Continual Learning With Unknown Task Boundary

Xiaoxie Zhu, Jinfeng Yi, *Member, IEEE*, and Lijun Zhang, *Senior Member, IEEE*

***Abstract*— Most existing studies on continual learning (CL) consider the task-based setting, where task boundaries are known to learners during training. However, they may be impractical for real-world problems, where new tasks arrive with unnotified distribution shifts. In this article, we introduce a new boundary-unknown continual learning scenario called continuum incremental learning (CoIL), where the incremental unit may be a concatenation of several tasks or a subset of one task. To identify task boundaries, we design a continual out-of-distribution (OOD) detection method based on softmax probabilities, which can detect OOD samples for the latest learned task. Then, we incorporate it with continual learning approaches to solve the CoIL problem. Furthermore, we investigate the more challenging task-reappear setting and propose a method named continual learning with unknown task boundary (CLUTaB). CLUTaB first adopts in-distribution detection and OOD loss to determine whether a set of data is sampled from any learned distribution. Then, a two-step inference technique is designed to improve the continual learning performance. Experiments show that our methods work well with existing continual learning approaches and achieve good performance on CIFAR-100 and mini-ImageNet datasets.**

***Index Terms*— Continual learning (CL), continuum incremental learning (CoIL), deep learning, out-of-distribution (OOD) detection.**

I. Introduction

TRADITIONAL machine learning methods learn from independent and identically distributed data. In practical applications, a learner often encounters new tasks over time. The data distribution shifts with the arrival of new tasks, and we hope that the learner could pick up the new task while maintaining performance on existing ones. However, deep learning models often suffer from catastrophic forgetting (a significant performance degradation on old tasks) without access to training data from previously learned tasks [1], [2].

To address the issue, continual learning (CL) aims to learn a sequence of tasks without forgetting. In recent years, numerous task-based continual learning approaches have been proposed. They consolidate previous knowledge by regularization [3], [4], replay [5], [6], and expansion [7], [8]. These task-based approaches assume that a model will not encounter a new task before finishing learning the current one. However, in a more realistic scenario, we collect training data until a certain point, e.g., out of disk space in a memory-constrained environment, and drop these data after learning. We refer to the segment of data collected in a continuous period of time as a data continuum. Data in the continuum may come from the latest learned task, a new task, or even an old task. In this situation, it is hard to learn tasks one after the other. In addition, given the task boundaries during training, task-based approaches consolidate knowledge when the model is about to learn a new task. Unfortunately, input distribution shifts can occur without notification. For example, a classifier distinguishing between mammals and fish first learns from images of lions and carps, and then, we collect more images of giraffes and sharks but they are still labeled as mammals and fish. Ideally, the classifier can predict class labels (distinguish mammals from fish) and identify distribution shifts (distinguish giraffes from lions and sharks from carps). With distribution shift awareness, we can further understand the collected data and better evaluate the model, like designing more fine-grained labels and evaluating model performance by task.

The observations above motivate us to propose a new continual learning scenario called continuum incremental learning (CoIL). Different from the task-based setting [9], the incremental unit in CoIL is a data continuum instead of a task. We assume that adjacent data in continua are sampled from the same distribution unless they are separated by an unknown task boundary. The main obstacle in CoIL is to identify distribution shifts based on a continual learning model. Inspired by previous out-of-distribution (OOD) detection methods for nonincremental learning [10], [11], we design a continual OOD detection method to distinguish OOD samples, which are sampled from a new distribution, and in-distribution (InD) samples, which are sampled from the latest learned one. Specifically, we adopt an adaptive threshold to accommodate OOD detection for different tasks. When sufficient samples are detected as outliers, we think a task boundary is identified. Then, we incorporate continual OOD detection with continual learning approaches to solve a CoIL problem, where all data from the same task are consecutive. Furthermore, we investigate the more challenging task-reappear setting, where a continuum may consist of extra data from any learned task. To determine whether a set of data is sampled from a learned distribution, we propose continual learning with unknown task boundary (CLUTaB) with continual InD detection, which leverages exemplars to widen the gap between the max probabilities of InD and

Manuscript received 26 May 2022; revised 1 December 2022, 22 May 2023, and 11 May 2024; accepted 29 May 2024. This work was supported in part by NSFC under Grant U23A20382 and in part by the Collaborative Innovation Center of Novel Software Technology and Industrialization. *(Corresponding author: Lijun Zhang.)*

Xiaoxie Zhu is with the National Key Laboratory for Novel Software Technology, Nanjing University, Nanjing 210023, China (e-mail: zhuxx@lamda.nju.edu.cn).

Jinfeng Yi is with 4Paradigm Inc., Beijing 100085, China (e-mail: yijinfeng@4paradigm.com).

Lijun Zhang is with the National Key Laboratory for Novel Software Technology and the School of Artificial Intelligence, Nanjing University, Nanjing 210023, China (e-mail: zhanglj@lamda.nju.edu.cn).

Digital Object Identifier 10.1109/TNNLS.2024.3412934

OOD samples. Experiments show that our methods work well with existing continual learning approaches and achieve good performance on common benchmark datasets.

This article is organized as follows. Section II summarizes recent research related to continual learning and OOD detection. Section III formally describes the CoIL scenario and compares it with other continual learning scenarios. We propose the continual OOD detection method to identify task boundaries in Section IV. In Section V, we design InD detection and CLUTaB to solve a task-reappear CoIL setting. In Section VI, we report and analyze the experimental results, and we conclude this article in Section VII.

II. Related Work

In this section, we review existing literature that relates to our proposed methodology from three perspectives: task-based continual learning, task-free continual learning, and OOD detection.

A. Task-Based Continual Learning

In task-based continual learning, models learn a sequence of tasks one after the other without access to training data from previously learned tasks [12]. Task-based continual learning can be divided into three scenarios [9], [13]: task-incremental learning, domain-incremental learning, and class-incremental learning. In task-incremental learning, task identifiers are always given by an oracle, while they are not available at test time in domain-incremental and class-incremental learning. Models need to infer which task they are dealing with in class-incremental learning and need not in domain-incremental learning. Class-incremental learning attracts more attention since it is more difficult and more practical.

Based on how to consolidate knowledge, approaches for task-based continual learning can be classified into three branches [14]: regularization [3], [4], [15], [16], [17], [18], replay [5], [19], [20], [21], [22], and expansion [7], [8], [23], [24]. Regularization-based approaches add regularization terms in loss function to prevent catastrophic forgetting. Replay-based approaches store a subset of samples from previous tasks for replay when learning new tasks. As a replay-based method, SS-IL [20] adopts separated softmax combined with taskwise knowledge distillation to resolve the severe data imbalance between new tasks and previous tasks. PMR [25] proposes synthetic prototypes as knowledge representations to guide the sample selection for memory replay. Expansion-based approaches add new network components to learn new data. Neural architecture search [26] and network pruning [27] are employed to discern optimal network architectures for novel tasks while ensuring memory efficiency. To avoid catastrophic forgetting, these approaches need to consolidate knowledge after learning a task, like storing exemplars [6], [28] and old model parameters [29], and allocating more memory for new network architecture [26]. Since models learn from all data of one task at a time in task-based continual learning, task boundaries are always located at the end of each incremental unit. However, task-based continual learning approaches cannot be directly applied to CoIL. They are unaware of task boundaries so they do not know when to consolidate knowledge, and also they cannot distinguish which task they are presented with at both training and test times.

B. Task-Free Continual Learning

Task-free continual learning aims at learning from a stream of data without task identifiers and boundaries [12]. Approaches for task-free continual learning can also be classified into three branches: regularization [12], replay [30], [31], and expansion [32], [33]. However, in task-free continual learning, new classes are introduced when distribution shifts occur, and many works exploit label tricks to determine when and how to consolidate knowledge. CoPE [34] sets up a prototype for a class if it is discovered for the first time and adopts a balanced replay for each class according to exemplars' class labels. CN-DPM [33] utilizes class labels to calculate weights for experts during training. To edit memory examples, GMED [35] needs to update model parameters with a loss function calculated on inputs and their class labels. Besides, task-free continual learning does not require models to identify distribution shifts. They do not distinguish samples with the same class label even though they are sampled from different distributions. On the contrary, in CoIL, data of different tasks share the same label space, and models must be able to distinguish different distributions.

C. OOD Detection

A robust machine learning system should not only output results with high quality but also detect unknown inputs and reject them. To achieve this goal, the OOD detection identifies test data that are sampled from a shifted distribution. A baseline method [10] detects OOD samples based on the maximum softmax probabilities since a well-trained neural network tends to assign higher softmax probabilities to InD samples than those to OOD samples. ODIN [36] improves the baseline with temperature scaling and input preprocessing to amplify the separability of InD and OOD samples. The energy-based detection method [37] uses an energy function instead of a softmax function as the detection score, and the energy function also maps the logit outputs to a scalar. DAGMM [38] uses an autoencoder to generate low-dimensional representations and detects OOD samples with the reconstructions from these representations. To enhance the sensitivity to distribution shifts, some methods focus on the hidden representations in the middle layers of neural networks. They leverage techniques like layerwise Mahalanobis distance [39] and gram matrix [40]. Some works [11], [41], [42] leverage a dataset of known OOD samples to further improve detection performance. Nevertheless, these methods are designed to detect OOD samples based on a well-trained single-task model. To detect distribution shifts, we propose a continual OOD detection method that works on models that learn tasks incrementally and suffer from catastrophic forgetting.

III. Continuum Incremental Learning

In this section, we formally describe the CoIL scenario. We also compare CoIL with other CL scenarios, including class-incremental learning and task-free continual learning.

TABLE I
DISTINCTIONS BETWEEN COIL AND OTHER CONTINUAL LEARNING SCENARIOS

	Training Sample	Label Space	Prediction	Distribution Shift	Task Boundary
Class-Incremental learning	$(\mathbf{x}, y, t)$	$\mathcal{Y}_i \cap \mathcal{Y}_j = \emptyset$ for $i \neq j$	y	Notified	Known
Task-Free Continual Learning	$(\mathbf{x}, y)$	$\mathcal{Y}_i \cap \mathcal{Y}_j = \emptyset$ for $i \neq j$	y	Notified	Unknown
Continuum Incremental Learning	$(\mathbf{x}, y)$	$\mathcal{Y}_i = \mathcal{Y}_j$ for $\forall i, j$	y and t	Unnotified	Unknown

A. Problem Settings

Here, we introduce CoIL. A model $f(\mathbf{x}; \theta)$ parameterized by θ learns a sequence of T tasks. Task t is composed of a training set $D_t = (\mathbf{x}_t^i, y_t^i)_{i=1}^{N_t}$ with N_c classes. $\mathbf{x}_t^i \in \mathcal{X}_t$, $y_t^i \in \mathcal{Y}_t$, and N_t denote the input data, the class label, and the number of training samples in task t, respectively. Note that all class labels from different tasks share the same label space, i.e., $\mathcal{Y}_i = \mathcal{Y}_j$ for $\forall i, j$. As a result, we cannot identify a task boundary by the change of the label space, and the distribution shifts between tasks are unnotified.

Since the task boundaries are unknown, data are not organized by tasks. Instead, the incremental unit is a continuum. A continuum is either concatenated by N_k training sets $\{D_{k_1}, D_{k_2}, \ldots, D_{k_{N_k}}\}$ of different tasks $k_1, k_2, \ldots, k_{N_k}$, or only a subset (maybe a whole set) of a training set D_{k_1} of task k_1. In this case, task boundaries could be found anywhere in continua. During training, the model f learns from a continuum of data at each step. At test time, the model f predicts both a class label y and a task identifier t for a test input $\mathbf{x}$ of any learned task.

The main obstacle of CoIL is how to identify distribution shifts during training. Once a new task is detected, we can assign a new label space to the new task and train model f with existing continual learning approaches. At test time, f predicts a scalar in the extended label space, and the scalar can be interpreted as a class label and a task identifier.

B. Comparisons With Other Continual Learning Scenarios

To better illustrate the distinctions between CoIL and some related settings, we introduce class-incremental learning and task-free continual learning and compare them with CoIL. We summarize the distinctions in Table I.

1) Class-Incremental Learning: A model $f(\mathbf{x}; \theta)$ parameterized by θ learns a sequence of T tasks. Each task is composed of a training set $D_t = (\mathbf{x}_t^i, y_t^i)_{i=1}^{N_t}$ with N_c classes and a task identifier t. Note that label spaces of tasks do not overlap, i.e., $\mathcal{Y}_i \cap \mathcal{Y}_j = \emptyset$ for $\forall i \neq j$. The goal of class-incremental learning is to incrementally update the model f so that it can predict a class label y for a test input $\mathbf{x}$ of any learned task.

In class-incremental learning, the incremental unit is a task. Hence, task boundaries are known during training. On the contrary, the only assumption in CoIL is that adjacent data in continua are sampled from the same distribution unless an unknown task boundary separates them. In addition, the model has to predict task identifiers without task supervision in CoIL, while task labels are available during training in class-incremental learning.

2) Task-Free Continual Learning: A model $f(\mathbf{x}; \theta)$ parameterized by θ learns a sequence of mini-batch data $(\mathbf{x}_{ts}^i, y_{ts}^i, t_{ts}^i)_{i=1}^{N_{ts}}$ which sequentially arrive at each timestamp ts, and the data form a nonstationary data stream [43]. Each sample is associated with a latent task identifier t_{ts}, where $\mathbf{x}_{ts}$ denotes the input sample received at timestamp ts, y_{ts} is the data label associated with $\mathbf{x}_{ts}$, and N_{ts} is the number of training samples at timestamp ts. In a more general definition of task-free CL, data distributions might shift gradually without clear task boundaries [12]. Similar to class-incremental learning, label spaces of different tasks do not overlap. During training and testing, the task identifier t_{ts} is unavailable to the learner. At test time, the model is required to predict a class label y for a test input $\mathbf{x}$ of any learned task.

In task-free continual learning, task boundaries remain unknown until label space changes. The arrival of new classes implies that distribution shifts happen even when task boundaries are blurry. However, in CoIL, all class labels from different tasks share the same label space, so the learner has to detect distribution shifts with only the input $\mathbf{x}$.

IV. TASK BOUNDARY IDENTIFICATION WITH OOD DETECTION

In this section, we focus on how to identify task boundaries with OOD detection and how to leverage this technique to solve CoIL. We first design the continual OOD detection method to identify task boundaries. Then, we show how to incorporate task boundary identification with existing continual learning approaches.

A. Continual OOD Detection

To identify distribution shifts, softmax-based OOD detection methods [10], [11], [36] have been proven simple and effective without introducing extra network structures. The rationale is that a well-trained neural network tends to assign higher softmax probabilities to InD samples than those to OOD samples. However, in continual learning, models suffering from catastrophic forgetting often misclassify samples of old tasks, which are apparently not "well-trained." Furthermore, models in CoIL have to alternate between detecting OOD samples in continua and incrementally learning new tasks, while previous OOD detection methods detect OOD samples in a test dataset.

Therefore, we design an OOD detection method for continual learning, as illustrated in Fig. 1. To detect whether a batch of samples X is OOD to the latest learned task t, the model f evaluates X and a stored batch of samples X_t^{InD} of task t. To utilize the well-trained part of the model for OOD detection, we calculate the local probabilities on the

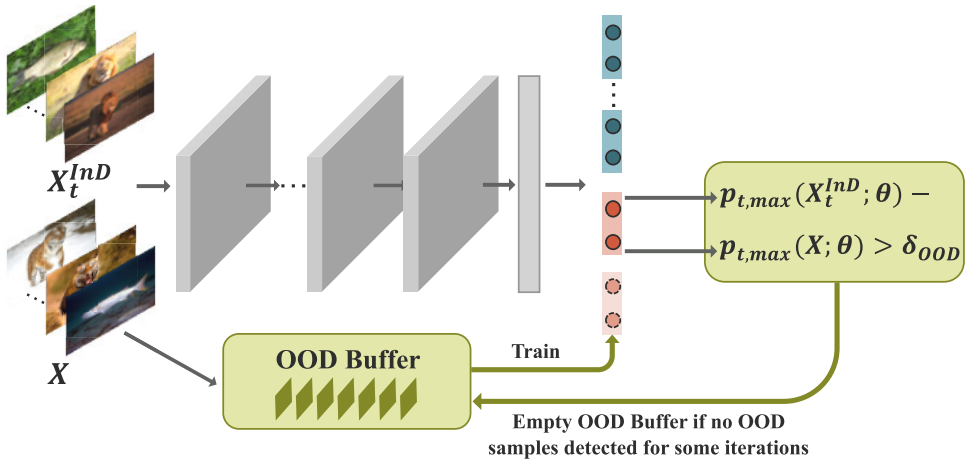

Fig. 1. Overview of continual OOD detection and task boundary identification. We design an adaptive threshold and only use the probabilities on the latest task t for OOD detection. Once the OOD buffer is full, we add new logits and train it with data in the buffer. Then, we continue to detect OOD samples based on the new logits.

tth head of the multihead classifier only with logits of task t, i.e., the $[N_c * (t-1)]$th to the $(N_c * t - 1)$th outputs (before softmax) of model f, where N_c is the number of classes per task. X is considered to be OOD to task t when there is a large gap between the max probabilities of X and X_t^{InD}. Formally, we define the ith component of the probability vector $p_t^\tau(\mathbf{x};\theta) \in \mathbb{R}^{N_c}$ as

$$p_{t,i}^\tau(\mathbf{x};\theta) = \frac{e^{f_{t,i}(\mathbf{x};\theta)/\tau}}{\sum_{j=1}^{N_c} e^{f_{t,j}(\mathbf{x};\theta)/\tau}} \tag{1}$$

where τ is a temperature scaling factor and $f_{t,i}(\mathbf{x};\theta)$ is the ith output logit of task t. Then, we define

$$p_{t,\max}^\tau(\mathbf{x};\theta) = \max_i p_{t,i}^\tau(\mathbf{x};\theta) \tag{2}$$

for a single input $\mathbf{x}$ and

$$p_{t,\max}^\tau(X;\theta) = \frac{1}{|X|}\sum_{\mathbf{x}\in X} p_{t,\max}^\tau(\mathbf{x};\theta) \tag{3}$$

for a batch of inputs X. Given the definitions above, the continual OOD detector for the latest learned task t can be described as

$$g\left(X, X_t^{\text{InD}}, t\right) = \begin{cases} 1, & \text{if } p_{t,\max}^\tau(X;\theta) < p_{t,\max}^\tau\left(X_t^{\text{InD}};\theta\right) - \delta_{\text{OOD}} \\ 0, & \text{otherwise.} \end{cases} \tag{4}$$

A batch of input X is considered as OOD samples on task t when $p_{t,\max}^\tau(X;\theta)$ is smaller than an adaptive threshold $p_{t,\max}^\tau(X_t^{\text{InD}};\theta) - \delta_{\text{OOD}}$. The threshold is calibrated by a small number of InD samples X_t^{InD} since the model may output different ranges of probabilities on different tasks. Here, we assume that data in the same batch are sampled from the same distribution, which is also adopted by previous works on continual learning [8], [33], [44].

As mentioned above, previous OOD detection methods are designed for a well-trained neural network. Hence, it is natural to detect OOD samples only on the latest learned task, where the model tends to perform well without catastrophic forgetting.

Algorithm 1 Task Boundary Identification

Input: Model f, continuum C_k, latest learned task t, InD samples X^{InD} and its size N_{InD}, number of tasks learned so far N, OOD buffer capacity O, max iterations between two OOD batches I,
Output: Dataset D

$f' = \text{DeepCopy}(f)$; $D = \emptyset$; $\mathcal{O} = \emptyset$; $n = N + 1$
for X in C_k **do**
  $\mathcal{O} = \mathcal{O} \cup X$
  if $n = 1$ **then**
    Recognize X as OOD samples
  else
    Calculate $g(X, X^{\text{InD}}, t)$ by Eq. (4)
  end if
  if No OOD samples detected in last I iterations **then**
    Assign task identifier t to samples in $\mathcal{O}$
    $D = D \cup \mathcal{O}$; $\mathcal{O} = \emptyset$
  end if
  if $|\mathcal{O}| > O$ **then**
    Assign task identifier n to samples in $\mathcal{O}$
    Train f' with $\mathcal{O}$
    Randomly select N_{InD} samples in $\mathcal{O}$ as X^{InD}
    $t = n$; $n = n + 1$; $D = D \cup \mathcal{O}$; $\mathcal{O} = \emptyset$
  end if
end for
Assign task identifier t to samples in $\mathcal{O}$;
$D = D \cup \mathcal{O}$; $\mathcal{O} = \emptyset$
return D

B. Continual Learning With OOD Detection

The learning procedure in CoIL can be split into two phases: task identifier prediction and model training. Once a new continuum arrives, we first assign task identifiers to inputs. Then, we train the model f with existing continual learning approaches. Here, we only consider the situation where all data from the same task are consecutive, so identifying a task

Algorithm 2 Task Identifier Prediction

Input: Model f, latest learned task t', dataset D returned in Algorithm 1, number of tasks learned so far N
Output: Dataset D'

$D' = \emptyset$; $n = N + 1$
for s in all task identifiers appear in D **do**
  X_s = all samples with task identifier s in D
  if $N \leq 1$ **or** $s = t'$ **then**
    Assign task identifier s to X_s
  else
    Calculate $h(X_s, X_{t'}^{\text{InD}}, t)$ by Eq. (5) for each task t except t'
    if $h(X_s, X_{t'}^{\text{InD}}, t) = 1$ for any t **then**
      Assign task identifier $\arg\max_t p_{t,\max}^{\tau}(X_s;\theta) - p_{t,max}^{\tau}(X_{t'}^{\text{InD}};\theta)$ to X_s
    else
      Assign task identifier n to X_s; $n = n + 1$
    end if
  end if
  $D' = D' \cup X_s$
end for
return D'

boundary implies a new task arrives. We will investigate a more general setting in Section V.

Algorithm 1 describes the task boundary identification process. To increase the stability of continual learning and the resistance to OOD detection errors, a task boundary is identified only when enough inputs in a consecutive sequence are detected as OOD samples. To this end, we employ an OOD buffer $\mathcal{O}$ to collect data. Note that no extra memory for $\mathcal{O}$ is required because data in C_k are stored in memory before finishing learning C_k. Once $\mathcal{O}$ reaches the maximum capacity O, we add new logits on the model f' and train f' with data in $\mathcal{O}$. After that, a small number of samples in the buffer is stored as X^{InD}, and $\mathcal{O}$ is emptied. We continue to process the rest data in C_k and calculate probabilities on the new logits for OOD detection. This procedure is repeated until all data in C_k are processed.

V. Continual Learning With Unknown Task Boundary

In this section, we propose CLUTaB for a more challenging task-reappear setting of CoIL. First, we propose the continual InD detection method for task identifier prediction. Then, we design OOD loss to improve InD detection accuracy. Finally, we propose the two-step inference at test time based on InD detection to improve the continual learning performance.

A. Task Identifier Prediction With Continual InD Detection

We investigate the task-reappear setting where a continuum of extra data from an already learned task may arrive. In this case, a task boundary implies the arrival of a new or old task. Since all class labels share the same label space in CoIL, we need to determine whether a set of data is sampled from a new distribution or any learned distribution. To this end, we propose a method named CLUTaB, which also consists of a task identifier prediction phase and a model training phase.

For task identifier prediction, we design a continual InD detection method. Calibrated by $X_{t'}^{\text{InD}}$, the InD detection threshold $p_{t,\max}^{\tau}(X_{t'}^{\text{InD}};\theta) + \delta_{\text{InD}}$ is adaptive to different tasks, where t' denotes the latest learned task. Here, we consider $X_{t'}^{\text{InD}}$ as OOD samples on task t where $t \neq t'$ since they are sampled from a different distribution. The continual InD detector for task t can be described as

$$h(X, X_{t'}^{\text{InD}}, t) = \begin{cases} 1, & \text{if } p_{t,\max}^{\tau}(X;\theta) > p_{t,\max}^{\tau}\left(X_{t'}^{\text{InD}};\theta\right) + \delta_{\text{InD}} \\ 0, & \text{otherwise} \end{cases} \tag{5}$$

where $t \neq t'$. A set of input X is considered as InD samples on task t when the max probabilities of X surpass those of OOD samples by a margin δ_{InD}.

Algorithm 2 describes the task identifier prediction phase, which runs after Algorithm 1. To further improve the resistance to InD detection errors, the InD detection unit here is a set of all samples between two task boundaries instead of a batch. We use (5) to detect whether the set of data X_s is sampled from any learned distribution.

B. OOD Loss

To improve InD detection performance, we propose an OOD loss to expand the probability discrepancy between ID and OOD samples. We regard samples from other tasks as OOD samples on task t and define the OOD loss

$$l_t(X_s) = \max\left(m - \frac{1}{|X_s|}\sum_{\mathbf{x}\in X_s} H(p_t(\mathbf{x};\theta)), 0\right) \tag{6}$$

where X_s is a set of data labeled with task identifier s, and H is the entropy over the probability distribution. To prevent the model from forgetting knowledge learned on task s, we use a hyperparameter m to stop penalizing the model when the probability distribution of OOD samples is flat enough. Given the definition of l_t, the overall OOD loss is formulated as

$$L_{\text{OOD}}(X) = \frac{1}{N(N-1)}\sum_{t=1}^{N}\sum_{s=1}^{N}\mathbb{1}\{t \neq s\} l_t(X_s) \tag{7}$$

where N is the number of tasks so far. Note that we only apply the OOD loss when $N > 1$. In practice, we leverage exemplars to make sure that X contains samples of all tasks.

For model training, we incorporate taskwise knowledge distillation and separated-softmax training [20] with the OOD loss. When learning C_k with data from a set of tasks $\mathcal{T}$, the loss is

$$\begin{aligned} &L_{\text{SS}}(X) \\ &= \frac{1}{|X|}\sum_{(\mathbf{x},y,t)\in X}\sum_{s=1}^{N}\mathcal{D}_{\text{KL}}\left(p_s^{\tau'}(\mathbf{x};\theta_{\text{old}})||p_s^{\tau'}(\mathbf{x};\theta)\right)\mathbb{1}\{s \notin \mathcal{T}\} \\ &\quad + \mathcal{D}_{\text{KL}}(\mathbf{y}_t||p_t(\mathbf{x};\theta))\mathbb{1}\{t \in \mathcal{T}\} \\ &\quad + \mathcal{D}_{\text{KL}}\left(\mathbf{y}_{/\mathcal{T}}||p_{/\mathcal{T}}(\mathbf{x};\theta)\right)\mathbb{1}\{t \notin \mathcal{T}\} \\ &\quad + \lambda L_{\text{OOD}}(X) \end{aligned} \tag{8}$$

where N is the number of tasks so far, $\mathcal{D}_{\mathrm{KL}}(\cdot||\cdot)$ is the Kullback–Leibler divergence, τ' is a temperature scaling factor for knowledge distillation, θ_{old} are the model parameters stored for knowledge distillation, $p_t(\mathbf{x};\theta)$ is short for $p_t^{\tau}(\mathbf{x};\theta)$ when $\tau = 1$, $\mathbf{y}_t$ is a one-hot vector in $\mathbb{R}^{N_c}$ that has value one at the yth coordinate, $\mathbf{y}_{/\mathcal{T}}$ is a one-hot vector in $\mathbb{R}^{N-|\mathcal{T}|\cdot N_c}$ that has one at the corresponding coordinate according to the class label y and the predicted task identifier t, and $p_{/\mathcal{T}}$ is a vector of probabilities calculated on all logits except the logits of tasks in $\mathcal{T}$.

C. Two-Step Inference

Here, we propose the two-step inference at test time. For a batch of test samples X, we first leverage continual InD detection to predict a task identifier $\hat{t}$ by inferring on all heads of the multihead classifier, and then predict class labels by $p_{\hat{t}}(\mathbf{x};\theta)$ for $\mathbf{x} \in X$. We describe the task identifier prediction in two-step inference as

$$\hat{t} = \begin{cases} t', & \text{if } g(X, X_{t'}^{\mathrm{InD}}, t') = 0 \\ \arg\max_t p_{t,\max}^{\tau}(X;\theta) - p_{t,\max}^{\tau}\left(X_{t'}^{\mathrm{InD}};\theta\right), & \\ & \text{otherwise} \end{cases} \tag{9}$$

where t' is the latest learned task. It assumes that adjacent data at test time are highly correlated, which is also adopted in previous continual learning studies [8], [44]. Compared to the original one-step single-head inference in which the model predicts a task identifier and a class label at the same time, our two-step inference can greatly improve performance without adding extra model parameters.

VI. Experiment

In this section, we compare our methods with other continual learning methods on CIFAR-100 [45] and mini-ImageNet [46]. We also perform ablation experiments to analyze the effects of the OOD loss and the performance of two-step inference with different test batch sizes.

A. Dataset Details

We use two widely used benchmark datasets: CIFAR-100 and mini-ImageNet. Both CIFAR-100 and mini-ImageNet contain 60 000 32×32 images of 100 classes. Each class has 500 training images and 100 testing images. Unless otherwise stated, the class order is randomly shuffled as in iCarl [28]. We first split 100 classes into T tasks and remap the class labels to make samples of different tasks that share the same label space if the experiment runs in CoIL, and then reorganize them into K continua. We take different values of T and K, and use different reorganization methods.

1) Consecutive Split/Concatenation: Data are split into T tasks. If $T < K$, data in each task are split into $\frac{K}{T}$ continua; otherwise, data in every $\frac{T}{K}$ task are concatenated into one continuum. The order of these continua is not changed. The sequence of continua can be denoted as $\{t_1, t_1', \ldots, t_2, t_2', \ldots, t_T, t_T', \ldots\}$ or $\{t_{1,\ldots,k}, t_{k+1,\ldots,2k}, \ldots, t_{T-k+1,\ldots,T}\}$.

2) Reappearing Split: Data are split into T tasks, and then, data in each task are split into $\frac{K}{T}$ continua. We reorder the continua so that each task reappears successively. The sequence of continua can be denoted as $\{t_1, t_2, \ldots, t_T, t_1', t_2', \ldots, t_T', \ldots\}$.

B. Implementation Details

We use a 32-layer ResNet [47] for CIFAR-100 and a reduced 18-layer Resnet for mini-ImageNet. A multihead classifier is adopted and each head is a single fully-connected layer. We train the model for 160 epochs for each task. The learning rate starts from 0.1 initially and reduces to 0.01 and 0.001 after 80 and 120 epochs, respectively. The training batch size is 128 and the batch size of experience replay is 32. The weight decay is set to 0.0002. We adopt random cropping and horizontal flip for data augmentation following the original ResNet implementation. We randomly select 20 samples as exemplars for each class. For continual OOD detection and task boundary identification, we use $\delta_{\mathrm{OOD}} = 0.07$, $N_{\mathrm{InD}} = 64$, $O = 2000$, and $I = 5$. For continual InD detection and task identifier prediction, we use $m = 2$, 1.5, and 0.8 for $T = 5$, 10, and 20 in L_{OOD}, $\delta_{\mathrm{InD}} = 0.12$ and 0.1 for CIFAR-100 and mini-ImageNet, and $\lambda = 2$, $\tau' = 2$ in all experiments. $\tau = 1$ for both continual OOD and InD detection. The test batch size is 32 in two-step inference.

C. Evaluation

To compare with existing continual learning methods in CoIL, a simple trick is adopted: every time a continuum arrives, these methods treat it as a new task. For a test sample $(\mathbf{x}, y, t)$, we use the following accuracy metrics.

1) Task Accuracy A_t: The model predicts a class label $\hat{y}$ given task identifier t, which means the model can only focus on the corresponding distribution. $A_t = 1$ when $\hat{y} = y$.

2) Domain Accuracy A_d: The model only predicts a class label $\hat{y}$ without task identifier t. The model has no knowledge of which distribution the sample follows but is also not required to figure it out. $A_d = 1$ when $\hat{y} = y$.

3) Class Accuracy A_c: The model predicts both a class label $\hat{y}$ and a task identifier $\hat{t}$. $A_c = 1$ if and only if $\hat{y} = y$ and $\hat{t} = t$.

For CLUTaB, we report A_c of both one-step and two-step inference, denoted as $A_{c,1}$ and $A_{c,2}$, respectively. Other methods cannot adopt two-step inference, so we only report A_c of one-step inference. Furthermore, to evaluate InD detection performance, we report task accuracy A_t as an upper bound for $A_{c,2}$, where task identifiers are given [instead of being predicted by (9)] to help predict class labels. All accuracy metrics above are calculated after learning the latest task and averaged over test data of all learned tasks.

We also use the backward transfer (BWT) to measure catastrophic forgetting. Let A denote an accuracy metric, $A_{t,k}$ denote the model accuracy of task t after learning continuum k, $\mathcal{T}$ denote the set of all tasks, and $\mathcal{T}_k$ denote the set of tasks that appear in continuum k, and then, the BWT of A is

$$\mathrm{BWT} = \frac{1}{|\mathcal{T} - \mathcal{T}_K|} \sum_{t \in (\mathcal{T} - \mathcal{T}_K)} A_{t,K} - \max_k A_{t,k}. \tag{10}$$

TABLE II
RESULTS OF CONSECUTIVE SPLIT ON CIFAR-100 AND MINI-IMAGENET. $T = 5$ AND $K = 10$

Methods	CIFAR-100				mini-ImageNet			
	A_d	BWT of A_d	A_c	BWT of A_c	A_d	BWT of A_d	A_c	BWT of A_c
Finetuning	20.12 ± 0.23	-73.44 ± 0.19	-	-	19.38 ± 0.19	-67.85 ± 0.57	-	-
Joint	72.27 ± 0.18	-	-	-	69.97 ± 0.16	-	-	-
ER	38.43 ± 0.18	-49.72 ± 0.43	-	-	32.14 ± 0.49	-51.20 ± 0.74	-	-
ER w. OOD Detection	**41.29 ± 0.27**	**-46.54 ± 0.26**	**38.46 ± 0.29**	**-50.49 ± 0.44**	**35.87 ± 0.22**	**-47.85 ± 0.32**	**33.08 ± 0.18**	**-50.80 ± 0.51**
LwF	36.78 ± 0.70	-53.12 ± 0.47	-	-	28.53 ± 0.31	-55.25 ± 0.66	-	-
LwF w. OOD Detection	**42.97 ± 0.35**	**-43.96 ± 0.28**	**40.24 ± 0.41**	**-47.14 ± 0.08**	**35.78 ± 0.67**	**-46.27 ± 0.87**	**32.93 ± 0.56**	**-49.71 ± 0.47**
iCaRL	50.32 ± 0.34	-21.78 ± 0.91	-	-	45.16 ± 0.55	-18.63 ± 0.84	-	-
iCaRL w. OOD Detection	**51.94 ± 1.38**	**-17.98 ± 1.57**	**48.85 ± 2.74**	**-19.33 ± 1.32**	**46.05 ± 1.57**	**-16.20 ± 1.24**	**44.02 ± 1.66**	**-16.92 ± 1.53**
BiC	49.10 ± 0.36	-30.77 ± 0.39	-	-	40.90 ± 0.84	-37.59 ± 0.98	-	-
BiC w. OOD Detection	**52.29 ± 0.34**	**-20.73 ± 0.15**	**50.16 ± 0.37**	**-22.36 ± 0.19**	**47.69 ± 0.94**	**-23.80 ± 0.53**	**45.54 ± 0.91**	**-25.42 ± 0.68**
SS-IL	48.77 ± 0.51	-19.83 ± 0.33	-	-	42.78 ± 0.94	-23.17 ± 1.22	-	-
SS-IL w. OOD Detection	**50.42 ± 0.30**	**-18.35 ± 0.34**	**48.10 ± 0.28**	**-16.63 ± 0.24**	**45.62 ± 0.22**	**-20.87 ± 0.36**	**43.37 ± 0.22**	**-21.44 ± 0.80**

TABLE III
RESULTS OF CONSECUTIVE CONCATENATION ON CIFAR-100 AND MINI-IMAGENET. $T = 20$ AND $K = 10$

Methods	CIFAR-100				mini-ImageNet			
	A_d	BWT of A_d	A_c	BWT of A_c	A_d	BWT of A_d	A_c	BWT of A_c
Finetuning	25.73 ± 0.36	-69.01 ± 0.24	-	-	25.32 ± 0.19	-65.05 ± 0.38	-	-
Joint	76.56 ± 0.21	-	-	-	75.30 ± 0.20	-	-	-
ER	45.98 ± 0.34	-46.34 ± 0.58	-	-	40.50 ± 0.60	-47.76 ± 0.94	-	-
ER w. OOD Detection	**48.71 ± 0.12**	**-44.41 ± 0.43**	**35.90 ± 0.20**	**-56.68 ± 0.37**	**43.11 ± 0.13**	**-46.41 ± 0.22**	**30.23 ± 0.42**	**-55.46 ± 0.39**
LwF	46.63 ± 0.34	-44.88 ± 0.53	-	-	38.30 ± 0.53	-49.02 ± 1.45	-	-
LwF w. OOD Detection	**48.54 ± 0.41**	**-43.21 ± 0.73**	**35.30 ± 0.52**	**-55.72 ± 0.66**	**41.03 ± 0.44**	**-47.48 ± 0.86**	**27.34 ± 0.51**	**-60.07 ± 0.96**
iCaRL	52.20 ± 0.56	-21.57 ± 0.45	-	-	46.43 ± 0.69	-15.64 ± 1.06	-	-
iCaRL w. OOD Detection	**56.87 ± 0.77**	**-17.87 ± 0.85**	**45.26 ± 1.05**	**-22.97 ± 1.52**	**52.27 ± 0.54**	**-14.68 ± 0.49**	**41.16 ± 0.81**	**-18.90 ± 0.76**
BiC	53.44 ± 0.51	-31.66 ± 0.71	-	-	46.62 ± 0.67	-36.74 ± 0.99	-	-
BiC w. OOD Detection	**57.76 ± 0.59**	**-26.56 ± 0.49**	**46.41 ± 0.72**	**-38.46 ± 1.13**	**50.37 ± 0.55**	**-32.69 ± 0.88**	**38.86 ± 0.77**	**-41.47 ± 0.51**
SS-IL	52.71 ± 0.06	-15.73 ± 0.14	-	-	48.58 ± 0.34	-13.98 ± 0.29	-	-
SS-IL w. OOD Detection	**55.33 ± 0.50**	**-14.06 ± 0.39**	**43.47 ± 0.31**	**-16.58 ± 0.35**	**50.65 ± 0.89**	**-13.24 ± 0.25**	**38.70 ± 1.24**	**-16.81 ± 0.56**

TABLE IV
RESULTS OF TASK-REAPPEAR SETTINGS ON CIFAR-100 AND MINI-IMAGENET. WE REPORT A_d UNLESS OTHERWISE STATED IN THE TABLE

Methods	Reappearing $T = 5$, $K = 10$				Reappearing $T = 10$, $K = 20$			
	CIFAR-100		mini-ImageNet		CIFAR-100		mini-ImageNet	
	Accuracy	BWT	Accuracy	BWT	Accuracy	BWT	Accuracy	BWT
Finetuning	19.86 ± 0.11	-73.33 ± 0.30	19.38 ± 0.20	-69.65 ± 0.54	15.52 ± 0.67	-73.69 ± 1.36	16.74 ± 0.52	-67.39 ± 1.20
Joint	71.67 ± 0.53	-	70.07 ± 0.58	-	74.23 ± 0.35	-	72.10 ± 0.33	-
ER	38.81 ± 0.43	-48.33 ± 0.59	32.99 ± 0.51	-49.45 ± 0.66	40.30 ± 0.67	-45.58 ± 0.89	33.13 ± 0.82	-49.19 ± 1.57
LwF	39.64 ± 1.03	-44.15 ± 1.17	32.19 ± 0.43	-48.10 ± 0.71	38.49 ± 0.80	-46.56 ± 1.36	30.69 ± 0.32	-47.54 ± 0.55
iCaRL	49.39 ± 0.98	**-11.00 ± 0.32**	44.07 ± 0.99	**-14.75 ± 0.41**	44.00 ± 1.82	**-13.28 ± 0.77**	39.44 ± 1.39	**-13.72 ± 0.54**
BiC	50.52 ± 0.68	-18.93 ± 0.95	45.02 ± 0.52	-22.83 ± 0.34	46.66 ± 0.56	-25.25 ± 0.78	38.85 ± 1.88	-29.84 ± 2.63
SS-IL	49.27 ± 0.24	-16.84 ± 0.16	43.83 ± 1.65	-16.47 ± 0.97	46.34 ± 0.50	-16.87 ± 0.31	40.12 ± 1.01	-17.67 ± 0.73
CLUTaB	**51.95 ± 0.63**	-16.73 ± 0.49	**47.56 ± 1.61**	-16.13 ± 0.97	**49.35 ± 0.68**	-16.27 ± 1.26	**42.73 ± 1.79**	-17.15 ± 1.44
CLUTaB ($A_{c,1}$)	49.84 ± 0.70	-17.68 ± 0.54	44.73 ± 2.12	-19.83 ± 1.39	43.44 ± 0.56	-18.72 ± 0.87	36.55 ± 2.98	-20.59 ± 1.70
CLUTaB ($A_{c,2}$)	**72.46 ± 1.02**	**-3.47 ± 1.58**	**65.07 ± 3.50**	**-3.41 ± 2.36**	**77.30 ± 0.60**	**-3.67 ± 0.43**	**68.03 ± 3.66**	**-4.58 ± 2.46**
CLUTaB (A_t)	**73.03 ± 0.48**	**-5.65 ± 0.33**	**68.02 ± 1.38**	**-5.43 ± 1.57**	**77.81 ± 0.24**	**-5.39 ± 0.56**	**70.18 ± 1.42**	**-5.71 ± 1.49**

D. Main Results

1) Results for Consecutive Split/Concatenation: Tables II and III show the results in the consecutive settings. We compare ER [5], LwF [29], iCaRL [28], BiC [6], and SS-IL [20] with their corresponding versions that leverage continual OOD detection for task boundary identification. We also include finetuning and joint training [9] as baselines. Since the original continual learning methods cannot identify task boundaries during training, they predict meaningless task identifiers, so we do not report A_c for them. Compared A_c with A_d in the same settings, methods with OOD detection can predict task identifiers with high accuracy for samples whose $\hat{y} = y$. Furthermore, methods with OOD detection also perform better on A_d, which means that continual OOD detection does find better timings for knowledge consolidation.

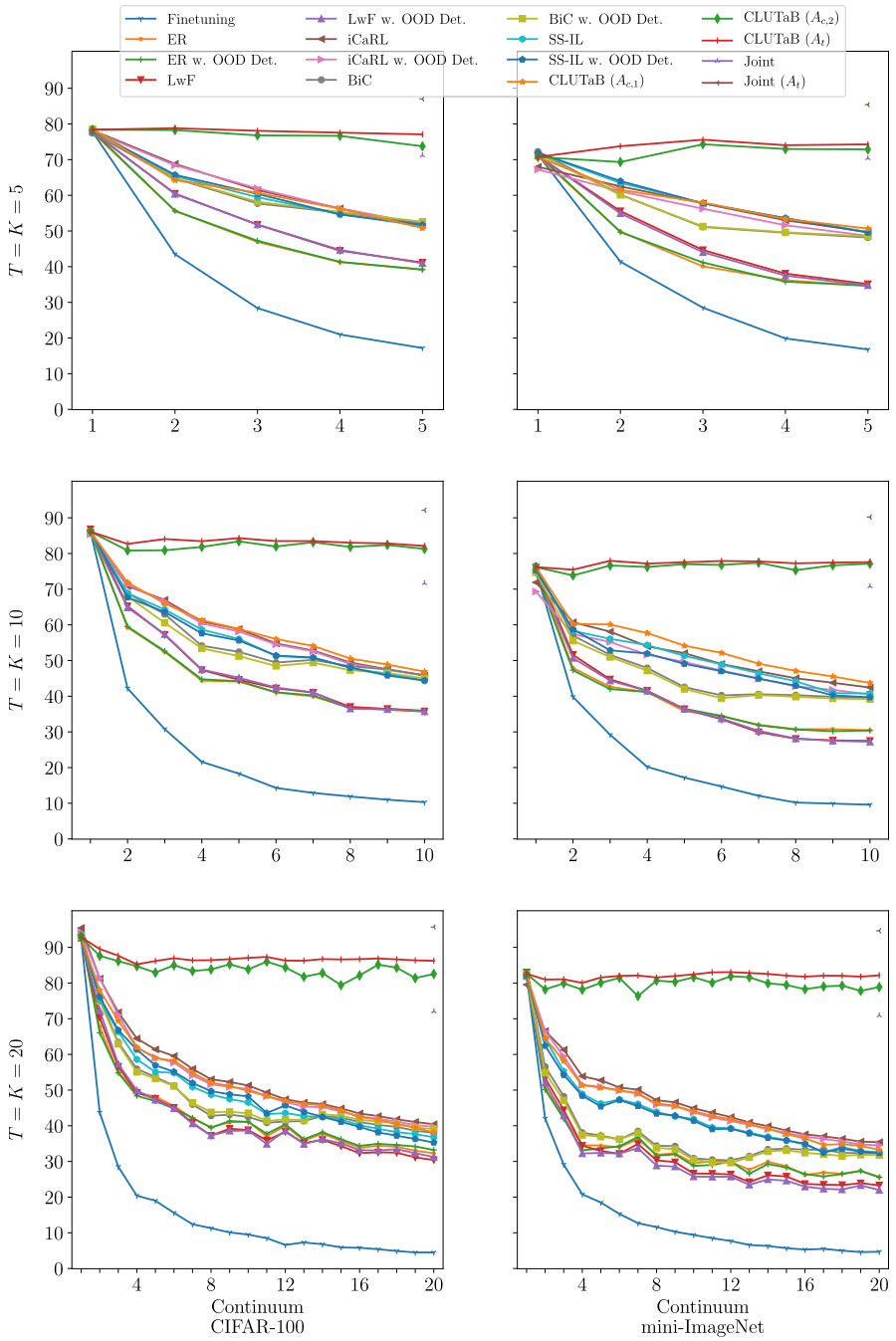

Fig. 2. Results for aligned tasks and continua. We report A_c unless otherwise stated on CIFAR-100 and mini-ImageNet when $T = K = 5$, 10, and 20.

2) Results for Reappearing Split: Table IV shows the results of task-reappear settings. We report $A_{c,1}$, $A_{c,2}$, and A_t for CLUTaB, and A_d for all methods. We observe that CLUTaB with two-step inference significantly surpasses methods with one-step inference. Note that $A_{c,2}$ is close to its upper bound A_t, indicating high-quality InD detection at test time. Besides, CLUTaB still outperforms other methods on A_d. This implies that task identifier prediction helps the model better recapture the forgotten knowledge.

3) Results for Aligned Tasks and Continua: We also perform experiments in a setting where each C_k is exactly D_k of task k. In Fig. 2, we report $A_{c,1}$, $A_{c,2}$, and A_t for CLUTaB, A_c and A_t for joint training, and A_c for all the other methods when $T = K = 5$, 10, and 20. In this case, task boundaries are known to ER, LwF, iCaRL, BiC, and SS-IL, while their corresponding versions with continual OOD detection and CLUTaB still run in the CoIL scenario. Although task boundaries are unknown to methods with OOD detection, they still perform almost as well as their original versions, which means continual OOD detection can identify proper task boundaries. Like the results of task-reappear settings, CLUTaB with two-step inference still outperforms other methods even including the joint training. It is because we assume that adjacent data at test time are highly correlated and our batchwise task identifier prediction achieves high accuracy.

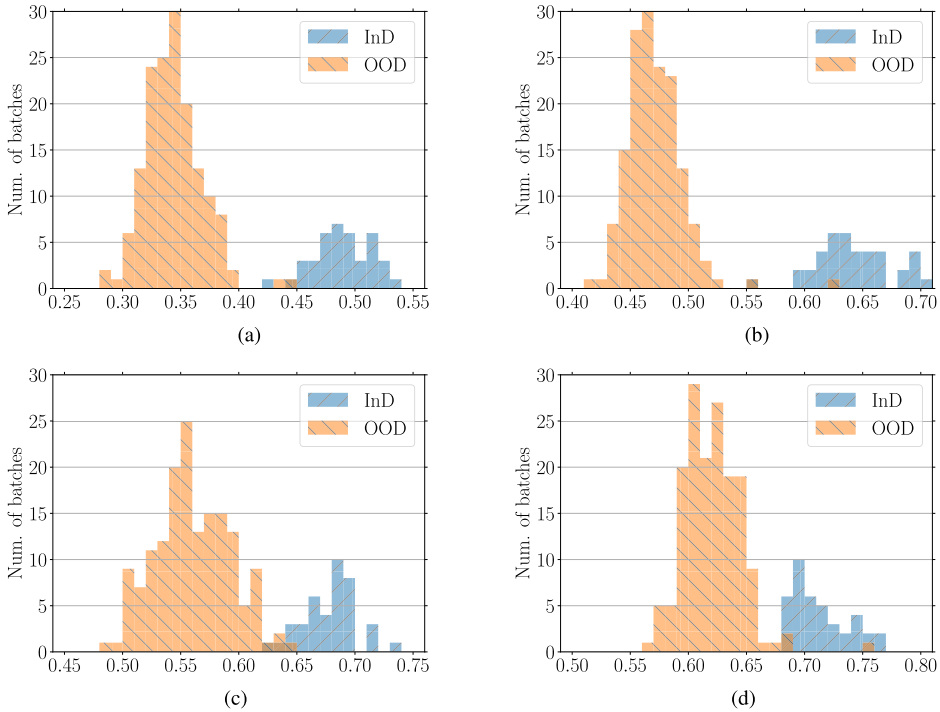

Fig. 3. Distributions of maximum probabilities $p_{t,\max}(X_t^{\text{InD}})$ and $p_{t,\max}(X_t^{\text{OOD}})$ with and without L_{OOD} after learning first five continua in the $T = 5$, $K = 10$, task-reappear setting by SS-IL. X_t^{InD} are data of task t in the left five continua and X_t^{InD} are the others. (a) $t = 0$, with L_{OOD}. (b) $t = 4$, with L_{OOD}. (c) $t = 0$, without L_{OOD}. (d) $t = 4$, without L_{OOD}.

TABLE V
RESULTS OF FINE-GRAINED CONTINUAL LEARNING ON CIFAR-100. WE REPORT A_d UNLESS OTHERWISE STATED IN THE TABLE

Methods	Accuracy	BWT
Finetuning	18.92 ± 0.38	-75.44 ± 0.65
Joint	71.63 ± 0.11	-
ER	38.69 ± 0.16	-48.03 ± 0.34
LwF	39.57 ± 0.11	-46.03 ± 0.08
iCaRL	49.51 ± 1.54	**-12.85 ± 1.51**
BiC	50.57 ± 0.43	-20.15 ± 0.67
SS-IL	49.00 ± 0.67	-16.88 ± 0.92
CLUTaB	**52.05 ± 0.81**	-16.97 ± 1.19
CLUTaB ($A_{c,1}$)	49.94 ± 0.80	-19.10 ± 1.21
CLUTaB ($A_{c,2}$)	**71.33 ± 1.84**	**-3.60 ± 1.08**
CLUTaB (A_t)	**72.90 ± 0.67**	**-5.79 ± 0.95**

TABLE VI
ACCURACY OF DIFFERENT METHODS WITH AND WITHOUT L_{OOD}

Methods	ER	LwF	iCaRL	BiC	SS-IL
Acc. w. L_{OOD}	35.03	34.77	**45.80**	42.02	**44.70**
Acc. w/o. L_{OOD}	**35.72**	**35.78**	45.78	**45.90**	44.38

TABLE VII
$A_{c,2}$ WITH DIFFERENT TEST BATCH SIZES

Batch Size	1	2	4	8	32	64	A_t	$A_{c,1}$
$A_{c,2}$	42.20	53.24	66.16	75.54	81.28	81.86	82.12	46.90

E. Fine-Grained Continual Learning

To demonstrate how CLUTaB helps understand collected data, we design the fine-grained continual learning experiment. We split CIFAR-100 into five tasks, and each task contains 20 classes. Then, we split each task into two continua. We reorder the continua as *Reappearing Split* in Section VI-A. Different from the task-reappear setting in the main results, we leverage the superclass in CIFAR-100 and adopt a fixed class order. The details of superclass and class order are given in Appendix A. Now the class label y represents a superclass and the unknown task identifier t encodes a fine-grained subclass. Since the class order is deliberately designed, distributions between tasks are more similar, and it is more difficult to identify distribution shifts.

Table V shows the results for fine-grained continual learning. Like the task-reappear settings, we report $A_{c,1}$, $A_{c,2}$, and A_t for CLUTaB, and A_d for all methods. We observe higher performance on A_d for CLUTaB as expected. That means our continual OOD detection and InD detection still work well during training to find better timings for knowledge consolidation and recapture the forgotten knowledge when distribution changes mildly. Compared to the results of the $T = 5$, $K = 10$, task-reappear setting in Table IV, CLUTaB performs similarly on A_t but slightly worse on $A_{c,2}$, indicating that the accuracy of InD detection in two-step inference with a small batch size decreases due to the similarity among the distributions. Nevertheless, in Table V, $A_{c,2}$ of CLUTaB still surpasses $A_{c,1}$ by a large margin and is close to its upper bound A_t, which means InD detection in two-step inference predicts task identifiers with high accuracy. From these results, we conclude that CLUTaB is also effective for continual learning settings where distribution does not change dramatically and can design more fine-grained labels for data.

TABLE VIII
SUPERCLASS, SUBCLASS, AND CLASS ORDER FOR FINE-GRAINED CONTINUAL LEARNING ON CIFAR-100

Superclass	**Subclass**				
	Task 1	Task 2	Task 3	Task 4	Task 5
aquatic mammals	beaver	dolphin	otter	seal	whale
fish	aquarium fish	flatfish	ray	shark	trout
flowers	orchids	poppies	roses	sunflowers	tulips
food containers	bottles	bowls	cans	cups	plates
fruit and vegetables	apples	mushrooms	oranges	pears	sweet peppers
household electrical devices	clock	computer keyboard	lamp	telephone	television
household furniture	bed	chair	couch	table	wardrobe
insects	bee	beetle	butterfly	caterpillar	cockroach
large carnivores	bear	leopard	lion	tiger	wolf
large man-made outdoor things	bridge	castle	house	road	skyscraper
large natural outdoor scenes	cloud	forest	mountain	plain	sea
large omnivores and herbivores	camel	cattle	chimpanzee	elephant	kangaroo
medium-sized mammals	fox	porcupine	possum	raccoon	skunk
non-insect invertebrates	crab	lobster	snail	spider	worm
people	baby	boy	girl	man	woman
reptiles	crocodile	dinosaur	lizard	snake	turtle
small mammals	hamster	mouse	rabbit	shrew	squirrel
trees	maple	oak	palm	pine	willow
vehicles 1	bicycle	bus	motorcycle	pickup truck	train
vehicles 2	lawn-mower	rocket	streetcar	tank	tractor

TABLE IX
ACCURACY OF CLUTAB WITH TWO-STEP INFERENCE AND ICARL WITH THREE INFERENCE METHODS

T	5	10	20
CLUTaB($A_{c,2}$)	73.74	81.28	82.54
iCaRL(A_t)	**76.37**	**82.41**	**84.58**
iCaRL($A_{c,1}$)	51.72	45.77	39.35
iCaRL($A_{c,2}$)	53.26	41.99	23.76

TABLE X
ACCURACY OF TASK PREDICTION IN TESTING

Methods	Acc.
CCGN	72.61
CLUTaB (Batch Size = 1)	58.02
CLUTaB (Batch Size = 2)	71.02
CLUTaB (Batch Size = 4)	85.20
CLUTaB (Batch Size = 8)	93.03
CLUTaB (Batch Size = 16)	97.71
CLUTaB (Batch Size = 32)	98.37
CLUTaB (Batch Size = 64)	**99.68**

TABLE XI
CONTINUAL OOD DETECTION ACCURACY WITH DIFFERENT δ_{OOD}

δ_{OOD}	0.01	0.03	0.05	0.07	0.09	0.11	0.13
Acc.	78.72	91.35	95.83	**98.10**	96.76	92.38	84.03

TABLE XII
CONTINUAL IND DETECTION ACCURACY WITH DIFFERENT δ_{InD}

δ_{InD}	0.06	0.08	0.10	0.12	0.14	0.16	0.18
Acc.	91.67	97.22	**100.00**	**100.00**	**100.00**	94.45	77.78

TABLE XIII
CONTINUAL OOD DETECTION ACCURACY WITH DIFFERENT N_{InD}

N_{InD}	1	4	8	16	32	64	128
Acc.	62.29	74.47	80.65	94.33	97.49	98.10	**98.33**

F. Analysis

In this section, we perform more detailed analyses on continual OOD and InD detection, including the effects of L_{OOD}, $A_{c,2}$ with different test batch sizes. All experiments are performed on CIFAR-100.

1) Effect of L_{OOD} on Probability Distribution: Fig. 3 illustrates the distributions of maximum probabilities $p_{t,\max}(X_t^{\text{InD}})$ and $p_{t,\max}(X_t^{\text{OOD}})$ after learning first five continua in the reappearing $T = 5$, $K = 10$ setting by CLUTaB. X_t^{InD} is a batch of samples from task t in the rest five continua and X_t^{OOD} is from the left four tasks. We use $m = 2$ and $\lambda = 2$ in L_{OOD}. We observe that the gap between the max probabilities of InD and OOD data widens when L_{OOD} is applied. Compared between the distributions on different tasks, the optimal boundaries of InD and OOD data are different, which indicates the importance of the adaptive detection threshold.

2) Effect of L_{OOD} on Learning: We add L_{OOD} into the original loss functions in continual learning approaches. Table VI shows the accuracy of different methods with and without L_{OOD} in a standard ten-task continual learning setting. Here, $m = 1.5$ and $\lambda = 0.5$. L_{OOD} leads to a minor degradation for ER and LwF, and a minor improvement for SS-IL. However, BiC suffers almost a 4% reduction in accuracy. We consider that L_{OOD} is contradictory to BiC since BiC only flattens the probability distribution on the new task, while L_{OOD} does the same for all old tasks.

TABLE XIV
CONTINUAL OOD DETECTION ACCURACY WITH DIFFERENT O

O	500	1000	1500	2000	2500
Acc.	93.92	96.86	97.57	98.10	**98.12**

TABLE XV
ACCURACY WITH DIFFERENT m IN L_{OOD}

m	0	0.5	1	1.5	2	2.5	3
$A_{c,1}$	48.74	48.91	49.23	49.63	**49.84**	48.70	45.21
$A_{c,2}$	68.25	68.74	70.52	72.30	**72.46**	72.07	66.95
A_t	**73.63**	73.37	73.18	73.03	73.03	72.94	67.94

3) Size of Test Batch: Table VII shows $A_{c,2}$ of CLUTaB with the test batch size b in the aligned setting where $T = K = 10$. With more data in a batch, $A_{c,2}$ increases because the InD detection estimates more accurate $p_{t,\max}$. $A_{c,2}$ is close to its upper bound A_t when $b \geq 32$. In addition, $A_{c,1}$ is surpassed by $A_{c,2}$ when $b \geq 2$, which means we could adopt two-step inference even when the distribution is changing fast.

VII. CONCLUSION

In this article, we investigate CoIL, a new continual learning scenario with unknown task boundaries. The incremental unit in CoIL is a data continuum instead of a task, and each continuum may contain several tasks or only a subset of one certain task. The key idea to solving a CoIL problem is to identify task boundaries and predict task identifiers. First, we consider a simpler situation where all data from the same task are consecutive, so a task boundary means that a new task arrives. To identify task boundaries, we design a continual OOD detection method based on softmax probabilities. Once enough samples in a consecutive sequence are detected as OOD samples, we assume that a task boundary is identified. The continual OOD detection method calculates the local probabilities with the logits of the latest task to detect OOD samples on the well-trained part of the whole network. We also design an adaptive threshold for continual OOD detection, which can accommodate OOD detection for different tasks. Then, we further investigate the task-reappear setting, where a continuum may contain extra data from a learned task. We propose CLUTaB for the more challenging setting. To determine whether a set of data is sampled from any learned distribution, CLUTaB uses a continual InD detection method, which is based on local probabilities and leverages the output of task boundary identification. Also, an OOD loss is employed in CLUTaB to widen the gap of the max probabilities between InD and OOD samples. In addition, CLUTaB enables two-step inference, which first predicts task identifiers and then predicts class labels based on the predicted identifiers. Our approach outperforms task-based approaches in CoIL on CIFAR-100 and mini-ImageNet because of the proposed methods, and CLUTaB with the two-step inference significantly improves the performance because of its high InD detection accuracy.

Future work will focus on developing solutions to more general continual learning scenarios with unnotified distribution shifts, such as a scenario where the basic unit of the distribution shift is a class instead of a task. This will be achieved by adapting classwise anomaly detection or novelty detection to continual learning. We will also focus on accurate samplewise OOD detection for continual learning, which will allow our approach to solve continual learning problems with blurry task boundaries. In addition, it is worth studying how to train a model continually with repeated tasks [48], [49] after the task identifier prediction. A proper strategy instead of finetuning could further improve the performance.

APPENDIX A
MORE DETAILS OF FINE-GRAINED CONTINUAL LEARNING EXPERIMENTS

The details of superclass and class order are given in Table VIII. As shown in Table VIII, there are 100 classes in CIFAR-100, and these classes can be grouped into 20 superclasses. A superclass exactly contains five classes. Each image comes with a "fine" label (the class to which it belongs) and a "coarse" label (the superclass to which it belongs). As a result, we split CIFAR-100 into five tasks, and each task contains 20 classes. Different from the task-reappear setting in the main results, we adopt a fixed class order. In each task, there is exactly one class from each superclass. For example, in Table VIII, Task 1 contains classes from "beaver" to "lawn-mower." Besides, images with the same superclass have the same class label y. For example, in Table VIII, images of aquatic mammals (including "beaver," "dolphin," "otter," "seal," and "whale") have the same class label. Hence, the class label y now represents a superclass and the unknown task identifier t encodes a fine-grained subclass.

APPENDIX B
MORE ANALYSIS AND ABLATION STUDY

In this section, we perform detailed analyses on continual OOD and InD detection as well as more experiments to evaluate the effects of hyperparameters. All experiments are performed on CIFAR-100.

1) Two-Step Inference and Multihead Inference: We compare CLUTaB with two-step inference and iCaRL with three different inference methods in the aligned tasks and continua settings. Note that task identifiers are provided by a task oracle for iCaRL with multihead inference during both training and testing. Table IX shows the accuracy with a different number of tasks and continua. We find that iCaRL with multihead inference(A_t) outperforms CLUTaB and iCaRL with other inference methods in all experiments, which is reasonable since CLUTaB and iCaRL with other inference methods have to infer the task identifiers. Compared with iCaRL with one-step single-head inference ($A_{c,1}$), iCaRL with two-step inference ($A_{c,2}$) performs better when the total number of tasks is small but degenerates with more tasks. The results show that our methods help the classifier infers task identifiers more accurately.

2) Distribution Shift Detection: We compare our continual OOD detection with a change-detection test (CDT) method [50] that detects concept drift [51] on CNN, which

can be directly adopted to detect image distribution shifts. The CDT method monitors the classification error over time by a cumulative sum test to detect variations in its probability distribution. In the consecutive split setting where $T = 5$, $K = 10$, the detection accuracy of the CDT method is 98.35%, and the accuracy of our continual OOD detection is 98.10%. Our unsupervised continual OOD detection performs basically as well as CDT, even though CDT relies on class labels to calculate the classification error, while continual OOD detection does not need class labels as supervised signals.

3) Task Prediction in Testing: We compare our task prediction step in two-step inference with CCGN [52], which adds an extra classifier for task prediction in class-incremental learning. Table X shows the task prediction accuracy of CCGN and CLUTaB with different sizes of test batch where $T = K = 5$. Note that CCGN runs in the class-incremental setting, while CLUTaB still works in CoIL. Although CCGN outperforms CLUTaB when test batch sizes are small, CLUTab surpasses CCGN after the batch size is greater than 4 without extra model parameters.

4) Effect of δ_{OOD}: To investigate the effects of different δ_{OOD} in (4), we perform experiments in the consecutive setting where $T = 10$ and $K = 5$ with SS-IL w. OOD detection. We report the overall accuracy of the continual OOD detection in Table XI. With a proper δ_{OOD}, our model can identify OOD samples with high accuracy. Also, the continual OOD detection still works well in a range of values of δ_{OOD}, indicating that the continual OOD detection is not sensitive to δ_{OOD}.

5) Effect of δ_{InD}: To investigate the effects of different δ_{InD} in (5), we perform experiments in the task-reappear setting where $T = 10$ and $K = 20$. We report the overall accuracy of the continual InD detection in Table XII. Since we use all detected OOD samples for the continual InD detection and the OOD loss widens the gap between the max probabilities of InD and OOD data, our model can assign the correct task identifiers to all samples during training.

6) Effect of N_{InD}: To compare the number of stored InD samples N_{InD} in Algorithm 1, we perform experiments in the consecutive setting where $T = 10$ and $K = 5$ with SS-IL w. OOD detection. With more stored InD samples, the performance of the continual OOD detection improves. The results in Table XIII indicate that it is important to store enough InD samples to calibrate the detection threshold.

7) Effect of O: To compare the OOD buffer capacity O in Algorithm 1, we perform experiments in the consecutive setting where $T = 10$ and $K = 5$ with SS-IL w. OOD detection. The results are shown in Table XIV. The performance of the continual OOD detection improves with an increasing number of the OOD buffer capacity. For both efficiency and sufficiently exploring the input distribution to detect OOD samples with better performance, we choose $O = 2000$ as the OOD buffer capacity.

8) Effect of m: In the task-reappear setting where $T = 5$ and $K = 10$, Table XV shows the results with different entropy thresholds m in (6). We can find that proper values of m can improve $A_{c,1}$ and the accuracy of InD detection at test time. With larger m, A_t tends to get lower, which is reasonable since the OOD loss sacrifices the accuracy within one task for the better performance of distinguishing different tasks.

REFERENCES

[1] R. Ratcliff, "Connectionist models of recognition memory: Constraints imposed by learning and forgetting functions," *Psychol. Rev.*, vol. 97, no. 2, p. 285, 1990.

[2] I. J. Goodfellow, M. Mirza, D. Xiao, A. Courville, and Y. Bengio, "An empirical investigation of catastrophic forgetting in gradient-based neural networks," 2013, *arXiv:1312.6211*.

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

[4] S. Jung, H. Ahn, S. Cha, and T. Moon, "Continual learning with node-importance based adaptive group sparse regularization," in *Proc. Adv. Neural Inf. Process. Syst.*, 2020, pp. 3647–3658.

[5] A. Robins, "Catastrophic forgetting, rehearsal and Pseudorehearsal," *Connect. Sci.*, vol. 7, no. 2, pp. 123–146, 1995.

[6] Y. Wu et al., "Large scale incremental learning," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 374–382.

[7] S. Yan, J. Xie, and X. He, "DER: Dynamically expandable representation for class incremental learning," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2021, pp. 3014–3023.

[8] O. Ostapenko, P. Rodriguez, M. Caccia, and L. Charlin, "Continual learning via local module composition," 2021, *arXiv:2111.07736*.

[9] G. M. van de Ven and A. S. Tolias, "Three scenarios for continual learning," 2019, *arXiv:1904.07734*.

[10] D. Hendrycks and K. Gimpel, "A baseline for detecting misclassified and out-of-distribution examples in neural networks," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2017.

[11] D. Hendrycks, M. Mazeika, and T. G. Dietterich, "Deep anomaly detection with outlier exposure," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2019.

[12] R. Aljundi, K. Kelchtermans, and T. Tuytelaars, "Task-free continual learning," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 11246–11255.

[13] Y.-C. Hsu, Y.-C. Liu, A. Ramasamy, and Z. Kira, "Re-evaluating continual learning scenarios: A categorization and case for strong baselines," 2018, *arXiv:1810.12488*.

[14] 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.

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

[16] A. Chaudhry, P. K. Dokania, T. Ajanthan, and P. H. S. Torr, "Riemannian walk for incremental learning: Understanding forgetting and intransigence," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2018, pp. 556–572.

[17] 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, Oct. 2023.

[18] Z. Zhang and L. Zhang, "NeCa: Network calibration for class incremental learning," in *Proc. Asian Conf. Pattern Recognition*, 2023, pp. 385–399.

[19] J. Kim and D. Choi, "Split-and-bridge: Adaptable class incremental learning within a single neural network," in *Proc. 35th AAAI Conf. Artif. Intell.*, 2021, pp. 8137–8145.

[20] H. Ahn, J. Kwak, S. Lim, H. Bang, H. Kim, and T. Moon, "SS-IL: Separated softmax for incremental learning," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2021, pp. 844–853.

[21] 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.

[22] Y. Jian, J. Yi, and L. Zhang, "Adaptive feature generation for online continual learning from imbalanced data," in *Proc. Adv. Knowl. Discovery Data Mining*, 2022, pp. 276–289.

[23] J. Yoon, E. Yang, J. Lee, and S. J. Hwang, "Lifelong learning with dynamically expandable networks," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2018.

[24] J. Rajasegaran, M. Hayat, S. H. Khan, F. S. Khan, and L. Shao, "Random path selection for continual learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019, pp. 12669–12679.

[25] S. Ho, M. Liu, L. Du, L. Gao, and Y. Xiang, "Prototype-guided memory replay for continual learning," *IEEE Trans. Neural Netw. Learn. Syst.*, early access, 2023. [Online]. Available: https://ieeexplore.ieee.org/document/10058177

[26] Q. Gao, Z. Luo, D. Klabjan, and F. Zhang, "Efficient architecture search for continual learning," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 34, no. 11, pp. 8555–8565, Aug. 2023.

[27] Y. Zhao, D. Saxena, and J. Cao, "AdaptCL: Adaptive continual learning for tackling heterogeneity in sequential datasets," *IEEE Trans. Neural Netw. Learn. Syst.*, early access, 2023. [Online]. Available: https://ieeexplore.ieee.org/document/10365578

[28] 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.

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

[30] R. Aljundi, M. Lin, B. Goujaud, and Y. Bengio, "Gradient based sample selection for online continual learning," in *Proc. Int. Conf. Adv. Neural Inf. Process. Syst.*, 2019, pp. 11816–11825.

[31] R. Aljundi et al., "Online continual learning with maximal interfered retrieval," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019, pp. 11849–11860.

[32] D. Rao, F. Visin, A. A. Rusu, R. Pascanu, Y. W. Teh, and R. Hadsell, "Continual unsupervised representation learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019, pp. 7645–7655.

[33] S. Lee, J. Ha, D. Zhang, and G. Kim, "A neural Dirichlet process mixture model for task-free continual learning," *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2020.

[34] M. D. Lange and T. Tuytelaars, "Continual prototype evolution: Learning online from non-stationary data streams," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2021, pp. 8250–8259.

[35] X. Jin, A. Sadhu, J. Du, and X. Ren, "Gradient-based editing of memory examples for online task-free continual learning," 2020, *arXiv:2006.15294*.

[36] S. Liang, Y. Li, and R. Srikant, "Enhancing the reliability of out-of-distribution image detection in neural networks," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2018.

[37] W. Liu, X. Wang, J. D. Owens, and Y. Li, "Energy-based out-of-distribution detection," in *Proc. Adv. Neural Inf. Process. Syst.*, 2020, pp. 21464–21475.

[38] B. Zong et al., "Deep autoencoding Gaussian mixture model for unsupervised anomaly detection," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2018.

[39] K. Lee et al., "A simple unified framework for detecting out-of-distribution samples and adversarial attacks," in *Proc. NeurIPS*, Dec. 2018, pp. 7167–7177.

[40] C. S. Sastry and S. Oore, "Detecting out-of-distribution examples with Gram matrices," in *Proc. 37th Int. Conf. Mach. Learn.*, Jul. 2020, pp. 8491–8501.

[41] Q. Yu and K. Aizawa, "Unsupervised out-of-distribution detection by maximum classifier discrepancy," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 9517–9525.

[42] S. Fort, J. Ren, and B. Lakshminarayanan, "Exploring the limits of out-of-distribution detection," 2021, *arXiv:2106.03004*.

[43] Z. Wang, L. Shen, L. Fang, Q. Suo, T. Duan, and M. Gao, "Improving task-free continual learning by distributionally robust memory evolution," in *Proc. 39th Int. Conf. Mach. Learn. (ICML)*, 2022, pp. 22985–22998.

[44] J. Rajasegaran, S. H. Khan, M. Hayat, F. S. Khan, and M. Shah, "iTAML: An incremental task-agnostic meta-learning approach," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Aug. 2020, pp. 13585–13594.

[45] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," 2009.

[46] O. Vinyals, C. Blundell, T. Lillicrap, and D. Wierstra, "Matching networks for one shot learning," in *Proc. Adv. Neural Inf. Process. Syst. (NIPS)*, 2016, pp. 3630–3638.

[47] 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.

[48] A. Cossu et al., "Is class-incremental enough for continual learning?" *Frontiers Artif. Intell.*, vol. 5, Mar. 2022, Art. no. 829842.

[49] T. Lesort et al., "Challenging common assumptions about catastrophic forgetting," 2022, *arXiv:2207.04543*.

[50] S. Disabato and M. Roveri, "Learning convolutional neural networks in presence of concept drift," in *Proc. Int. Joint Conf. Neural Netw.*, Budapest, Hungary, 2019, pp. 1–8.

[51] J. Lu, A. Liu, F. Dong, F. Gu, J. Gama, and G. Zhang, "Learning under concept drift: A review," *IEEE Trans. Knowl. Data Eng.*, vol. 31, no. 12, pp. 2346–2363, Dec. 2018.

[52] 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.*, Sep. 2020, pp. 3931–3940.

Xiaoxie Zhu received the B.E. degree from the Software College, Jilin University, Changchun, China, in 2020, and the M.S. degree from the Department of Computer Science and Technology, Nanjing University, Nanjing, China, in 2023.

His current research interests include continual learning and incremental learning in nonstationary and open environments.

Jinfeng Yi (Member, IEEE) received the B.E. degree from the University of Science and Technology of China, Hefei, China, in 2009, and the Ph.D. degree from Michigan State University, East Lansing, MI, USA, in 2014.

He is currently the Vice-President with 4Paradigm Technology, Beijing, China. In addition, he is an Adjunct Professor with Fudan University, Shanghai, China, Renmin University of China, Beijing, China, and the University of Science and Technology of China. His research interests primarily lie in machine learning, data mining, and trustworthy artificial intelligence (AI).

Dr. Yi has served as an Area Chair for NeurIPS from 2021 to 2023 and ICML from 2023 to 2024. He was recognized as one of the MIT Technology Review's "Innovators Under 35 in China" and received the Management Science Innovation Award.

Lijun Zhang (Senior Member, IEEE) received the B.E. and Ph.D. degrees in software engineering and computer science from Zhejiang University, Hangzhou, China, in 2007 and 2012, respectively.

He was a Post-Doctoral Researcher with the Department of Computer Science and Engineering, Michigan State University, East Lansing, MI, USA. He is currently a Professor with the School of Artificial Intelligence (AI), Nanjing University, Nanjing, China. His research interests include machine learning and optimization.