episodic memory $\mathcal{M}_b$, which keeps the samples of past tasks evenly. Figure 1 illustrates the overall architecture of CLiCK. Here, k is the index of tasks for model training. For the first task t_1 with $k = 1$, there is only one task, and it corresponds to a single-task classification. Thus, the classification decision is made using the only model $\hat{\theta}_1$ generated by the dataset belonging to t_1. For $k = 2$, $\hat{\theta}_2$ is trained with both its own dataset and those of a slack class sampled from 'Training Data 1' in Figure 1. After completing the training of $\hat{\theta}_2$, the episodic memory $\mathcal{M}_b$ is filled with the samples of t_1 and t_2 evenly. From $k = 3$, the model is trained with both its own dataset and the samples in $\mathcal{M}_b$. After the training of t_k, $1/k$ of the samples in $\mathcal{M}_b$ are replaced with those belonging to t_k, and then $\mathcal{M}_b$ is randomly shuffled. Note that the model $\hat{\theta}_1$ is not trained with a slack class in Fig. 1. It may cause inaccuracy of inference when its inference result is combined with the others in the future. Under the assumption that the dataset for $k = 1$ is readily available at $k = 2$, the model $\hat{\theta}_1$ for the first task is trained again with the slack class data sampled from the second task's dataset when the second task t_2 is created. This exception is required only for $\hat{\theta}_1$, and the results reported in this paper are obtained with the retraining of $\hat{\theta}_1$ at $k = 2$. Unlike other episodic memory-based approaches, CLiCK does not use episodic memory to infer each class of the samples in episodic memory. Instead, it uses the memory to generate the samples of the single slack class to differentiate between the current task and unseen/undetermined tasks.

C. CLASSIFICATION INFERENCE OF CLiCK

The classification decision is made by integrating the inference results of all the models including the current model, i.e., $\hat{\theta}_1, \cdots, \hat{\theta}_{k-1}$ and $\hat{\theta}_k$. As the knowledge about past tasks is already retained in the models of the preceding tasks, CLiCK does not need to train the model using the datasets belonging to the past tasks. Instead, it exploits the existing intermediate models of the preceding tasks. As shown in Fig. 1, for an input x, the inference results of $\Theta(x; \hat{\theta}_i)$ are computed for

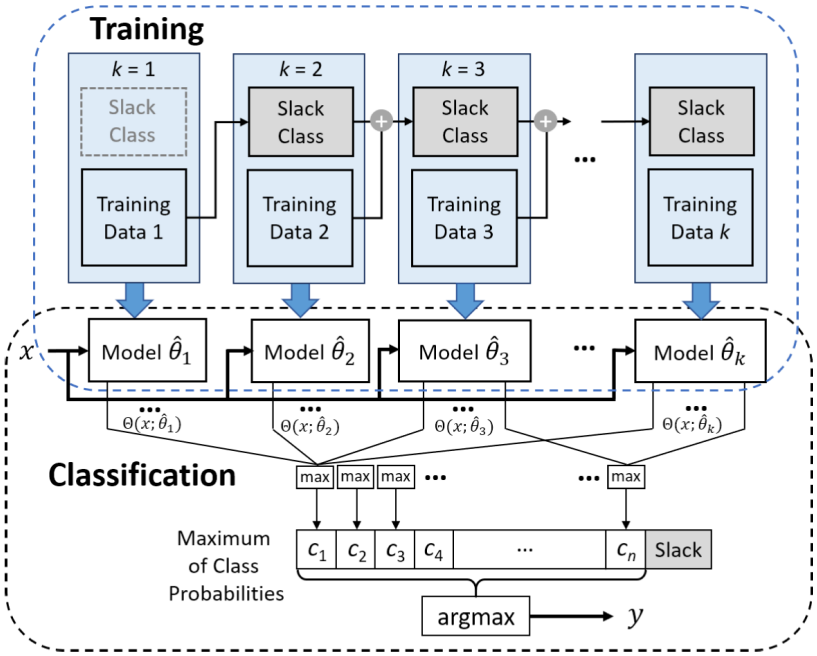


FIGURE 1. Architecture of CLiCK. Training datasets are merged with slack class samples to train the corresponding models. For each class, the maximum of the class probabilities from the models is taken to choose the class with the largest maximum as the inference result. If training data 1 is available at $k = 2$, the first model $\hat{\theta}_1$ can be retrained using the slack class samples drawn from training data 2.

$\forall i \in \{1, \cdots, k\}$. Using the inference results, the maximum value of class probabilities for each class is obtained. Here, the probabilities for the slack class are not involved as illustrated in Fig. 1. If the probability of a model has a large probability for the slack class, it implies that the model does not have meaningful knowledge about the input. In this case, the model would give small inference values for non-slack classes. Therefore, the input x is not likely to be classified as one of the classes in the task belonging to the model. By taking the maximum probability value for each class, the inference result of the models that have not seen the input can be simply discarded. The final classification decision is made by choosing the one that has the largest probability among the classes. Formally, given the models of $\hat{\theta}_1, \cdots, \hat{\theta}_k$, the classification decision for an input x is given by

$$y = \text{argmax}_{c \in \mathcal{C}} \left(\max_{i \in \mathcal{T}_c := \{t \in \mathcal{T} | c \in \mathcal{C}_t\}} \text{P}(\Theta(x; \hat{\theta}_i) = c) \right), \quad (4)$$

where $\mathcal{T}_c$ is a set of tasks that include class c during the training. As an example, let us consider a general CL where $C_1 \cap C_2 \neq \emptyset$, e.g., $C_1 = \{0, 1\}$ and $C_2 = \{1, 2\}$. If the input x is of class 1, $\text{P}(\Theta(x; \hat{\theta}_1) = 1)$ and $\text{P}(\Theta(x; \hat{\theta}_2) = 1)$ for $c = 1$ are larger than $\text{P}(\Theta(x; \hat{\theta}_1) = 0)$ for $c = 0$ and $\text{P}(\Theta(x; \hat{\theta}_2) = 2)$ for $c = 2$. As a result, it is classified as 1. If the distributions of $c = 1$ are different at t_1 and t_2, an input sample x with $c = 1$ would have either $\text{P}(\Theta(x; \hat{\theta}_1) = 1) \approx 1$ or $\text{P}(\Theta(x; \hat{\theta}_2) = 1) \approx 1$ depending on the task that the input sample belongs to. In this case, the classification of (4) still gives the right decision. Note that the task to which the input sample belongs can be identified in (4).

D. OVERHEAD AND LIMITATION OF CLiCK

1) COMPUTATIONAL COMPLEXITY

Each model of CLiCK requires to be trained for an extra slack class, and the output size of the model is increased by one. The final classification is made by combining the inference results of all the models, resulting in an increase in inference time with respect to the number of models. These architecture-based CLs' limitations can be mitigated by reducing the size of each model because a small-sized model that focuses on its own task rather than the entire classification tasks can be used.

2) STORAGE CAPACITY

CLiCK needs to store past models. However, each model does not need to be too large to consider the potential expansion of tasks in the future. As new tasks arrive, the corresponding models are added gradually. As like other rehearsal-based CLs, episodic memory space is needed. While rehearsal-based CLs store past datasets to train the model for each class in the past datasets, CLiCK maps the entire dataset in the episodic model into the slack class. The performance of CLiCK is less sensitive to the size of memory and thus requires relatively small memory in comparison with other rehearsal-based CLs. When the second model is trained, CLiCK retrains the first model using the dataset belonging to the first model and the

slack dataset sampled from the second dataset. For doing this, the first dataset needs to be stored. To save the storage, the first dataset is stored in the episodic memory instead of using a separate memory space.

IV. EXPERIMENTS

We evaluate the performance of the proposed CLiCK method under various configurations against classic and the latest state-of-the-art CL baselines. Domain-incremental and class-incremental continual learning scenarios are considered. All experiments were conducted on a workstation equipped with an AMD Ryzen 9 5900X CPU and two NVIDIA GeForce RTX 4090 GPUs.

Dataset and augmentation: We use four popular datasets, which have been widely used in recent CL methods [7], [10], [11], [21], [22]. The datasets are as follows:

- **MNIST** [23] contains 60,000 training images and 10,000 test images of handwritten digits, each of which is 28 × 28 pixels in size and grayscale. Each image is labeled with the correct digit, ranging from 0 to 9, and there are 10 classes in total.
- **Permuted MNIST** [16] is a variation of the original MNIST dataset, but the pixels of each image have been randomly shuffled.
- **Rotated MNIST** [24] is another variation of the MNIST. The images of the original MNIST have been rotated at random angles.
- **CIFAR-100** [25] comprises 60,000 images of 32 × 32 size, assigned with one of 100 object classes. We applied data augmentation techniques, such as random cropping, random horizontal flipping, and color jittering, to CIFAR-100 training image data.

Target model and comparing schemes: We employ the three most commonly used models: VGG-11 [26], GoogLeNet [27], and ResNet-18 [28]. The intention of this experimental design is to show that CLiCK is not dependent on any specific type of network model. ResNet-18 is composed of 4 basic blocks following the first convolution layer. CLiCK is trained with the stochastic gradient descent (SGD) optimizer with a learning rate of 0.1, momentum of 0.9, and weight decay of 5e-4. The results presented in this section are based on the average of five separate trials. We compare the proposed method with **joint learning**, **iCaRL [7]**, **A-GEM [10]**, **DER++ [11]**, **GDumb [14]**, and **finetuning**. The default values for epochs, batch size, and buffer size were 200, 128, and 2,000.

A. RESULTS IN CLASS INCREMENTAL SETTINGS WITH MNIST DATASET

Class incremental learning involves training a model on a sequence of tasks, each with a different set of classes but from the same feature space or domain. The goal is to create a model that performs well on all tasks seen so far without forgetting what it has learned on previous tasks. In all of our experiments using the MNIST dataset, we utilized a subset of 500 randomly selected images from a total of 6,000 images per class for model training. The leftmost results in Table 1 show the accuracy of ResNet-18 for the MNIST dataset in a class-incremental CL scenario, where the total number of tasks is five, and each task includes two classes. The results demonstrate the performance of the algorithms in terms of the average accuracy over the entire task and the accuracy of the last task. It is observed that the accuracy performance degradation of CLiCK compared to joint learning was about 5.56% when using a rehearsal memory size of 100, and 3.88% when using a memory size of 500 in terms of the average accuracy. It seems that a memory size of 500 is sufficient for all rehearsal-based strategies to retain the knowledge of past tasks in the class-incremental scenario on the MNIST dataset. Our method outperforms the compared several baselines, indicating the effectiveness of our method in preventing CF.

B. RESULTS IN DOMAIN INCREMENTAL SETTINGS WITH MNIST DATASET

Domain incremental learning involves training a model on a sequence of tasks, each with the same set of classes but from different feature spaces or domains. The goal is to create a model that performs well on all tasks seen so far without forgetting what it has learned on previous tasks, and without being affected by changes in the feature space or distribution. Our proposed method was evaluated on both permuted MNIST and rotated MNIST datasets. In our experiments, we evaluated the performance of our proposed method on two datasets: permuted MNIST and rotated MNIST. The permuted MNIST dataset was created by randomly rearranging the pixels in each image, while the rotated MNIST dataset was created by randomly rotating each image by an angle between 0 and 360 degrees. These datasets were used to simulate changes in the feature space or distribution, which are common in real-world scenarios. Our method achieved comparable performance to joint learning in both datasets in Table 1 and Fig. 2, indicating that the intermediate models are able to classify their own classes accurately and that the classes not belonging to the task are classified as slack classes. We observed that DER++ outperformed A-GEM, GDumb, and finetuning. GDumb was particularly sensitive to memory size, as it determines the total number of samples used for training. Overall, the results demonstrate that the proposed method is effective in both class incremental learning and domain incremental learning scenarios where data distribution and classes can change over time. These two CL settings are crucial in the field of continual learning as they reflect real-world scenarios where data distribution and classes can change over time. The experiment results indicate that our proposed method is effective in both class incremental learning and domain incremental learning scenarios.

C. RESULTS IN CLASS INCREMENTAL SETTINGS WITH CIFAR DATASET

For the CIFAR-100, we performed a set of experiments in six CL scenarios. i) The first three scenarios were created by

TABLE 1. Accuracy result of class/domain incremental CL scenario of MNIST dataset.

Scenario		MNIST		Permuted MNIST		Rotated MNIST	
Method	Memory	Average	Last	Average	Last	Average	Last
Joint	-	98.09±0.17	99.19±0.11	95.97±0.21	96.07±0.11	98.71±0.05	98.76±0.09
CLiCK (proposed)	100	**92.53**±0.19	**83.44**±0.39	**95.98**±0.18	**96.15**±0.21	**97.86**±0.21	**96.62**±0.38
	500	**94.21**±0.31	**85.67**±0.29	**96.02**±0.21	**96.38**±0.19	**98.26**±0.19	**97.83**±0.27
DER++	100	86.92±0.21	76.14±0.63	87.49±0.73	81.54±0.77	88.08±0.48	82.19±0.96
	500	90.34±0.44	83.43±0.63	91.37±0.25	87.51±1.07	91.15±0.57	88.25±1.26
A-GEM	100	74.74±0.43	62.76±0.32	79.85±0.29	64.07±0.45	80.06±0.37	64.70±1.43
	500	78.23±0.49	66.45±0.64	81.96±0.21	68.49±1.02	87.44±0.24	81.39±1.08
GDumb	100	43.31±0.87	30.19±2.37	42.62±1.17	28.64±3.36	46.32±1.03	34.36±0.78
	500	64.21±0.93	57.23±3.22	66.15±0.86	54.74±5.11	75.83±0.63	60.06±2.10
Finetuning	-	52.37±0.76	39.58±1.24	51.08±0.83	29.18±1.41	60.66±0.62	43.19±0.78

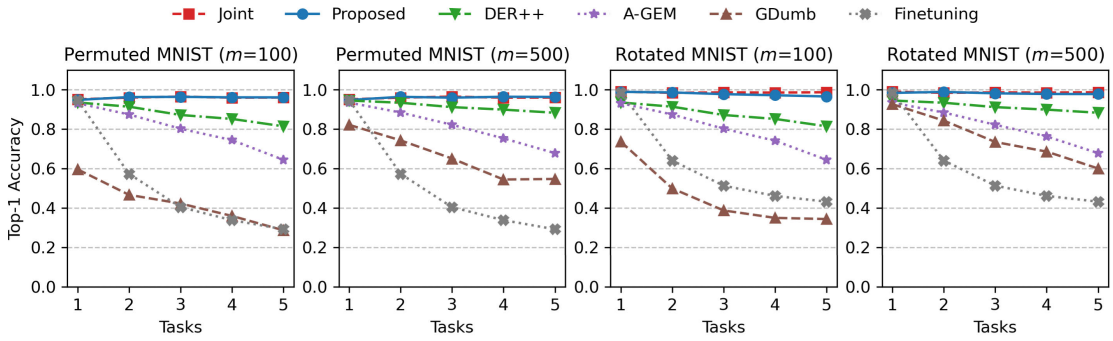


FIGURE 2. Top-1 results of ResNet-18 for domain incremental scenarios with MNIST dataset.

making every task have the same number of classes. The three scenarios in this category have 5, 10, and 20 classes per task. As a result, they consist of 20, 10, and 5 tasks for the CIFAR-100 dataset with 100 classes. ii) In the next three scenarios, the half of classes were trained during Task 1. Then, the remaining 50 classes were evenly divided (i.e., 2, 5, and 10 classes per task) and were incrementally added to the existing tasks.

1) COMPARISON OF PERFORMANCE ACROSS DIFFERENT SCENARIOS

Figure 3 shows the Top-1 accuracy of ResNet-18 for the CIFAR-100 dataset. In Figure 3, the joint learning shows the highest accuracy performance in the six scenarios and serves as a maximum performance bound of the accuracy. Even in joint learning, the accuracy at the last task is lower than that at the first task, and it is perhaps due to the increase in the problem complexity with more possible choices. The other algorithms significantly suffer from CF. If the number of classes belonging to the current task is relatively small compared with the number of previous classes, the accuracy performance is not good due to the CF problem. This is why the cases of 20 classes per task give higher accuracy performance in Fig. 3.

In the first-category scenarios, the accuracy performance degradation of CLiCK compared to joint learning was approximately 24.95% for 5 classes per task and 7.14% for 20 classes per task in Fig. 3. The accuracy in the second-category scenarios is less than in the first-category scenarios for each number of classes. Under CLiCK, the images belonging to the slack class were sampled from more classes on average in the second-category scenarios, resulting in the best accuracy performance except the joint learning. DER++ outperformed A-GEM, iCaRL, GDumb, and finetuning, and achieved slightly better performance than CLiCK in the third and sixth scenarios of Fig. 3.

2) EFFECTS OF DIFFERENT MEMORY SIZE

Figure 4 shows the Top-1 accuracy of the rehearsal memory-based algorithms with respect to the memory sizes. In this experiment set, each task consists of 10 image classes, and each image class includes 500 images. We have changed the memory size m to 1,000, 2,000, and 5,000. It is observed that the larger memory achieves a more accurate performance because more samples belonging to the previous tasks can be retained in the memory. For example, if $m = 1{,}000$, the number of image samples per class in the rehearsal memory (i.e., images belonging to the previous tasks) becomes 100, 50, $\cdots$, 11 in Task 2, 3, $\cdots$, 10, respectively, which correspond to only 20%, 10%, $\cdots$, 2.2% of the joint learning case. Note that the number of image samples per class in joint learning is 500. In Fig. 4, CLiCK does not show a significant change in the accuracy performance compared to the other algorithms when m changes. The reason is that the samples in the memory are not used for classifying individual classes in CLiCK and they are classified as the single slack class, unlike the other algorithms. The experiment results of the average incremental accuracy (over all the tasks) and the last task's accuracy were reported in Table 2.

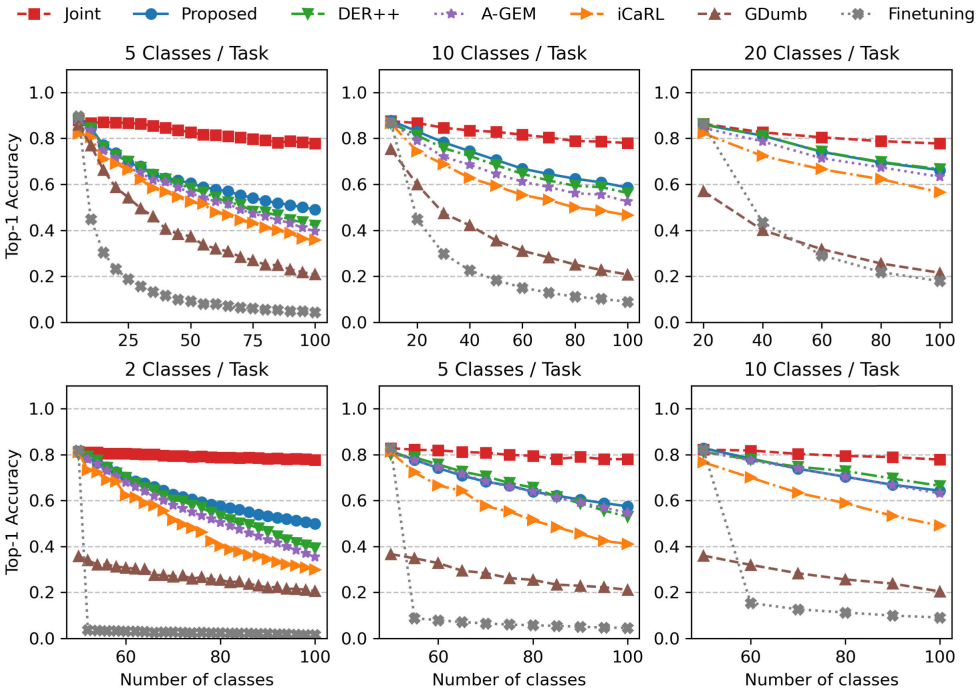


FIGURE 3. Top-1 results of ResNet-18 for the CIFAR-100 dataset in six different CL scenarios.

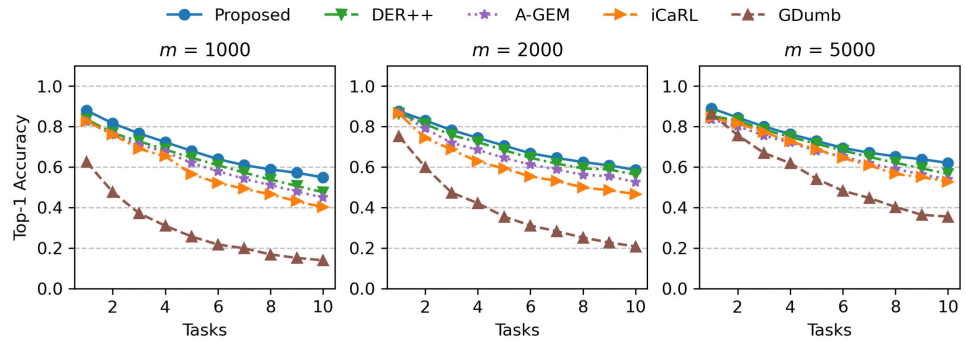


FIGURE 4. Top-1 accuracy result for the CIFAR-100 dataset with respect to the different memory sizes.

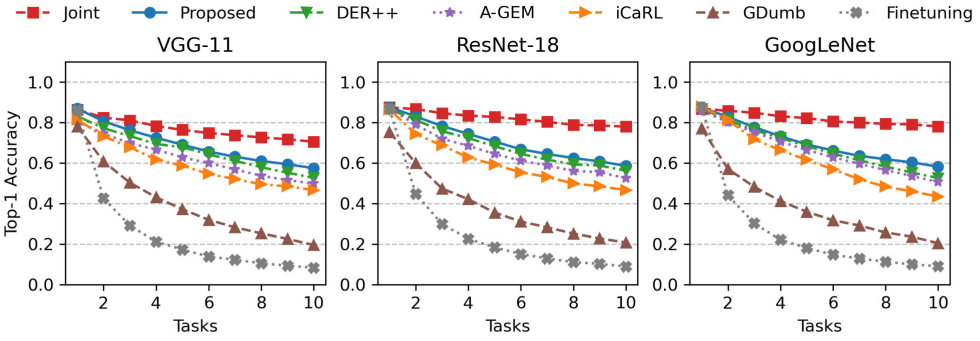


FIGURE 5. Top-1 accuracy results of VGG-11, ResNet-18, and GoogLeNet for the CIFAR-100 dataset.

3) COMPARISON OF THE IMPACT OF DIFFERENT MODELS

Figure 5 shows the Top-1 accuracy performance of VGG-11, ResNet-18, and GoogLeNet. For the three different learning models, CLiCK shows the best performance that is the closest to that of joint learning. Table 3 and 4 show the Top-1 accuracy results of VGG-11 and GoogLeNet, respectively, for the class-incremental CL scenarios with the CIFAR-100 dataset. In the tables, CLiCK achieved the best performance except for a few cases. Even in the cases where DER++ outperforms CLiCK, the performance discrepancy is quite small.

TABLE 2. Top-1 accuracy result of ResNet-18 for CIFAR-100 dataset with image augmentation.

Classes / Task		5 Classes		10 Classes		20 Classes	
Method	Memory	Average	Last	Average	Last	Average	Last
Joint	-	83.81±0.85	78.29±0.34	83.45±1.23	78.82±0.32	82.44±0.68	78.59±0.46
CLiCK (proposed)	1,000	**70.08**±0.89	**52.23**±1.74	**71.35**±1.46	**57.32**±1.23	**75.38**±0.74	**65.21**±0.47
	2,000	**71.15**±0.54	**59.01**±0.68	**70.72**±0.74	**58.57**±0.62	**76.45**±0.67	**68.11**±0.35
	5,000	**72.56**±0.47	**58.26**±0.29	**74.22**±1.12	**63.71**±0.92	77.22±0.84	69.26±0.46
DER++	1,000	65.80±1.38	57.48±2.15	63.31±0.72	45.49±1.40	73.07±1.57	63.55±1.01
	2,000	68.27±1.28	60.93±1.12	68.59±0.91	57.45±1.95	76.12±0.87	66.34±1.13
	5,000	72.41±1.38	67.03±0.81	70.12±0.43	58.31±1.50	**78.82**±1.32	**69.96**±1.21
A-GEM	1,000	62.28±0.49	52.71±0.89	60.78±1.24	45.34±1.33	71.12±1.21	64.44±1.89
	2,000	57.35±0.82	47.73±0.65	66.35±1.13	52.25±0.84	71.76±1.11	63.23±1.64
	5,000	67.38±1.16	61.76±1.05	68.22±0.71	54.29±1.69	73.04±1.32	64.66±1.76
iCaRL	1,000	63.68±0.38	57.21±0.57	57.79±0.69	42.29±1.65	67.45±1.25	57.45±1.81
	2,000	64.48±0.39	59.21±0.27	60.41±0.88	49.26±1.22	70.33±0.76	59.68±1.28
	5,000	67.06±0.42	61.57±1.26	67.42±0.77	55.23±0.92	72.12±1.03	61.45±1.61
GDumb	1,000	22.39±0.67	15.71±0.36	29.14±1.79	13.91±0.48	24.76±0.99	13.88±0.79
	2,000	31.72±0.55	22.10±0.40	38.81±1.37	20.76±1.15	35.16±1.65	21.49±0.25
	5,000	48.84±0.85	34.64±0.95	55.02±1.70	35.49±1.42	51.09±1.07	34.52±1.04
Finetuning	-	17.67±0.21	7.37±0.43	26.02±0.24	8.87±0.37	39.68±0.34	17.85±0.23

TABLE 3. Top-1 accuracy result of VGG-11 for CIFAR-100 dataset with image augmentation.

Classes / Task		5 Classes		10 Classes		20 Classes	
Method	Memory	Average	Last	Average	Last	Average	Last
Joint	-	77.10±1.57	70.36±0.12	76.84±0.59	70.51±0.34	75.93±1.92	70.46±0.30
CLiCK (proposed)	1,000	**64.85**±1.12	**51.76**±0.74	**69.66**±0.68	**57.13**±1.14	**71.94**±1.43	**60.33**±0.39
	2,000	**67.19**±1.32	**54.65**±1.12	**72.45**±0.68	**59.37**±1.30	**72.27**±1.33	**63.15**±0.67
	5,000	**70.15**±1.45	57.28±1.15	**73.17**±0.89	**61.22**±0.92	**74.16**±1.22	65.73±1.37
DER++	1,000	62.43±0.67	48.69±1.45	67.56±0.67	53.91±1.65	69.89±1.42	59.34±0.72
	2,000	64.65±1.18	53.45±2.19	68.92±3.16	54.43±2.11	70.72±0.83	61.97±1.71
	5,000	68.07±0.67	**57.71**±1.55	71.69±0.21	59.03±1.16	73.97±0.55	**66.12**±1.45
A-GEM	1,000	61.14±0.56	49.32±1.51	64.97±1.87	54.61±1.23	67.19±0.96	59.40±1.37
	2,000	62.39±1.18	51.91±1.89	66.81±1.32	55.85±1.55	68.02±1.33	60.33±1.97
	5,000	64.52±1.34	52.78±1.60	68.96±1.54	58.22±1.75	70.25±1.63	61.78±2.11
iCaRL	1,000	59.56±0.94	47.06±1.35	62.77±1.43	52.05±1.88	63.28±0.92	55.37±2.78
	2,000	62.13±1.56	50.43±2.03	65.63±1.26	46.13±2.18	64.95±0.73	54.73±1.83
	5,000	64.41±1.43	51.83±1.64	66.95±0.97	53.24±1.61	67.17±0.92	59.14±1.98
GDumb	1,000	31.45±1.67	13.76±1.76	28.21±1.88	14.96±1.33	23.89±1.43	13.18±1.67
	2,000	41.39±1.54	20.14±0.72	39.68±0.62	19.47±0.94	37.45±1.67	21.26±1.17
	5,000	54.10±1.50	32.96±0.67	52.78±0.91	32.88±1.09	49.57±0.96	32.64±1.23
Finetuning	-	15.34±0.73	4.22±0.44	25.01±0.25	8.29±0.25	39.22±0.97	18.13±0.23

TABLE 4. Top-1 accuracy result of GoogleNet for CIFAR-100 dataset with image augmentation.

Classes / Task		5 Classes		10 Classes		20 Classes	
Method	Memory	Average	Last	Average	Last	Average	Last
Joint	-	83.01±0.85	79.22±0.54	83.12±0.93	78.81±0.56	82.12±1.12	79.25±1.33
CLiCK (proposed)	1,000	**69.12**±1.34	**49.56**±0.93	**70.23**±1.45	**56.23**±0.78	**74.42**±1.22	**63.34**±0.64
	2,000	**70.45**±1.59	**56.21**±1.36	**69.86**±0.89	**57.77**±0.66	**74.50**±1.33	**64.89**±1.78
	5,000	**73.41**±1.57	**57.12**±1.35	**73.69**±1.28	**61.72**±0.62	76.16±0.74	67.96±1.44
DER++	1,000	67.38±1.27	47.49±1.48	65.42±1.54	51.68±1.31	74.29±0.64	64.44±0.98
	2,000	69.44±1.65	55.84±2.04	66.33±0.77	54.12±1.67	72.59±1.32	62.67±1.61
	5,000	72.59±1.44	56.12±1.65	71.12±1.34	59.86±1.30	**76.79**±1.56	**68.43**±1.44
A-GEM	1,000	63.39±1.13	41.46±1.16	61.68±1.44	38.82±1.29	68.78±0.52	57.54±1.11
	2,000	64.13±1.61	47.16±1.08	65.58±0.86	48.96±1.21	67.10±0.32	56.84±0.99
	5,000	69.26±1.87	55.09±0.90	69.94±0.94	59.41±0.80	70.28±1.27	61.71±2.71
iCaRL	1,000	52.23±1.34	34.28±1.49	59.98±1.34	51.12±1.76	59.45±1.55	49.89±1.19
	2,000	63.45±1.32	42.47±2.48	63.83±1.65	43.34±1.84	63.97±1.34	53.84±1.34
	5,000	68.98±1.13	49.40±1.15	67.34±1.33	58.29±1.34	65.77±1.69	57.28±1.65
GDumb	1,000	31.42±1.27	18.12±0.58	27.98±1.43	14.23±1.54	24.25±1.12	15.45±0.84
	2,000	40.65±1.04	20.53±1.27	39.00±1.21	20.39±0.66	35.01±1.86	20.43±0.79
	5,000	59.20±0.65	39.25±1.80	57.20±1.06	37.52±2.35	53.55±0.94	38.42±0.74
Finetuning	-	18.12±0.19	4.67±0.56	28.47±0.45	10.12±0.49	40.56±1.45	18.53±0.66

V. CONCLUSION

We have proposed a new approach, called CLiCK, for addressing the problem of catastrophic forgetting in continual learning. CLiCK trains models for new datasets with a slack class, and the models that do not have knowledge about input are automatically neglected in the final decision. We have conducted a thorough evaluation of various scenarios. The results indicate that CLiCK outperforms existing methods and has great potential for further development and practical use.

REFERENCES

[1] R. French, "Catastrophic forgetting in connectionist networks," *Trends Cognit. Sci.*, vol. 3, no. 4, pp. 128–135, Apr. 1999.

[2] G. I. Parisi, R. Kemker, J. L. Part, C. Kanan, and S. Wermter, "Continual lifelong learning with neural networks: A review," *Neural Netw.*, vol. 113, pp. 54–71, May 2019.

[3] A. A. Rusu, N. C. Rabinowitz, G. Desjardins, H. Soyer, J. Kirkpatrick, K. Kavukcuoglu, R. Pascanu, and R. Hadsell, "Progressive neural networks," 2016, *arXiv:1606.04671*.

[4] 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.

[5] M. Wortsman, V. Ramanujan, R. Liu, A. Kembhavi, M. Rastegari, J. Yosinski, and A. Farhadi, "Supermasks in superposition," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 33, 2020, pp. 15173–15184.

[6] H. Kang, R. J. L. Mina, S. R. H. Madjid, J. Yoon, M. Hasegawa-Johnson, S. J. Hwang, and C. D. Yoo, "Forget-free continual learning with winning subnetworks," in *Proc. Int. Conf. Mach. Learn.*, 2022, pp. 10734–10750.

[7] 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. 2001–2010.

[8] H. Shin, J. K. Lee, J. Kim, and J. Kim, "Continual learning with deep generative replay," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 30, 2017, pp. 2994–3003.

[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," in *Proc. Int. Conf. Learn. Represent.*, 2019, pp. 1–20.

[11] P. Buzzega, M. Boschini, A. Porrello, D. Abati, and S. Calderara, "Dark experience for general continual learning: A strong, simple baseline," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 33, 2020, pp. 15920–15930.

[12] L. Caccia, R. Aljundi, N. Asadi, T. Tuytelaars, J. Pineau, and E. Belilovsky, "New insights on reducing abrupt representation change in online continual learning," in *Proc. Int. Conf. Learn. Represent.*, 2022, pp. 1–27.

[13] J. Bang, H. Kim, Y. Yoo, J.-W. Ha, and J. Choi, "Rainbow memory: Continual learning with a memory of diverse samples," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2021, pp. 8218–8227.

[14] A. Prabhu, P. H. Torr, and P. K. Dokania, "GDumb: A simple approach that questions our progress in continual learning," in *Proc. Eur. Conf. Comput. Vis.* Cham, Switzerland: Springer, 2020, pp. 524–540.

[15] J. Kirkpatrick, R. Pascanu, N. Rabinowitza, J. Venessa, G. Desjardinsa, A. A. Rusu, K. Milan, J. Quan, T. Ramalhond, A. Grabska-Barwinska, D. Hassabis, C. Clopath, D. Kumaran, and R. Hadsell, "Overcoming catastrophic forgetting in neural networks," *Proc. Nat. Acad. Sci. USA*, vol. 114, pp. 3521–3526, Mar. 2017.

[16] F. Zenke, B. Poole, and S. Ganguli, "Continual learning through synaptic intelligence," in *Proc. Int. Conf. Mach. Learn.*, 2017, pp. 3987–3995.

[17] Z. Li and D. Hoiem, "Learning without forgetting," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 12, pp. 2935–2947, Dec. 2017.

[18] X. Ye and Q. Zhu, "Class-incremental learning based on feature extraction of CNN with optimized softmax and one-class classifiers," *IEEE Access*, vol. 7, pp. 42024–42031, 2019.

[19] F. Wiewel, A. Brendle, and B. Yang, "Continual learning through one-class classification using VAE," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, May 2020, pp. 3307–3311.

[20] W. Hu, M. Wang, Q. Qin, J. Ma, and B. Liu, "HRN: A holistic approach to one class learning," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 33, 2020, pp. 19111–19124.

[21] M. M. Derakhshani, X. Zhen, L. Shao, and C. Snoek, "Kernel continual learning," in *Proc. Int. Conf. Mach. Learn.*, 2021, pp. 2621–2631.

[22] M. Riemer, I. Cases, R. Ajemian, M. Liu, I. Rish, Y. Tu, and G. Tesauro, "Learning to learn without forgetting by maximizing transfer and minimizing interference," in *Proc. Int. Conf. Learn. Represent.*, 2019, pp. 1–13.

[23] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.

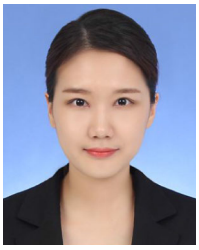
[24] S. I. Mirzadeh, M. Farajtabar, R. Pascanu, and H. Ghasemzadeh, "Understanding the role of training regimes in continual learning," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 33, 2020, pp. 7308–7320.

[25] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," Dept. Comput. Sci., University of Toronto, Toronto, ON, Canada, Tech. Rep., 2009.

[26] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *Proc. Int. Conf. Learn. Represent.*, 2015, pp. 1–14.

[27] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 1–9.

[28] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770–778.



HYEJIN KIM received the B.S. degree in electronic engineering from Dong-A University, Busan, Republic of Korea, in 2017. She is currently pursuing the integrated M.S./Ph.D. degrees with the AI Graduate School, Gwangju Institute of Science and Technology (GIST), Gwangju, Republic of Korea. Her research interests include cybersecurity and artificial intelligence.

SEUNGHYUN YOON (Member, IEEE) received the B.S. degree in computer science from Handong Global University, Pohang, Republic of Korea, in 2016, and the Ph.D. degree in electrical engineering and computer science from the Gwangju Institute of Science and Technology (GIST), Gwangju, Republic of Korea, in 2021. He has been an Assistant Professor with the Department of Energy Engineering, Korea Institute of Energy Technology (KENTECH), since August 2022. From 2019 to 2020, he was a Visiting Scholar with the Department of Computer Science, Virginia Tech. From 2021 to 2022, he was a Senior Researcher with the Korea Institute of Industrial Technology (KITECH). His research interests include artificial intelligence (AI), software-defined networking (SDN), network resource allocation and optimization, cybersecurity, and moving target defense (MTD). He is also conducting active research on continual learning and federated/distributed learning in AI.

HYUK LIM (Member, IEEE) received the B.S., M.S., and Ph.D. degrees from the School of Electrical Engineering and Computer Science, Seoul National University, Seoul, Republic of Korea, in 1996, 1998, and 2003, respectively. From 2003 to 2006, he was a Postdoctoral Research Associate with the Department of Computer Science, University of Illinois at Urbana–Champaign, Champaign, IL, USA. From 2006 to 2021, he was a Professor with the AI Graduate School and jointly with the School of Electrical Engineering and Computer Science, Gwangju Institute of Science and Technology (GIST), Gwangju, Republic of Korea. He was the Dean of the School of Electrical Engineering and Computer Science and the Director of the GIST Institute for AI. In 2022, he joined the Korea Institute of Energy Technology (KENTECH), Naju, Republic of Korea, as a Full Professor. His research interests include artificial intelligence, cybersecurity, big data privacy, software-defined networking, and wireless communication systems.