Artificial Intelligence 338 (2025) 104237

Contents lists available at ScienceDirect

Artificial Intelligence

journal homepage: www.elsevier.com/locate/artint

# Open-world continual learning: Unifying novelty detection and continual learning

Gyuhak Kim [a,1], Changnan Xiao [b,1], Tatsuya Konishi [a,2], Zixuan Ke [a], Bing Liu [a,*]

[a] University of Illinois Chicago, 851 S Morgan St, Chicago, IL, 60607, United States
[b] ByteDance, Building 24, Zone B, 1999 Yishan Road, Shanghai, 201100, China

ARTICLE INFO

*Keywords:*
Open world learning
Continual learning
OOD detection

ABSTRACT

As AI agents are increasingly used in the real open world with unknowns or novelties, they need the ability to (1) recognize objects that (a) they have learned before and (b) detect items that they have never seen or learned, and (2) learn the new items incrementally to become more and more knowledgeable and powerful. (1) is called *novelty detection* or *out-of-distribution* (OOD) *detection* and (2) is called *class incremental learning* (CIL), which is a setting of *continual learning* (CL). In existing research, OOD detection and CIL are regarded as two completely different problems. This paper first provides a theoretical proof that good OOD detection for each task within the set of learned tasks (called *closed-world OOD detection*) is *necessary* for successful CIL. We show this by decomposing CIL into two sub-problems: *within-task prediction* (WP) and *task-id prediction* (TP), and proving that TP is correlated with closed-world OOD detection. The *key theoretical result* is that regardless of whether WP and OOD detection (or TP) are defined explicitly or implicitly by a CIL algorithm, good WP and good closed-world OOD detection are *necessary* and *sufficient* conditions for good CIL, which unifies novelty or OOD detection and continual learning (CIL, in particular). We call this traditional CIL the *closed-world CIL* as it does not detect future OOD data in the open world. The paper then proves that the theory can be generalized or extended to *open-world CIL*, which is the proposed *open-world continual learning*, that can perform CIL in the open world and detect future or open-world OOD data. Based on the theoretical results, new CIL methods are also designed, which outperform strong baselines in CIL accuracy and in continual OOD detection by a large margin.

## 1. Introduction

The current dominant machine learning paradigm (ML) makes the *closed-world assumption*, which means that the classes of objects seen by the system in testing or deployment must have been seen during training [60,59,5,14,39], i.e., there is nothing novel occurring during testing or deployment. This assumption is invalid in practice as the real environment is an **open world** that is full of unknowns or novel objects. To make an AI agent thrive in the open world, it has to detect novelties and learn them incrementally to make the

* Corresponding author.
*E-mail addresses:* gkim87@uic.edu (G. Kim), changnanxiao@gmail.com (C. Xiao), tt-konishi@kddi.com (T. Konishi), zke4@uic.edu (Z. Ke), liub@uic.edu (B. Liu).

[1] Equal contribution.

[2] The work was done when this author was visiting Bing Liu's group at the University of Illinois Chicago. The original affiliation and address of this author are *KDDI Research, Inc., 2-1-15 Ohara, Fujimino-shi, Saitama, 356-8502, Japan.*

https://doi.org/10.1016/j.artint.2024.104237
Received 1 February 2023; Received in revised form 14 October 2024; Accepted 18 October 2024

Available online 31 October 2024
0004-3702/© 2024 The Author(s). Published by Elsevier B.V. This is an open access article under the CC BY license (http://creativecommons.org/licenses/by/4.0/).

system more knowledgeable and adaptable over time. This process involves multiple activities, such as *novelty/OOD detection*, *novelty characterization*, *adaption*, *risk assessment*, and *continual learning* of the detected novel items or objects [40,41]. Novelty detection, also called *out-of-distribution* (OOD) *detection*, aims to detect unseen objects that the agent has not learned. On detecting novel objects or situations, the agent has to respond or adapt its actions. But in order to adapt, it must first *characterize* the novel object as without it, the agent would not know how to respond or adapt. For example, it may characterize a detected novel object as looking like a dog. Then, the agent may react like it would react to a dog. In the process, the agent also constantly assesses the risk of its actions. Finally, it also learns to recognize the new object incrementally so that it will not be surprised when it sees the same kind of object in the future. This incremental learning is called *continual learning* (CL) or *lifelong learning* [13,27]. Note that before learning, the agent must obtain labeled training data, which can be collected by the agent through interaction with the environment or human users. This aspect is out of the scope of this paper. See [40,41] for details.

This paper focuses only on the key learning aspects of the open world scenario: (1) OOD/novelty detection and (2) continual learning, more specifically *class incremental learning* (CIL) (see the definition below). In the research community, (1) and (2) are regarded as two completely different problems, but this paper theoretically unifies them by proving that good OOD detection for each task within the set of learned tasks, which we call *closed-world OOD detection*, is in fact *necessary* for CIL. Below, we define the concepts of OOD detection and continual learning.

***Out-of-distribution (OOD) detection***: Given the training data $\mathcal{D} = \{(x^i, y^i)_{i=1}^n\}$, where $n$ is the number of data samples, and $x^i \in \mathbf{X}$ is an input sample and $y^i \in \mathbf{Y}$ (the set of all class labels in $\mathcal{D}$) is its class label, our goal is to build a classifier $f : \mathbf{X} \rightarrow \mathbf{Y} \cup \{O\}$ that can detect test instances that do not belong to any classes in $\mathbf{Y}$ (called *OOD detection*), which are assigned to the class $O$. $\mathbf{Y}$ is often called the *in-distribution* (IND) *classes*.

We also call this *open-world OOD detection*. As we can see from the definition, an OOD detection algorithm can also classify test instances belonging to $\mathbf{Y}$ to their respective classes, which is called IND classification, although most OOD detection papers do not report the IND classification results.

Continual learning (CL) aims to incrementally learn a sequence of tasks. Each task consists of a set of classes to be learned together (the set may contain only a single class). Once a task is learned, its training data (at least a majority of it) is no longer accessible. Thus, unlike multitask learning, in learning a new task, CL will not be able to use the data of the previous tasks. A major challenge of CL is *catastrophic forgetting* (CF), which refers to the phenomenon that in learning a new task, the neural network model parameters need to be modified, which may corrupt the knowledge learned for previous tasks in the network and cause performance degradation for the previous tasks [47]. Although many CL techniques have been proposed, they are mainly empirical. Limited theoretical work has been done on how to solve CL. This paper performs such a theoretical study about the necessary and sufficient conditions for effective CL. Two main CL settings have been extensively studied: *class incremental learning* (CIL) and *task incremental learning* (TIL) [68]. In CIL, the learning process builds a single classifier for all tasks/classes learned so far. In testing, a test instance from any class may be presented for the model to classify. No prior task information (e.g., task-id) of the test instance is provided. Formally, CIL is defined as follows.

***Class incremental learning*** (CIL). CIL learns a sequence of tasks, $1, 2, \cdots$. Let $T$ be the number of tasks that have been learned so far. Each task $k$ ($1 \leq k \leq T$) has a training dataset $\mathcal{D}_k = \{(x_k^i, y_k^i)_{i=1}^{n_k}\}$, where $n_k$ is the number of data samples in task $k$, and $x_k^i \in \mathbf{X}$ is an input sample and $y_k^i \in \mathbf{Y}_k$ (the set of all classes of task $k$) is its class label. All $\mathbf{Y}_k$'s are disjoint ($\mathbf{Y}_k \cap \mathbf{Y}_{k'} = \emptyset, \forall k \neq k'$) and $\bigcup_{k=1}^T \mathbf{Y}_k = \mathbf{Y}$. The goal of CIL is to construct a single predictive function or classifier $f : \mathbf{X} \rightarrow \mathbf{Y}$ that can identify the class label $y$ of each given test instance $x$ from the $T$ tasks.

Based on CIL, we can also define the term *close-world OOD detection*.

***Closed-world OOD detection***: *Closed-world OOD detection* for a given task $k$ among the $T$ tasks that have been learned so far is *OOD detection* regarding the classes of task $k$ as the IND classes and those of the other $T - 1$ tasks as the OOD classes.

From now on when we refer to *OOD detection* on its own (which is *open-world OOD detection*), we mean it is not limited to the $T$ learned tasks, as opposed to the *closed-world OOD detection*. Clearly, (open-world) OOD detection encompasses closed-world OOD detection, but not vice versa.

Unlike CIL, each task in TIL is a separate or independent classification problem. For example, one task could be to classify different breeds of dogs and another task could be to classify different types of animals (the tasks may not be disjoint). One model is built for each task in a shared network. In testing, the task-id of each test instance is provided and the system uses only the specific model for the task (dog or animal classification) to classify the test instance. Formally, TIL is defined as follows.

***Task incremental learning*** (TIL). TIL learns a sequence of tasks, $1, 2, \cdots$. Let $T$ be the number of tasks that have been learned so far. Each task $k$ ($1 \leq k \leq T$) has a training dataset $\mathcal{D}_k = \{((x_k^i, k), y_k^i)_{i=1}^{n_k}\}$, where $n_k$ is the number of data samples in task $k \in \mathbf{T} = \{1, 2, ..., T\}$, and $x_k^i \in \mathbf{X}$ is an input sample and $y_k^i \in \mathbf{Y}_k \subset \mathbf{Y}$ is its class label. The goal of TIL is to construct a predictor $f : \mathbf{X} \times \mathbf{T} \rightarrow \mathbf{Y}$ to identify the class label $y \in \mathbf{Y}_k$ for $(x, k)$ (the given test instance $x$ from task $k$).

This paper focuses on CIL, which involves incrementally learning new or novel object classes—a key aspect of open-world learning. While the proposed methods are also applicable to TIL, we do not address it in this paper. TIL is generally simpler, and several existing techniques can achieve it without CF [61,69]. In contrast, CIL remains highly challenging due to the difficulty of Inter-task Class

Separation (ICS), i.e., establishing decision boundaries between classes from the new task and those from previous tasks in learning the new task without accessing the training data of previous tasks.

**Problem Statement** (*open-world continual learning*): *Open-world continual learning* (OWCL) is defined as CIL with the OOD detection capability. We also call it *open-world CIL* or CIL$^+$. At any time, the resulting open-world CIL model can classify test instances belonging to the classes in the $T$ tasks that have been learned so far to their respective classes and also detect OOD instances that do not belong to any of the learned classes so far.

Note that OOD detection in CIL$^+$ is different from traditional OOD detection (which sees the full IND data together) because, in CIL$^+$, the model does not see all the IND data together. Instead, the IND data comes in a sequence of tasks incrementally, and in learning each task, the model does not see any data (or only a very small sample) of the old or previous tasks.

**Main contributions:** This paper makes three main contributions. First, it theoretically proves the necessary and sufficient conditions for solving the CIL problem. A good closed-world OOD detection performance is one of the necessary conditions, which connects or unifies OOD detection and CIL. Since in this traditional CIL, the test instances are assumed to be from one of the $T$ tasks that have been learned, we call the existing CIL the *closed-world CIL*. Second, we prove that the theory can naturally be generalized or extended to the *open-world CIL*, which is the proposed *open-world continual learning*. Open-world CIL can perform CIL in the open world and detect OOD test data that do not belong to any of the $T$ tasks learned so far. Third, based on the theory, several new CIL algorithms are designed, which are also able to detect novel (or OOD) instances for the open-world continual learning (OWCL) setting. Note that from here onward, when we do not explicitly say open-world CIL, CIL means the traditional CIL.

**Theory.** We conduct a theoretical study of CIL, which is applicable to any CIL classification model. Instead of focusing on the traditional PAC generalization bound [53] or neural tangent kernel (NTK) [26], we focus on how to solve the CIL problem. We first decompose the CIL problem into two sub-problems in a probabilistic framework: *Within-task Prediction* (WP) and *Task-id Prediction* (TP). WP means that the prediction for a test instance is only made within the classes of the task to which the test instance belongs, which is basically the TIL problem. TP predicts the task-id. TP is needed because, in CIL, task-id is not provided at test time. This paper then proves based on cross-entropy loss that (i) the CIL performance is bounded by WP and TP performances, and (ii) TP and closed-world OOD detection performance bound each other. This paper further generalizes the result to open-world CIL (or CIL$^+$). These results unify CIL and OOD detection.

***Key theoretical results***: Regardless of whether WP and TP or OOD detection are defined explicitly or implicitly by a closed-world or open-world CIL algorithm, (1) good WP and good TP or closed-world OOD detection are *necessary* and *sufficient* conditions for good closed-world CIL performances and (2) good WP and good TP or open-world OOD detection are *necessary* and *sufficient* conditions for good open-world CIL performances.[3]

The intuition of the theory is simple because if a closed-world or open-world CIL model is perfect at detecting OOD samples for each task, which solves the ICS problem, then closed-world or open-world CIL is reduced to WP, which is the traditional single-task supervised learning for each task. Note that many OOD detection algorithms can also perform IND classification, which is WP.

**New CIL Algorithms for OWCL.** The theoretical result provides principled guidance for solving the (closed-world or open-world) CIL problem. Based on the theory, several new CIL methods are designed. (1) The first few methods integrate a TIL method and an OOD detection method, which outperform strong baselines in both the CIL and TIL settings by a large margin. This combination is attractive because TIL has achieved no forgetting, and we only need a strong OOD detection technique that can perform both IND prediction and OOD detection to learn each task to achieve strong CIL results. We do not propose a new OOD detection method as there are numerous such methods in the literature. We use two existing ones. (2) Another method is based on a pre-trained model and an OOD replay technique, which performs even better, outperforming existing baselines markedly in both CIL and OOD detection in the OWCL setting.

## 2. Related work

Although a large number of algorithms have been proposed to solve the CIL problem, they are mainly empirical. Two papers have focused on studying the traditional PAC generalization bound [53] or NTK [26], but they do not tell how to solve the CL problem. This paper focuses on how to solve the CIL problem. To the best of our knowledge, we are not aware of any work that has proposed a theory on how to solve CIL. Also, none of the existing work has connected CIL and OOD detection. Our work shows that a good CIL algorithm can naturally perform OOD detection in the open world. Below, we first survey four popular families of CL approaches, which are mainly for overcoming catastrophic forgetting (CF). We then discuss related works about open-world learning.

**Regularization-based methods** prevent forgetting by restricting the learned parameters for previous tasks from being modified significantly by using a regularization term to penalize such changes [31,76] or to regularize the learned representations or outputs so that they are not far from those of the previously learned network [37,77].

**Replay-based methods** [56,10,7,6,72] mitigate forgetting by saving a small amount of training data from previous tasks in a memory buffer and jointly train the network using the current data and the previous task data saved in the memory. Some methods

[3] This result applies to both batch/offline and online/stream CIL and to CIL problems with blurry task boundaries which means that some training data of a task may come later together with a future task.

in this family also study which samples in the memory should be used in replaying [2] or which samples in the training data should be saved for later replaying [56,43].

**Generative methods** construct a generative network to generate raw training data [62,48,3]. The generated data are used with the current task training data to jointly train the classification network. [77] generates features instead of raw data. The generated samples in these methods are used to prevent forgetting in both the generative network and the classification network.

**Parameter-isolation methods** [61,69] train a set of task-specific parameters to effectively protect the important parameters of each task from being updated, which thus has almost no forgetting. A limitation of the approach is that the correct task-id of each test instance must be known to the system to select the corresponding task-specific parameters at inference. These methods are thus mainly used for *task incremental learning* (TIL). Some CIL methods also used these methods [1,49,54,22] and they have separate mechanisms to predict the task-id (more on this below). However, their CIL performances are far below that of recent replay-based counterparts (see Sections 4.2 and 5.2 for details). Two of our proposed CIL methods also use two parameter-isolation methods (HAT [61] and SupSup [69]) for TIL as one of the components.

Using a TIL method for CIL means that CIL is decomposed into WP and TP. Task-id prediction (TP) is the key challenge. For example, CCG [1] constructs an additional network to predict the task-id. iTAML [54] identifies the task-id of the test data in a batch. A serious limitation of this is that it requires the test data in a batch to belong to the same task. Our methods are different as they can predict for a single test instance at a time. HyperNet [49] and PR [22] propose an entropy-based task-id prediction method. SupSup [69] predicts task-id by finding the optimal superpositions at inference. However, these methods perform poorly because they either do not know that OOD detection is the key for accurate task-id prediction or their task models are not built for OOD detection. It is also important to note that our theory does not explicitly predict task-id. Instead, it uses the TP probability and WP probability for test prediction.

Several papers have explicitly or implicitly indicated the use of OOD detection for task-id prediction in continual learning. For example, the CIL method in [24] is based on one-class classification, which is OOD detection with only a single class as the in-distribution (IND) class. In [22], the authors proposed an uncertainty-based OOD detection framework for task-id prediction. Two specific methods were presented. One uses *entropy* to quantify the uncertainty (which has also been used in some other systems discussed above) and the other is called *agree*, which selects the task that leads to the highest agreement in predictions across task models. There are also related works that did not explicitly make a connection between CIL and OOD detection, their methods implicitly imply it. For example, [72] uses a regularization similar to OOD detection, which employs the replay data from previous tasks as OOD samples. [67] proposed to train a VAE model for each class to be learned. It then estimates the likelihood $p(x|y)$ and uses the Bayes rule to predict the class ($y$) of each test instance ($x$). Our work makes a theoretical contribution by formally connecting CIL and OOD detection and proving that for a good CIL performance, a good OOD detection capability for each task is *necessary*.

*Open world learning* has been studied by many researchers [60,59,5,14,39,41]. However, the existing research mainly focused on novelty detection, also called *open set recognition* or *out-of-distribution* (OOD) *detection*. Some researchers have also studied learning the novel objects after they are detected and manually labeled [5,14,71,25]. However, none of them perform continual learning, which has additional challenges of catastrophic forgetting (CF) and inter-task class separation (ICS). Several researchers also studied other related tasks in addition to novelty detection, e.g., characterization of novelties and adaptation of novelties to maximize the performance task [46,65]. Again, these works are not about continual learning. Excellent surveys of novelty detection or OOD detection and open-world learning can be found in [73,51,52,25]. [16] did novelty detection and also continual learning, but its continual learning uses the regularization-based method. It is quite weak because it has serious forgetting. A position paper [33] recently presented some nice blue-sky ideas about open-world learning, but it does not propose or implement any algorithm.

Our proposed algorithms are quite different. In training, based on our theory, we use two existing OOD detection methods to verify that our theory can guide us to design new and much more effective CIL algorithms. In testing, our OOD detection is in the open-world continual learning (OWCL) setting, which has been described in the introduction section.

Several researchers have studied *novel class discovery* [17], which is defined as discovering the hidden classes in the detected novel or OOD instances. Our work does not perform this function. We assume that the training data for each new task is given. Performing automatic class label discovery is still very challenging as in many cases, the class assignments can be subjective and are determined by human users. For example, for a dog, whether it should just be labeled as a dog or a specific breed of dog is a subjective decision and depends on applications.

Some existing works have combined OOD detection and continual learning [16,18,57]. These papers use OOD score thresholds to determine OOD instances and also do continual learning afterward. However, their continual learning still assumes that the training data are given as it is hard to do real-time detection and learning. This is because, without human involvement, it is impossible to obtain novel class labels in general and verify the correctness of OOD detection results. Any error in OOD detection will propagate to the continual learning phase. [40,41] reported a continual learning chatbot that can detect novel user utterances that the system does not understand and chat with the user through its novelty characterization mechanism to get the ground truth. However, this system is based on saving new/novel utterances and performing matching and retrieval in subsequent chatting. [46] proposed an integrated online architecture that combines and extends probabilistic programming and planning to (1) detect novelty, (2) incrementally characterize the novelty, and (3) continually adapt its task-based reasoning to the evolving understanding of the novelty to maximize task performance. However, this work is not about continual learning. [50] also reported a system for continuous emotion novelty detection.

**Table 1**
Acronyms used in the paper.

| | |
|---|---|
| CL | Continual learning |
| CIL | Class incremental learning |
| TIL | Task incremental learning |
| OOD | Out-of-distribution |
| IND | In-distribution |
| WP | Within-task prediction |
| TP | Task-id prediction |
| CF | Catastrophic forgetting |
| OWCL | Open-world continual learning |
| NTK | Neural tangent kernel |
| AUC | Area Under the ROC Curve |

## 3. A theoretical study on solving CIL

This section presents our theory for solving CIL, which also covers novelty or OOD detection. It first shows that the CIL performance improves if the within-task prediction (WP) performance and/or the task-id prediction (TP) performance improve, and then shows that TP and OOD detection bound each other, which indicates that CIL performance is controlled by WP and OOD detection. This connects CIL and OOD detection. After that, we study the necessary conditions for a good CIL model, which includes a good WP, and a good TP or OOD detection. In the first four sub-sections, we focus on the traditional CIL that is limited to the number of tasks $T$ that have been learned so far, which we also call *closed-world CIL*. OOD detection in this context is also within the $T$ learned tasks and is called *closed-world OOD detection* (see Section 1). For simplicity in presentation, we will not add *closed-world* before CIL or OOD detection below. In Section 3.5, we generalize/extend the theory to open-world CIL or open-world continual learning, which will also detect OOD data that do not belong to any of the $T$ tasks learned so far. Table 1 gives the list of acronyms used in the paper.

### 3.1. CIL problem decomposition

This sub-section first presents the assumptions made by CIL based on its definition and then proposes a decomposition of the CIL problem into two sub-problems. Assume that a CIL system has learned a sequence of $T$ tasks $\{(\mathbf{X}_k, \mathbf{Y}_k)\}_{k=1,\ldots,T}$ so far, where $\mathbf{X}_k$ is the domain of task $k$ and $\mathbf{Y}_k$ are classes of task $k$ as $\mathbf{Y}_k = \{\mathbf{Y}_{k,j}\}$, where $j$ indicates the $j$th class in task $k$. Let $\mathbf{X}_{k,j}$ to be the domain of $j$th class of task $k$, where $\mathbf{X}_k = \bigcup_j \mathbf{X}_{k,j}$. For accuracy, we will use $x \in \mathbf{X}_{k,j}$ instead of $\mathbf{Y}_{k,j}$ in probabilistic analysis. Based on the definition of class incremental learning (CIL) (Section 1), the following assumptions are implied,

**Assumption 1.** The domains of classes of the same task are disjoint, i.e., $\mathbf{X}_{k,j} \cap \mathbf{X}_{k,j'} = \emptyset, \forall j \neq j'$.

**Assumption 2.** The domains of tasks are disjoint, i.e., $\mathbf{X}_k \cap \mathbf{X}_{k'} = \emptyset, \forall k \neq k'$.

For any ground event $D$, the goal of a CIL problem is to learn $\mathbf{P}(x \in \mathbf{X}_{k,j} | D)$. This can be decomposed into two probabilities, *within-task IND prediction* (WP) probability and *task-id prediction* (TP) probability. WP probability is $\mathbf{P}(x \in \mathbf{X}_{k,j} | x \in \mathbf{X}_k, D)$ and TP probability is $\mathbf{P}(x \in \mathbf{X}_k | D)$. We can rewrite the CIL problem using WP and TP based on the two assumptions,

$$\mathbf{P}(x \in \mathbf{X}_{k_0,j_0} | D) = \sum_{k=1,\ldots,n} \mathbf{P}(x \in \mathbf{X}_{k,j_0} | x \in \mathbf{X}_k, D)\mathbf{P}(x \in \mathbf{X}_k | D) \tag{1}$$

$$= \mathbf{P}(x \in \mathbf{X}_{k_0,j_0} | x \in \mathbf{X}_{k_0}, D)\mathbf{P}(x \in \mathbf{X}_{k_0} | D) \tag{2}$$

where $k_0$ means a particular task and $j_0$ is a particular class in the task.

Some remarks are in order about Eq. (2) and our subsequent analysis to set the stage.

**Remark 1.** Eq. (2) shows that if we can improve either the WP or TP performance, or both, we can improve the CIL performance.

**Remark 2.** It is important to note that our theory is not concerned with the learning algorithm or training process. But we will propose some concrete CIL algorithms based on the theoretical result in the experiment section.

**Remark 3.** We note that the CIL definition and the subsequent analysis apply to tasks with any number of classes (including only one class per task) and to online CIL where the training data for each task or class comes gradually in a data stream and may also cross task boundaries (blurry tasks [4]) because our analysis is based on an already-built CIL model after training. Regarding blurry task boundaries, suppose dataset 1 has classes {dog, cat, tiger} and dataset 2 has classes {dog, computer, car}. We can define task 1 as {dog, cat, tiger} and task 2 as {computer, car}. The shared class *dog* in dataset 2 is just additional training data of *dog* appeared after task 1.

**Remark 4.** CIL = WP * TP in Eq. (2) means that when we have WP and TP (defined either explicitly or implicitly by implementation), we can find a corresponding CIL model defined by WP * TP. Similarly, when we have a CIL model, we can find the corresponding underlying WP and TP defined by their probabilistic definitions.

In the following sub-sections, we develop this further concretely to derive the sufficient and necessary conditions for solving the CIL problem in the context of cross-entropy loss as it is used in almost all supervised CIL systems.

### 3.2. CIL improves as WP and/or TP improve

As stated in Remark 2 above, the study here is based on a *trained CIL model* and not concerned with the algorithm used in training the model. We use cross-entropy as the performance measure of a trained model as it is the most popular loss function used in supervised CL. For experimental evaluation, we use *accuracy* following CL papers. Denote the cross-entropy of two probability distributions $p$ and $q$ as

$$H(p,q) \overset{def}{=} -\mathbb{E}_p[\log q] = -\sum_i p_i \log q_i. \tag{3}$$

For any $x \in \mathbf{X}$, let $y$ to be the CIL ground truth label of $x$, where $y_{k_0,j_0} = 1$ if $x \in \mathbf{X}_{k_0,j_0}$ otherwise $y_{k,j} = 0$, $\forall (k,j) \neq (k_0,j_0)$. Let $\tilde{y}$ be the WP ground truth label of $x$, where $\tilde{y}_{k_0,j_0} = 1$ if $x \in \mathbf{X}_{k_0,j_0}$ otherwise $\tilde{y}_{k_0,j} = 0$, $\forall j \neq j_0$. Let $\bar{y}$ be the TP ground truth label of $x$, where $\bar{y}_{k_0} = 1$ if $x \in \mathbf{X}_{k_0}$ otherwise $\bar{y}_k = 0$, $\forall k \neq k_0$. Denote

$$H_{WP}(x) = H(\tilde{y}, \{\mathbf{P}(x \in \mathbf{X}_{k_0,j} | x \in \mathbf{X}_{k_0}, D)\}_j), \tag{4}$$

$$H_{CIL}(x) = H(y, \{\mathbf{P}(x \in \mathbf{X}_{k,j} | D)\}_{k,j}), \tag{5}$$

$$H_{TP}(x) = H(\bar{y}, \{\mathbf{P}(x \in \mathbf{X}_k | D)\}_k) \tag{6}$$

where $H_{WP}$, $H_{CIL}$, and $H_{TP}$ are the cross-entropy values of WP, CIL, and TP, respectively. We now present our first theorem. The theorem connects CIL to WP and TP and suggests that by having a good WP or TP, the CIL performance improves as the upper bound for the CIL loss decreases.

**Theorem 1.** *If $H_{TP}(x) \leq \delta$ and $H_{WP}(x) \leq \epsilon$, we have $H_{CIL}(x) \leq \epsilon + \delta$.*

The detailed proof is given in Appendix A.1. This theorem holds regardless of whether WP and TP are trained together or separately. When they are trained separately, if WP is fixed and we let $\epsilon = H_{WP}(x)$, $H_{CIL}(x) \leq H_{WP}(x) + \delta$, which means if TP is better, CIL is better. Similarly, if TP is fixed, we have $H_{CIL}(x) \leq \epsilon + H_{TP}(x)$. When they are trained concurrently, there exists a functional relationship between $\epsilon$ and $\delta$ depending on implementation. But no matter what it is, when $\epsilon + \delta$ decreases, CIL gets better.

Theorem 1 holds for any $x \in \mathbf{X} = \bigcup_k \mathbf{X}_k$ that satisfies $H_{TP}(x) \leq \delta$ or $H_{WP}(x) \leq \epsilon$. To measure the overall performance under expectation, we present the following corollary.

**Corollary 1.** *Let $U(\mathbf{X})$ represent the uniform distribution on $\mathbf{X}$. i) If $\mathbb{E}_{x\sim U(\mathbf{X})}[H_{TP}(x)] \leq \delta$, then $\mathbb{E}_{x\sim U(\mathbf{X})}[H_{CIL}(x)] \leq \mathbb{E}_{x\sim U(\mathbf{X})}[H_{WP}(x)] + \delta$. Similarly, ii) $\mathbb{E}_{x\sim U(\mathbf{X})}[H_{WP}(x)] \leq \epsilon$, then $\mathbb{E}_{x\sim U(\mathbf{X})}[H_{CIL}(x)] \leq \epsilon + \mathbb{E}_{x\sim U(\mathbf{X})}[H_{TP}(x)]$.*

The proof is given in Appendix A.2. The corollary is a direct extension of Theorem 1 in expectation. The implication is that given TP performance, CIL is positively related to WP. The better the WP is, the better the CIL is as the upper bound of the CIL loss decreases. Similarly, given WP performance, a better TP performance results in a better CIL performance. Due to the positive relation, we can improve CIL by improving either WP or TP using their respective methods developed in each area.

### 3.3. Task prediction (TP) to OOD detection

Building on Eq. (2), we have studied the relationship of CIL, WP, and TP in Theorem 1. We now connect TP and OOD detection. They are shown to be dominated by each other to a constant factor.

We again use cross-entropy $H$ to measure the performance of TP and OOD detection of a trained network as in Section 3.2. To build the connection between $H_{TP}(x)$ and OOD detection of each task, we first define the notations of OOD detection. We use $\mathbf{P}'_k(x \in \mathbf{X}_k | D)$ to represent the probability distribution predicted by the $k$th task's OOD detector. Notice that the task prediction (TP) probability distribution $\mathbf{P}(x \in \mathbf{X}_k | D)$ is a categorical distribution over $T$ tasks, while the OOD detection probability distribution $\mathbf{P}'_k(x \in \mathbf{X}_k | D)$ is a Bernoulli distribution. For any $x \in \mathbf{X}$, define

$$H_{OOD,k}(x) = \begin{cases} H(1, \mathbf{P}'_k(x \in \mathbf{X}_k | D)) = -\log \mathbf{P}'_k(x \in \mathbf{X}_k | D), & x \in \mathbf{X}_k, \\ H(0, \mathbf{P}'_k(x \in \mathbf{X}_k | D)) = -\log \mathbf{P}'_k(x \in \mathbf{X} \backslash \mathbf{X}_k | D), & x \in \mathbf{X} \backslash \mathbf{X}_k. \end{cases} \tag{7}$$

Note that the OOD detection here is the *closed-world OOD detection*. But for presentation simplicity, we still use OOD detection below. In CIL, the term OOD detection probability for a task can be defined using the output values corresponding to the classes of the task.

Some examples of the function are the sigmoid of maximum logit value and the maximum softmax probability after re-scaling to 0 to 1. It is also possible to define the OOD detector directly as a function of tasks instead of a function of the output values of all classes of tasks, i.e. Mahalanobis distance. The following theorem shows that TP and OOD detection bound each other.

**Theorem 2.** *i) If $H_{TP}(x) \le \delta$, let $\mathbf{P}'_k(x \in \mathbf{X}_k|D) = \mathbf{P}(x \in \mathbf{X}_k|D)$, then $H_{OOD,k}(x) \le \delta, \forall k = 1, \dots, T$. ii) If $H_{OOD,k}(x) \le \delta_k, k = 1, \dots, T$, let $\mathbf{P}(x \in \mathbf{X}_k|D) = \frac{\mathbf{P}'_k(x \in \mathbf{X}_k|D)}{\sum_k \mathbf{P}'_k(x \in \mathbf{X}_k|D)}$, then $H_{TP}(x) \le (\sum_k \mathbf{1}_{x \in \mathbf{X}_k} e^{\delta_k})(\sum_k 1 - e^{-\delta_k})$, where $\mathbf{1}_{x \in \mathbf{X}_k}$ is an indicator function.*

See Appendix A.3 for the proof. As we use cross-entropy, the lower the bound, the better the performance is. The first statement (i) says that the OOD detection performance improves if the TP performance gets better (i.e., lower $\delta$). Similarly, the second statement (ii) says that the TP performance improves if the OOD detection performance on each task improves (i.e., lower $\delta_k$). Besides, since $(\sum_k \mathbf{1}_{x \in \mathbf{X}_k} e^{\delta_k})(\sum_k 1 - e^{-\delta_k})$ converges to 0 as $\delta_k$'s converge to 0 in order of $O(|\sum_k \delta_k|)$, we further know that $H_{TP}$ and $\sum_k H_{OOD,k}$ are equivalent in quantity up to a constant factor.

For the traditional CIL, Theorem 1 studied how CIL is related to WP and TP. Theorem 2 showed that TP and OOD detection bound each other. Now we explicitly give the upper bound of CIL in relation to WP and OOD detection of each task. The detailed proof can be found in Appendix A.5.

**Theorem 3.** *If $H_{OOD,k}(x) \le \delta_k$, $k = 1, \dots, T$ and $H_{WP}(x) \le \epsilon$, we have*

$$H_{CIL}(x) \le \epsilon + (\sum_k \mathbf{1}_{x \in \mathbf{X}_k} e^{\delta_k})(\sum_k 1 - e^{-\delta_k}),$$

*where $\mathbf{1}_{x \in \mathbf{X}_k}$ is an indicator function.*

### 3.4. Necessary conditions for improving CIL

In Theorem 1, we showed that good performances of WP and TP are sufficient to guarantee a good performance of CIL. In Theorem 3, we showed that good performances of WP and OOD detection are sufficient to guarantee a good performance of CIL. Again, for simplicity, OOD detection here refers to the closed-world OOD detection. For completeness, we study the necessary conditions of a well-performed CIL in this sub-section.

**Theorem 4.** *If $H_{CIL}(x) \le \eta$, then there exist i) a WP, s.t. $H_{WP}(x) \le \eta$, ii) a TP, s.t. $H_{TP}(x) \le \eta$, and iii) an OOD detector for each task, s.t. $H_{OOD,k} \le \eta$, $k = 1, \dots, T$.*

The detailed proof is given in Appendix A.6. This theorem tells that if a good CIL model is trained, then a good WP, a good TP, and a good OOD detector for each task are always implied. More importantly, by transforming Theorem 4 into its contraposition, we have the following statements: If for any WP, $H_{WP}(x) > \eta$, then $H_{CIL}(x) > \eta$. If for any TP, $H_{TP}(x) > \eta$, then $H_{CIL}(x) > \eta$. If for any OOD detector, $H_{OOD,k}(x) > \eta$, $k = 1, \dots, T$, then $H_{CIL}(x) > \eta$. Regardless of whether WP and TP (or OOD detection) are defined explicitly or implicitly by a CIL algorithm, the existence of a good WP and the existence of a good TP or OOD detection are necessary conditions for good CIL performance. Note that the OOD detection here is closed-world OOD detection.

**Remark 5.** It is important to note again that our study in this section is based on a CIL model that has already been built. In other words, our study tells the CIL designers what should be achieved in the final model. Clearly, one would also like to know how to design a strong CIL model based on the theoretical results, which also considers catastrophic forgetting (CF). One effective method is to make use of a strong existing TIL algorithm, which can already achieve no or little forgetting (CF), and combine it with a strong OOD detection algorithm (as mentioned earlier, most OOD detection methods can also perform WP). Thus, any improved method from the OOD detection community can be applied to CIL to produce improved CIL systems (see Sections 4.2.3 and 4.2.4).

### 3.5. Generalization to open-world continual learning

As mentioned at the beginning of this section, the first four subsections focused on the traditional closed-world CIL. This subsection generalizes or extends the theory to the open-world CIL, denoted by CIL$^+$. CIL$^+$ is CIL with an additional pseudo-task on top of the $T$ learned tasks representing OOD detection beyond the $T$ tasks, which we call the *OOD task* with a single pseudo-class (called *OOD class*) as we cannot predict the unseen class of an OOD sample because it is unknown. In this context, OOD detection is referred to as *open-world OOD detection*. For simplicity, we will continue using the term OOD detection.

We first note that Eq. (2) still applies because CIL$^+$ only adds a new OOD task with one OOD class. Theorem 1 for the closed-world CIL can be extended to the open-world CIL (CIL$^+$) by replacing $H_{TP}$ with $H_{TP^+}$, and $H_{CIL}$ with $H_{CIL^+}$. $H_{WP}$ stays the same as the WP definition has no change in CIL$^+$. The proof is trivially identical to the proof of Theorem 1. The key extension is to Theorem 2 of the traditional *closed-world* CIL so that test samples that do not belong to any of the $T$ already-learned tasks (i.e., OOD to the $T$ tasks) can also be detected.

Theorem 2 can be generalized to CIL$^+$ by changing the closed-world TP to *open-world TP*, denoted by TP$^+$, which must now predict the additional OOD task.

We denote $\mathbf{X}^+$ as the open-world OOD domain beyond $\mathbf{X}$. For any $x \in \mathbf{X} \cup \mathbf{X}^+$, define

$$H_{CIL^+}(x) = H(y, \{\mathbf{P}(x \in \mathbf{X}_{k,j}|D)\}_{k,j} \cup \{\mathbf{P}(x \in \mathbf{X}^+|D)\}), \tag{8}$$

$$H_{TP^+}(x) = H(\bar{y}, \{\mathbf{P}(x \in \mathbf{X}_k|D)\}_k \cup \{\mathbf{P}(x \in \mathbf{X}^+|D)\}). \tag{9}$$

For any $x \in \mathbf{X} \cup \mathbf{X}^+$, define

$$H_{OOD^+,k}(x) = \begin{cases} H(1, \mathbf{P}'_k(x \in \mathbf{X}_k|D)) = -\log \mathbf{P}'_k(x \in \mathbf{X}_k|D), \\ \qquad\qquad x \in \mathbf{X}_k, \\ H(0, \mathbf{P}'_k(x \in \mathbf{X}_k|D)) = -\log \mathbf{P}'_k(x \in (\mathbf{X} \cup \mathbf{X}^+)\backslash \mathbf{X}_k|D), \\ \qquad\qquad x \in (\mathbf{X} \cup \mathbf{X}^+)\backslash \mathbf{X}_k, \end{cases} \tag{10}$$

where OOD$^+$ denotes the *open-world OOD detection*.

It is clear that open-world OOD detection implies closed-world OOD detection, but the reverse is not true. Since the classification in the closed-world CIL is limited to the $T$ tasks learned so far, it cannot derive open-world OOD detection but only closed-world OOD detection. Thus, only closed-world OOD detection is *necessary* for the traditional closed-world CIL.

We now generalize Theorem 2 to the open-world CIL (CIL$^+$) setting with the following Corollary. The proof is given in Appendix A.4.

**Corollary 2.** *i) If $H_{TP^+}(x) \le \delta$, let $\mathbf{P}'_k(x \in \mathbf{X}_k|D) = \mathbf{P}(x \in \mathbf{X}_k|D)$, then $H_{OOD^+,k}(x) \le \delta, \forall k = 1, \ldots, T$. ii) If $H_{OOD^+,k}(x) \le \delta_k, k = 1, \ldots, T$, let $\mathbf{P}(x \in \mathbf{X}_k|D) = \frac{\mathbf{P}'_k(x \in \mathbf{X}_k|D)}{\sum_k \mathbf{P}'_k(x \in \mathbf{X}_k|D) + \prod_k (1 - \mathbf{P}'_k(x \in \mathbf{X}_k|D))}$ and $\mathbf{P}(x \in \mathbf{X}^+|D) = \frac{\prod_k (1 - \mathbf{P}'_k(x \in \mathbf{X}_k|D))}{\sum_k \mathbf{P}'_k(x \in \mathbf{X}_k|D) + \prod_k (1 - \mathbf{P}'_k(x \in \mathbf{X}_k|D))}$, then $H_{TP^+}(x) \le \max((\sum_k \mathbf{1}_{x \in \mathbf{X}_k} e^{\delta_k})(\sum_k (1 + \mathbf{1}_{x \in \mathbf{X}_k})(1 - e^{-\delta_k})), \prod_k e^{\delta_k} \sum_k 1 - e^{-\delta_k})$, where $\mathbf{1}_{x \in \mathbf{X}_k}$ is an indicator function.*

For Corollary 2, we have that $(\sum_k \mathbf{1}_{x \in \mathbf{X}_k} e^{\delta_k})(\sum_k (1 + \mathbf{1}_{x \in \mathbf{X}_k})(1 - e^{-\delta_k}))$ converges to 0 as $\delta_k$'s converges to 0 in order of $O(|\sum_k \delta_k| + \max_k |\delta_k|) = O(|\sum_k \delta_k|)$, and $\prod_k e^{\delta_k} \sum_k 1 - e^{-\delta_k}$ converges to 0 in order of $O(|\sum_k \delta_k|)$. Therefore, we know that $H_{TP^+}$ and $\sum_k H_{OOD^+,k}$ are equivalent in quantity up to a constant factor.

We can extend Theorem 3 to the open-world CIL$^+$ using Corollary 2. By substituting $H_{OOD,k}$ with $H_{OOD^+,k}$ and $H_{CIL}$ with $H_{CIL^+}$, we obtain a new upper bound for the open-world CIL, $\epsilon + \max((\sum_k \mathbf{1}_{x \in \mathbf{X}_k} e^{\delta_k})(\sum_k (1 + \mathbf{1}_{x \in \mathbf{X}_k})(1 - e^{-\delta_k})), \prod_k e^{\delta_k} \sum_k 1 - e^{-\delta_k})$. The proof is trivially identical to the original proof of Theorem 3.

We can establish the same theorem as Theorem 4 for CIL$^+$ by replacing $H_{OOD,k}$ with $H_{OOD^+,k}$ and $H_{CIL}$ with $H_{CIL^+}$. Again, the proof is trivially identical to the original proof of Theorem 4.

The new theorems establish that a good TP$^+$ or OOD$^+$ (open-world OOD detection) and a good WP are necessary and sufficient for a good CIL$^+$.

## 4. Proposed approach 1: combining TIL and OOD detection

Based on the above theoretical result, we have designed two approaches to solving CIL that employ OOD detection methods, more precisely *open-world OOD detection* methods. Although theoretically speaking, open-world OOD detection implies closed-world OOD detection, in practical applications, we often do not need to distinguish whether an OOD detection method is a closed-world or an open-world method as they usually can be used for either closed-world or open-world CIL. We just want them to be as accurate as possible for the applications.

This section presents the first approach, which combines a task incremental learning (TIL) method and an OOD detection method. The approach does not save any training data from previous tasks. The OOD detection method here is an open-world method as it does not use any information from the other tasks learned in the CIL process. The next section presents the second approach, which is based on replay and needs to save some training data from previous tasks.[4] The OOD detection method used there is a closed-world OOD detection method as it treats the replay data from previous tasks as the OOD data in the model building, but this method can also be used for open-world CIL.

[4] Note that this paper focuses on establishing a theoretical connection between novelty (or OOD) detection and class incremental learning (CIL). Our experiments show the validity of the theory. We also report the OOD detection results using AUC but this paper does not focus on the problem of real-time decision-making and learning using OOD detection to detect each novel instance, acquire its class label, and incrementally learn it. The reason is that this will involve setting an OOD score threshold to decide each OOD instance and interacting with human users to acquire the class label to learn. Such user interactions wouldn't give the system a large number of labeled training data. Then the highly challenging few-shot continual learning is required. We leave this to future work.

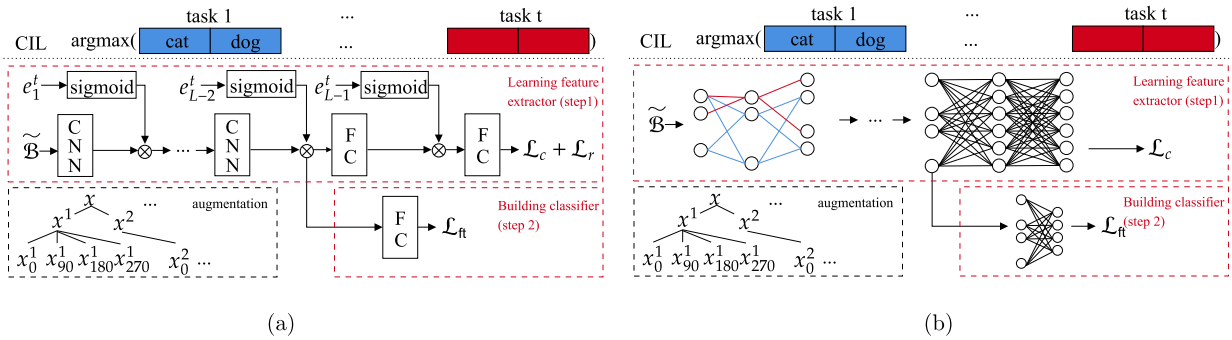


**Fig. 1.** Overview of prediction and training framework of HAT+CSI and Sup+CSI. **(a)** HAT+CSI: The CIL prediction is made by argmax over the concatenated output from each task. The training of each task uses CSI. That is, the training batch is augmented to give different views of the samples for contrastive training. The training consists of two steps following CSI. The first step learns the feature extractor by using the hard attention algorithm [61], which applies task embeddings to find hard masks at each layer. Then given the learned feature representations, it fine-tunes the classifier in step 2. **(b)** Sup+CSI: The CIL prediction is also made by taking argmax over the concatenated output values from each task as HAT+CSI. The model training for each task is similar to HAT+CSI except that it uses the Edge Popup algorithm of SupSup [55] for finding a sparse network for each task. The sparse networks are indicated by edges of different colors in the diagram. The second step fine-tunes the classifier only with the fixed feature extractor.

### *4.1. Combining a TIL method and an OOD detection method*

As mentioned earlier, several existing TIL methods can overcome CF. This proposed approach basically leverages the CF prevention ability in two TIL methods (HAT [61] and SupSup (Sup) [69]) and replaces their task learning methods with an OOD detection technique, called CSI [64], which can perform both within-task or IND prediction (WP) and OOD instance detection. Below, we first introduce the two TIL methods, HAT and SupSup, and the OOD detection method, CSI. The combinations give two new CIL methods, HAT+CSI and Sup+CSI. None of these methods needs to save any data from previous tasks.

Fig. 1 shows the overall training frameworks of HAT+CSI and Sup+CSI. Note that both HAT and Sup are multi-head methods (one head for each task) designed for task incremental learning (TIL).

#### *4.1.1. HAT: hard attention masks*

To prevent forgetting the trained OOD detection model $f^k \circ h^k$ for each task $k$ in subsequent task learning, the hard attention mask (HAT) [61] for TIL is employed (which prevents forgetting in the feature extractor). Specifically, in learning a task, a set of embeddings is trained to protect the important neurons so that the corresponding parameters are not interfered with by subsequent tasks. The importance of a neuron is measured by the 0-1 pseudo-step function, where 0 indicates not important and 1 indicates important (and thus protected).

The hard attention mask is an output of sigmoid function $u$ with a hyper-parameter $s$

$$a_l^k = u(se_l^k), \tag{11}$$

where $e_l^k$ is a learnable embedding at layer $l$ of task $k$. Since the step function is not differentiable, a sigmoid function with a large $s$ is used to approximate it. Sigmoid is approximately a 0-1 step function with a large $s$. The attention is multiplied to the output $h_l = \mathrm{ReLU}(W_l h_{l-1} + b_l)$ of layer $l$,

$$h'_l = a_l^k \otimes h_l \tag{12}$$

The $j$th element $a_{j,l}^k$ in the attention mask blocks (or unblocks) the information flow from neuron $j$ at layer $l$ if its value is 0 (or 1). With 0 value of $a_{j,l}^k$, the corresponding parameters in $W_l$ and $b_l$ can be freely changed as the output values $h'_l$ are not affected. The neurons with non-zero mask values are necessary to perform the task and thus need protection from catastrophic forgetting.

We modify the gradients of parameters that are important in performing the previous tasks $(1, \cdots, k-1)$ during training task $k$ so they are not interfered with. Denote the accumulated mask by

$$a_l^{<k} = \max(a_l^{<k-1}, a_l^{k-1}) \tag{13}$$

where max is element-wise maximum and the initial mask $a_l^0$ is a zero vector. It is a collection of mask values at layer $l$ where a neuron has value 1 if it has ever been activated previously. The gradient of parameter $w_{ij,l}$ is modified as

$$\nabla w'_{ij,l} = \left(1 - \min\left(a_{i,l}^{<k}, a_{j,l-1}^{<k}\right)\right) \nabla w_{ij,l} \tag{14}$$

where $a_{i,l}^{<k}$ is the $i$th unit of $a_l^{<k}$. The gradient flow is blocked if both neurons $i$ in the current layer and $j$ in the previous layer have been activated. We apply the mask for all layers except the last layer. The parameters in the last layer do not need to be protected as they are task-specific parameters.

A regularization is introduced to encourage sparsity in $a_l^k$ and parameter sharing with $a_l^{<k}$. The capacity of a network depletes when $a_l^{<k}$ becomes 1-vector in all layers. Despite a set of new neurons can be added to the network at any point in training for more capacity, we utilize resources more efficiently by minimizing the loss

$$\mathcal{L}_r = \lambda \frac{\sum_l \sum_i a_{i,l}^k \left(1 - a_{i,l}^{<k}\right)}{\sum_l \sum_i \left(1 - a_{i,l}^{<k}\right)} \tag{15}$$

where $\lambda$ is a hyper-parameter. The final objective of training a comprehensive task network without forgetting is

$$\mathcal{L} = \mathcal{L}_{ce} + \mathcal{L}_r \tag{16}$$

where $\mathcal{L}_{ce}$ is the cross-entropy loss. The overall framework of the algorithm is shown in Fig. 1(a).

Note that for TIL, HAT needs the task-id for each test instance in order to choose the right task model for prediction or classification. However, by replacing the original model building method for each task in HAT with the OOD detection method in CSI (more specifically, $\mathcal{L}_c$) during training, HAT+CSI does not require to know the task-id of each test instance at inference, which makes HAT+CSI suitable for CIL (class incremental learning). We will see the detailed prediction/classification method in Section 4.2.

#### 4.1.2. SupSup: supermasks in superposition

SupSup (Sup) [69] is also a highly effective method that can overcome forgetting in the TIL setting. Sup trains supermasks by Edge Popup algorithm in [55]. Specifically, given the initial weights of a base network $\mathbf{W}$, find binary masks $\mathbf{M}_k$ for task $k$ to minimize the cross-entropy loss.

$$\mathcal{L} = -\frac{1}{|\mathbf{X}_k|} \sum \log p(y|x,k), \tag{17}$$

where $\mathbf{X}_k$ is the training data for task $k$, and

$$p(y|x,k) = f(h(x; \mathbf{W} \otimes \mathbf{M}_k)), \tag{18}$$

where $\otimes$ indicates element-wise product. The masks are obtained by selecting the top $p$% of entries in the score matrices $\mathbf{V}$. The $p$ value determines the sparsity of the mask $\mathbf{M}_k$. The subnetwork found by Edge Popup algorithm is indicated by different colors in Fig. 1(b).

Like HAT, Sup is also for TIL and needs the task-id $k$ of each test instance at inference. With $k$, the system (which is referred to as Sup GG in the original Sup paper) uses the task-specific mask $\mathbf{M}_k$ to obtain the classification output. Like HAT+CSI, by replacing the cross-entropy loss in mask finding with the OOD detection loss in CSI, Sup+CSI also does not require the task-id of each test instance, which makes Sup+CSI applicable to CIL (class incremental learning). We will discuss the detailed prediction/classification method in Section 4.2.

#### 4.1.3. CSI: contrasting shifted instances for OOD detection

The OOD detection method CSI is based on contrastive learning [12,29], data and class augmentations, and results ensembling [64]. The OOD training process is similar to that of contrastive learning. It consists of two steps: Step 1 learns the feature representation by the composite $g \circ h$, where $h$ is the feature extractor and $g$ is the projection to contrastive representation, and Step 2 learns/fine-tunes the linear classifier $f$, mapping the feature representation of $h$ to the label space (the classifier is the OOD model)). This two-step training process is outlined in Fig. 1(b). In the following, we first describe the two-step training process and then explain how to make a prediction based on an ensemble method to further improve the prediction.

**Step 1 (Contrastive Loss for Feature Learning).** Supervised contrastive learning is used to try to repel data of different classes and align data of the same class more closely to make it easier to classify them. A key operation is data augmentation via transformations.

Given a batch of $N$ samples, each sample $x$ is first duplicated. Each version then goes through *three initial augmentations* (horizontal flip, color changes, and Inception crop [63]) to generate two different views $x^1$ and $x^2$ (they keep the same class label as $x$). Denote the augmented batch by $\mathcal{B}$, which now has $2N$ samples. In [21] and [64], it was shown that using image rotations is effective in learning OOD detection models because such rotations can effectively serve as out-of-distribution (OOD) training data. For each augmented sample $x \in \mathcal{B}$ with class $y$ of a task, we rotate $x$ by $90°, 180°, 270°$ to create three images, which are assigned *three new classes* $y_1, y_2$, and $y_3$, respectively. This results in a larger augmented batch $\tilde{\mathcal{B}}$. Since we generate three new images from each $x$, the size of $\tilde{\mathcal{B}}$ is $8N$. For each original class, we now have 4 classes. For a sample $x \in \tilde{\mathcal{B}}$, let $\tilde{\mathcal{B}}(x) = \tilde{\mathcal{B}} \backslash \{x\}$ and let $P(x) \subset \tilde{\mathcal{B}} \backslash \{x\}$ be a set consisting of the data of the same class as $x$ distinct from $x$. The contrastive representation of a sample $x$ is $z_x = g(h(x,k))/\|g(h(x,k))\|$, where $k$ is the current task. In learning, we minimize the supervised contrastive loss.

$$\mathcal{L}_c = \frac{1}{8N} \sum_{x \in \tilde{\mathcal{B}}} \frac{-1}{|P(x)|} \sum_{p \in P(x)} \log \frac{\exp(z_x \cdot z_p/\tau)}{\sum_{x' \in \tilde{\mathcal{B}}(x)} \exp(z_x \cdot z_{x'}/\tau)}, \tag{19}$$

where $\tau$ is a scalar temperature, $\cdot$ is dot product, and $\times$ is multiplication. The loss is reduced by repelling $z$ of different classes and aligning $z$ of the same class more closely. $\mathcal{L}_c$ basically trains a feature extractor with good representations for learning an OOD classifier.

Since the feature extractor is shared across tasks in continual learning, protection is needed to prevent catastrophic forgetting. HAT and Sup use their respective techniques to protect their feature extractor from forgetting. Therefore, the losses $\mathcal{L}$ of Eq. (17) and $\mathcal{L}_{ce}$ of Eq. (16) are replaced by Eq. (19) while the forgetting prevention mechanisms still hold.

**Step 2 (Fine-tuning the Classifier).** Given the feature extractor $h$ trained with the loss in Eq. (19), we *freeze* $h$ and only *fine-tune* the linear classifier $f$, which is trained to predict the classes of task $k$ *and* the augmented rotation classes. $f$ maps the feature representation to the label space in $\mathcal{R}^{4|C^k|}$, where 4 is the number of rotation classes including the original data with 0° rotation and $|C^k|$ is the number of original classes in task $k$. We minimize the cross-entropy loss,

$$\mathcal{L}_{\text{ft}} = -\frac{1}{|\tilde{\mathcal{B}}|} \sum_{(x,y)\in\tilde{\mathcal{B}}} \log \tilde{p}(y|x,k), \tag{20}$$

where ft indicates fine-tune, and

$$\tilde{p}(y|x,k) = \text{softmax}\,(f(h(x,k))) \tag{21}$$

where $f(h(x,k)) \in \mathcal{R}^{4|C^k|}$. The output $f(h(x,k))$ includes the rotation classes. The linear classifier is trained to predict the original *and* the rotation classes. Since an individual classifier is trained for each task and the feature extractor is frozen, no protection is necessary.

**Ensemble Class Prediction.** We now discuss the prediction of class label $y$ for a test sample $x$. Note that the network $f \circ h$ in Eq. (21) returns logits for rotation classes (including the original task classes). Note also for each original class label $j_k \in C^k$ (original classes) of a task $k$, we created three additional rotation classes. For class $j_k$, the classifier $f$ will produce four output values from its four rotation class logits, i.e., $f_{j_k,0}(h(x_0,k))$, $f_{j_k,90}(h(x_{90},k))$, $f_{j_k,180}(h(x_{180},k))$, and $f_{j_k,270}(h(x_{270},k))$, where 0, 90, 180, and 270 represent 0°, 90°, 180°, and 270° rotations respectively and $x_0$ is the original $x$. We compute an ensemble output $f_{j_k}(h(x))$ for each class $j_k \in C^k$ of task $k$,

$$f(h(x,k))_{j_k} = \frac{1}{4} \sum_{\text{deg}} f(h(x_{\text{deg}},k))_{j_k,\text{deg}}. \tag{22}$$

### 4.2. Experiments

We now present the experimental results of the combination techniques HAT+CSI and Sup+CSI for *class incremental learning* (CIL). We will also use another OOD detection method ODIN [38] to show that a better OOD detection method leads to better CIL results. We do not conduct extensive experiments on ODIN as it is much weaker than CSI in terms of ODD detection. Note that we will not report the ODD detection results for HAT+CSI and Sup+CSI in the open world in the continual learning process as the proposed method MORE in the next section performs better.

#### 4.2.1. Experimental datasets and baselines

**Datasets and CIL tasks**: We use three standard image classification benchmark datasets and construct five different CIL experiments.

**1. CIFAR-10** [32]: This dataset consists of 32x32 color images of 10 classes with 50,000 training and 10,000 testing samples. We construct an experiment (**C10-5T)** of 5 tasks with 2 classes per task.

**2. CIFAR-100** [32]: This dataset consists of 32x32 color images of 100 classes with 50,000 training and 10,000 testing samples. We construct two experiments of 10 tasks (**C100-10T**) and 20 tasks (**C100-20T**), where each task has 10 classes and 5 classes, respectively.

**3. Tiny-ImageNet** [34]: This is an image classification dataset with 64x64 color images of 200 classes with 100,000 training and 10,000 validation samples. Since the dataset does not provide labels for testing data, we use the validation data for testing. We construct two experiments of 5 tasks (**T-5T**) and 10 tasks (**T-10T**) with 40 classes per task and 20 classes per task, respectively.

**Baselines**: We use 18 diverse continual learning baselines:

**1.** One projection method (**OWM** [75]).

**2.** Two exemplar-free (no replay data is saved) regularization methods (**MUC** [42] and **PASS** [77]).

**3.** Nine replay-based methods (**LwF** [37], **iCaRL** [56], **A-GEM** [10], **EEIL** [7], **GD** [36], **Mnemonics** [43], **BiC** [70], **DER++** [6]], and **HAL** [9]).

**4.** Three parameter-isolation methods (**HAT** [61], **HyperNet** [49], and **SupSup** [69]).

**5.** Additionally, we report the accuracies of replay-based method **Co$^2$L** [8] and parameter isolation methods **CCG** [1] and **PR-Ent** [22] from their original papers as CCG has not released the code and we are unable to run Co$^2$L and PR-Ent on our machines.

#### 4.2.2. Training details and evaluation metrics

**Training Details.** For the backbone structure, we follow [69,77,6] and use ResNet-18 [19]. For CIFAR-100 and Tiny-ImageNet, the number of channels is doubled to fit more classes. For all baselines, the same ResNet-18 backbone architecture is employed except for OWM and HyperNet, for which we use their original architectures. OWM uses AlexNet. It is not obvious how to apply its orthogonal projection technique to the ResNet structure. HyperNet uses ResNet-32 and we are unable to replace it due to model initialization arguments unexplained in the original paper. For the replay methods, we use the memory buffer of 200 for CIFAR-10 and 2000 for CIFAR-100 and Tiny-ImageNet as in [56,6]. We use the hyper-parameters suggested by the authors. If we cannot reproduce

any result, we use 10% of the training data as a validation set to grid-search for good hyper-parameters. For our proposed methods, we report the hyper-parameters in Appendix F. All the results are averages over 5 runs with random seeds.

**Evaluation Metrics.**

**1.** *Average classification accuracy* over all classes after learning the last task. The final class prediction depends on *prediction methods* (see below). We also report *forgetting rate* in Appendix G.

**2.** *Average AUC* (Area Under the ROC Curve) over all task models for the evaluation of OOD detection. AUC is the main measure used in OOD detection papers. Using this measure, we show that a better OOD detection method will result in a better CIL performance. Let $AUC_k$ be the AUC score of task $k$. It is computed by using only the model (or classes) of task $k$ to score the test data of task $k$ as the in-distribution (IND) data and the test data from other tasks as the out-of-distribution (OOD) data. The average AUC score is: $AUC = \sum_k AUC_k / n$, where $n$ is the number of tasks.

It is not straightforward to change existing CL algorithms to include a new OOD detection method that needs training, e.g., CSI, except for TIL (task incremental learning) methods like HAT and Sup. For HAT and Sup, we can simply switch their methods for learning each task with CSI (see Section 4.1.1 and Section 4.1.2).

**Prediction Methods.** The theoretical result in Section 3 states that we use Eq. (2) to perform the final prediction. The first probability (WP) in Eq. (2) is easy to get as we can simply use the softmax values of the classes in each task. However, the second probability (TP) in Eq. (2) is tricky as each task is learned without the data of other tasks. There can be many options. We take the following approaches for prediction (which are a special case of Eq. (2), see below):

**1.** For those approaches that use a single classification head to include all classes learned so far, we predict as follows (which is also the approach taken by the existing papers.)

$$\hat{y} = \arg\max f(x) \tag{23}$$

where $f(x)$ is the logit output of the network.

**2.** For multi-head methods (e.g., HAT, HyperNet, and Sup), which use one head for each task, we use the concatenated output as

$$\hat{y} = \arg\max \bigoplus_k f(x)_k \tag{24}$$

where $\bigoplus$ indicate concatenation and $f(x)_k$ is the output of task $k$.[5]

These methods (in fact, they are the same method used in two different settings) are a special case of Eq. (2) if we define $OOD_k$ as $\sigma(\max f(x)_k)$, where $\sigma$ is the sigmoid. Hence, the theoretical results in Section 3 are still applicable. We present a detailed explanation of this prediction method and some other options in Appendix C. These two approaches work quite well.

#### *4.2.3. Better OOD detection produces better CIL performance*

The key theoretical result in Section 3 is that better OOD detection will produce better CIL performance. We compare a weaker OOD method ODIN with the strong CSI. ODIN is a post-processing method for OOD detection [38]. Note that it does not always improve the OOD detection performance compared to without the ODIN post-processing (see below).

**Applying ODIN.** We first train the baseline models using their original algorithms, and then apply temperature scaling and input noise of ODIN at testing for each task (no training data needed). More precisely, the output of class $j$ in task $k$ changes by temperature scaling factor $\tau_k$ of task $k$ as

$$s(x;\tau_k)_j = e^{f(x)_{kj}/\tau_k} / \sum_j e^{f(x)_{kj}/\tau_k} \tag{25}$$

and the input changes by the noise factor $\epsilon_k$ as

$$\tilde{x} = x - \epsilon_k \text{sign}(-\nabla_x \log s(x;\tau_k)_{\hat{y}}) \tag{26}$$

where $\hat{y}$ is the class with the maximum output value in task $k$. This is a positive adversarial example inspired by [15]. The values $\tau_k$ and $\epsilon_k$ are hyper-parameters and we use the same values for all tasks except for PASS, for which we use a validation set to tune $\tau_k$ (see Appendix B).

Table 2 gives the results for C100-10T. The CIL results clearly show that the CIL performance increases if the AUC increases with ODIN. For instance, the CIL of DER++ and Sup improves from 53.71 to 55.29 and 44.58 to 46.74, respectively, as the AUC increases from 85.99 to 88.21 and 79.16 to 80.58. It shows that when this method is incorporated into each task model in the existing trained CIL network, the CIL performance of the original method improves. We note that ODIN does not always improve the average AUC. For those experiencing a decrease in AUC, the CIL performance also decreases except LwF. The inconsistency of LwF is due to its severe classification bias towards later tasks as discussed in BiC [70]. The temperature scaling in ODIN has a similar effect as the bias correction in BiC, and the CIL of LwF becomes close to that of BiC after the correction. Regardless of whether ODIN improves AUC or

[5] The Sup paper proposed a one-shot task-id prediction assuming that the test instances come in a batch and all belong to the same task like iTAML. We assume a single test instance per batch. Its task-id prediction results in an accuracy of 50.2 on C10-5T, which is much lower than 62.6 by using Eq. (24). The task-id prediction of HyperNet also works poorly. The accuracy of its task-id prediction is 49.34 on C10-5T while it is 53.4 using Eq. (24). PR uses entropy to find task-id. Among many variations of PR, we use the variations that perform the best for each dataset with exemplar-free and single sample per batch at testing (i.e., no PR-BW).

**Table 2**
Performance Comparison between the Original Output and ODIN. Note that ODIN does not apply to iCaRL and Mnemonics as they are not based on softmax but some distance functions. As mentioned earlier, for **Co$^2$L**, **CCG**, and **PR-Ent**, they either have no code, or their codes do not run on our machine. The results for other datasets are in Appendix B.

| | AUC | | CIL | |
|---|---|---|---|---|
| Method | Original | ODIN | Original | ODIN |
| OWM | 71.31 | 70.06 | 28.91 | 28.88 |
| MUC | 72.69 | 72.53 | 30.42 | 29.79 |
| PASS | 69.89 | 69.60 | 33.00 | 31.00 |
| LwF | 88.30 | 87.11 | 45.26 | 51.82 |
| A-GEM | 78.01 | 79.00 | 9.29 | 13.48 |
| EEIL | 83.37 | 79.73 | 48.99 | 41.74 |
| GD | 85.37 | 82.98 | 49.67 | 47.28 |
| BiC | 87.89 | 86.73 | 52.92 | 48.65 |
| DER++ | 85.99 | 88.21 | 53.71 | 55.29 |
| HAL | 64.21 | 64.83 | 15.59 | 21.01 |
| HAT | 77.72 | 77.80 | 41.06 | 41.21 |
| HyperNet | 71.82 | 72.32 | 30.23 | 30.83 |
| Sup | 79.16 | 80.58 | 44.58 | 46.74 |

**Table 3**
Average CIL and AUC of HAT and Sup after applying OOD detection methods ODIN and CSI. ODIN is a traditional OOD detection method while CSI is a recent OOD detection method known to be better than ODIN. As CL methods produce better OOD detection performance by CSI, their CIL performances are better than the ODIN counterparts.

| CL | OOD | C10-5T | | C100-10T | | C100-20T | | T-5T | | T-10T | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | AUC | CIL | AUC | CIL | AUC | CIL | AUC | CIL | AUC | CIL |
| HAT | ODIN | 82.5 | 62.6 | 77.8 | 41.2 | 75.4 | 25.8 | 72.3 | 38.6 | 71.8 | 30.0 |
| | CSI | 91.2 | 87.8 | 84.5 | 63.3 | 86.5 | 54.6 | 76.5 | 45.7 | 78.5 | 47.1 |
| Sup | ODIN | 82.4 | 62.6 | 80.6 | 46.7 | 81.6 | 36.4 | 74.0 | 41.1 | 74.6 | 36.5 |
| | CSI | 91.6 | 86.0 | 86.8 | 65.1 | 88.3 | 60.2 | 77.1 | 48.9 | 79.4 | 45.7 |

not, the positive correlation between AUC and CIL (except LwF) verifies the efficacy of Theorem 3, indicating better OOD detection results in better CIL performances.

**Applying CSI.** We now apply the OOD detection method CSI. Due to its sophisticated data augmentation, supervised contrastive learning, and results ensemble, it is hard to apply CSI to other baselines without fundamentally changing them except for HAT and Sup (SupSup) as these methods are parameter isolation-based TIL methods. We can simply replace their model for training each task with CSI wholesale. As mentioned earlier, both HAT and Sup as TIL methods have almost no forgetting.

Table 3 reports the results of using CSI and ODIN. ODIN is a weaker OOD method than CSI. Both HAT and Sup improve greatly as the systems are equipped with a better OOD detection method CSI. These experiment results empirically demonstrate the efficacy of Theorem 3, i.e., the CIL performance can be improved if a better OOD detection method is used.

#### *4.2.4. Full comparison of HAT+CSI and Sup+CSI with baselines*

We now make a full comparison of the two proposed systems HAT+CSI and Sup+CSI designed based on the theoretical results with baselines. Since HAT and Sup are exemplar-free CL methods, HAT+CSI and Sup+CSI do not need to save any previous task data for replaying. Table 4 shows that HAT and Sup equipped with CSI outperform the baselines by large margins. DER++, the best replay method, achieves 66.0 and 53.7 on C10-5T and C100-10T, respectively, while HAT+CSI achieves 87.8 and 63.3 and Sup+CSI achieves 86.0 and 65.1. The large performance gap remains consistent in more challenging problems, T-5T and T-10T.

Due to the definition of OOD in the prediction method and the fact that each task is trained separately in HAT and Sup, the outputs $f(x)_k$ from different tasks can be in different scales, which will result in incorrect predictions. To deal with the problem, we can calibrate the output as $\alpha_k f(x)_k + \beta_k$ and use $OOD_k = \sigma(\alpha_k f(x)_k + \beta_k)$. The optimal $\alpha_k^*$ and $\beta_k^*$ for each task $k$ can be found by optimization with a memory buffer to save a very small number of training examples from previous tasks like that in the replay-based methods. We refer to the calibrated methods as HAT+CSI+c and Sup+CSI+c. They are trained by using a memory buffer of the same size as the replay methods (see Section 4.2.2). Table 4 shows that the calibration improves from their memory-free versions, i.e., without calibration. We provide the details about how to train the calibration parameters $\alpha_k$ and $\beta_k$ in Appendix D.

**Table 4**
Average accuracy (CIL) of all methods after all tasks are learned. The baselines are grouped into (a), (b), (c), and (d) for projection, regularization, replay, and parameter-isolation methods, respectively. Our proposed methods are grouped into (e). Exemplar-free methods are italicized. † indicates that in their original papers, PASS and Mnemonics are pre-trained with the first half of the classes. Their results with pre-train are 50.1 and 53.5 on C100-10T, respectively, which are still much lower than the proposed HAT+CSI and Sup+CSI without pre-training. We do not use pre-training in our experiment for fairness. ∗ indicates that iCaRL and Mnemonics report average incremental accuracy in their original papers. We report average accuracy over all classes after all tasks are learned. The last column Avg. shows the average CIL accuracy of each method over all datasets.

| | Method | C10-5T | C100-10T | C100-20T | T-5T | T-10T | Avg. |
|---|---|---|---|---|---|---|---|
| (a) | *OWM* | 51.8±0.05 | 28.9±0.60 | 24.1±0.26 | 10.0±0.55 | 8.6±0.42 | 24.7 |
| (b) | *MUC* | 52.9±1.03 | 30.4±1.18 | 14.2±0.30 | 33.6±0.19 | 17.4±0.17 | 29.7 |
| | *PASS*† | 47.3±0.98 | 33.0±0.58 | 25.0±0.69 | 28.4±0.51 | 19.1±0.46 | 30.6 |
| (c) | LwF | 54.7±1.18 | 45.3±0.75 | 44.3±0.46 | 32.2±0.50 | 24.3±0.26 | 40.2 |
| | iCaRL* | 63.4±1.11 | 51.4±0.99 | 47.8±0.48 | 37.0±0.41 | 28.3±0.18 | 45.6 |
| | A-GEM | 20.0±0.37 | 9.3±0.17 | 4.1±0.89 | 13.5±0.08 | 7.7±0.07 | 10.9 |
| | EEIL | 57.1±0.28 | 49.0±1.27 | 33.5±0.08 | 14.7±0.40 | 9.8±0.19 | 32.8 |
| | GD | 58.7±0.31 | 49.7±0.33 | 38.9±0.02 | 16.4±1.40 | 11.7±0.25 | 35.1 |
| | Mnemonics†* | 64.1±1.47 | 51.0±0.34 | 47.6±0.74 | 37.1±0.46 | 28.5±0.72 | 45.7 |
| | BiC | 61.4±1.74 | 52.9±0.64 | 48.9±0.54 | 41.7±0.74 | 33.8±0.40 | 47.7 |
| | DER++ | 66.0±1.20 | 53.7±1.30 | 46.6±1.44 | 35.8±0.77 | 30.5±0.47 | 46.5 |
| | HAL | 32.8±2.17 | 15.6±0.31 | 13.5±1.53 | 3.4±0.35 | 3.4±0.38 | 13.7 |
| | $Co^2L$ | 65.6 | | | | | |
| (d) | *CCG* | 70.1 | | | | | |
| | *HAT* | 62.7±1.45 | 41.1±0.93 | 25.6±0.51 | 38.5±1.85 | 29.8±0.65 | 39.5 |
| | *HyperNet* | 53.4±2.19 | 30.2±1.54 | 18.7±1.10 | 7.9±0.69 | 5.3±0.50 | 23.1 |
| | *Sup* | 62.4±1.45 | 44.6±0.44 | 34.7±0.30 | 41.8±1.50 | 36.5±0.36 | 44.0 |
| | *PR-Ent* | 61.9 | 45.2 | | | | |
| (e) | *HAT+CSI* | 87.8±0.71 | 63.3±1.00 | 54.6±0.92 | 45.7±0.26 | 47.1±0.18 | 59.7 |
| | *Sup+CSI* | 86.0±0.41 | 65.1±0.39 | 60.2±0.51 | 48.9±0.25 | 45.7±0.76 | 61.2 |
| | HAT+CSI+c | 88.0±0.48 | 65.2±0.71 | 58.0±0.45 | 51.7±0.37 | 47.6±0.32 | 62.1 |
| | Sup+CSI+c | 87.3±0.37 | 65.2±0.37 | 60.5±0.64 | 49.2±0.28 | 46.2±0.53 | 61.7 |

**Table 5**
TIL (WP) accuracy results of 3 best-performing baselines and our methods. The full results are given in Appendix E. The calibrated versions (+c) of our methods are omitted as calibration does not affect TIL performances.

| Method | C10-5T | C100-10T | C100-20T | T-5T | T-10T | Avg. |
|---|---|---|---|---|---|---|
| BiC | 95.4±0.35 | 84.6±0.48 | 88.7±0.19 | 61.5±0.60 | 62.2±0.45 | 78.5 |
| HAT | 96.7±0.18 | 84.0±0.23 | 85.0±0.98 | 61.2±0.72 | 63.8±0.41 | 78.1 |
| Sup | 96.6±0.21 | 87.9±0.27 | 91.6±0.15 | 64.3±0.24 | 68.4±0.22 | 81.8 |
| HAT+CSI | 98.7±0.06 | 92.0±0.37 | 94.3±0.06 | 68.4±0.16 | 72.4±0.21 | 85.2 |
| Sup+CSI | 98.7±0.07 | 93.0±0.13 | 95.3±0.20 | 65.9±0.25 | 74.1±0.28 | 85.4 |

We note that CSI uses extensive data augmentations in its OOD detection. However, the baseline systems do not. To be fair, we added the same data augmentations to the three top-performing baselines, Mnemonics, BiC, and DER++. The average accuracy values over the five CIL experiments are 36.66, 35.75, and 18.43 for Mnemonics, BiC, and DER++, respectively. Our methods, HAT+CSI, Sup+CSI, HAT+CSI+c, and Sup+CSI+c, achieve accuracy values of 59.7, 61.2, 62.1, and 61.7, respectively, which are significantly better. In fact, with the augmentations, the three baselines perform worse than their original versions. We believe the reason is that while augmentations improve the performance of the current task as they help learn finer-grained and more task-specific features, they also cause more model updates in learning a task due to the additional augmented data, which leads to significantly more forgetting of prior tasks. However, our technique incorporating robust TIL mechanisms prevents forgetting while also concurrently benefiting from the strong OOD detection performance for each task model of CSI, which exploits data augmentations.

Finally, as shown in Theorem 1, the CIL performance also depends on the TIL (WP) performance. We compare the TIL accuracies of the baselines and our methods in Table 5. Our systems again outperform the baselines by large margins on more challenging datasets (e.g., CIFAR100 and Tiny-ImageNet).

## 5. Proposed approach 2: out-of-distribution replay

The approach presented above does not save any training data from previous tasks except for the optional step of calibration. The method presented in this section is based on the replay approach to solving CIL, which saves a small number of training data from

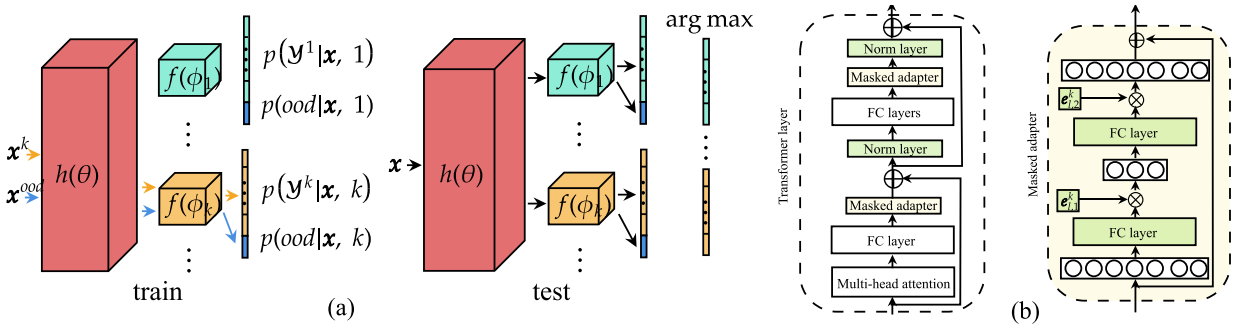


**Fig. 2.** (a) We train the feature extractor and the task classifier $k$ at task $k$. The output values of the classifier correspond to $|\mathbf{Y}_k|+1$ classes, in which the last class is for OOD (i.e., representing previous and unseen future classes). At inference/testing, the probability values of each task model without the OOD class are concatenated and the system chooses the class with the maximum score. (b) Transformer and adapter module. The masked adapter network consists of 2 fully connected layers and task-specific masks. During training, only the masked adapters and norm layers are updated and the other parts in the transformer layers remain unchanged.

each previous task. The proposed method is called *M*ulti-head model for continual learning via *O*OD *RE*play (MORE). As mentioned in Section 4, the OOD detection method used in this section is a closed-world method as it uses the saved samples from previous tasks as OOD samples in learning each new task.

### 5.1. The proposed MORE technique

Recall a replay-based method for continual learning works by memorizing or saving a small subset of the training samples from each previous task in a memory buffer. The saved data is called the *replay data*. In learning a new task, the new task data and the replay data are trained jointly to update the model. Clearly, using the replay data can partially deal with the inter-class separation (ICS) problem because the model sees some data from all classes learned so far. However, it cannot solve the ICS problem completely because the amount of replay data is often very small.

Unlike existing replay-based CIL methods, which simply use the replay data to update the decision boundaries between the old class and the new classes (in the new task), the proposed method uses the replay data to build an OOD detection model for each task in continual learning, which gives the name of the proposed method, i.e., *out-of-distribution replay*. Further, unlike existing OOD detection methods, which usually do not use any OOD data in training, the proposed method uses the replay data from previous tasks as the OOD data for the current new task in building its OOD detection model.

Unlike HAT+CSI and Sup+CSI, which do not use a pre-trained network, MORE trains a multi-head network as an adapter [23] to a pre-trained network (see Fig. 2(b)). Note that using a pre-trained transformer network and adapter modules is a common practice in existing continual learning methods in the natural language processing community [28,27]. Here we also leverage this approach for image classification tasks. In continual learning, the pre-trained network is frozen, only the adapters and the norm layers are trainable. Similar to HAT+CSI, a hard attention mask (HAT) is again employed to protect each task model or classifier to avoid forgetting. Each head is also an OOD detection model for a task, but, as mentioned above, MORE uses the replay data as the OOD data to build an OOD detection model. Since HAT has been described in Section 4.1.1, we will not discuss it further except to state that we need to use $\mathcal{L}_{ood}$ in Eq. (27) to replace $\mathcal{L}_{ce}$ in Eq. (16) after incorporating the trainable embedding $e^k$. We describe the whole training and prediction process in Appendix H.

#### 5.1.1. Training an OOD detection model

At task $k$, the system receives the training data $\mathcal{D}_k = \{(x_k^i, y_k^i)_{i=1}^{n_k}\}$, where $n_k$ is the number of samples, and $x_k^i \in \mathbf{X}_k$ is an input sample and $y_k^i \in \mathbf{Y}_k$ (the set of all classes of task $k$) is its class label. We train the feature extractor $z = h(x,k;\theta)$ and task-specific classifier $f(z;\phi_k)$ using $\mathcal{D}_k$ and the samples in the memory buffer $\mathcal{M}$. We treat the buffer data as OOD data to encourage the network to learn the current task and also detect ODD samples (the models or classifiers of the previous tasks are not touched). We achieve it by maximizing $p(y|x,k) = \text{softmax} f(h(x,k;\theta);\phi_k)$ for an IND sample $x \in \mathbf{X}_k$ and maximizing $p(ood|x,k)$ for an OOD sample $x \in \mathcal{M}$. The additional label *ood* is reserved for previous and possible future unseen classes. Fig. 2(a) shows the overall idea of the proposed approach. We formulate the problem as follows.

Given the training data $\mathbf{X}_k$ of size $n_k$ at task $k$ and the memory buffer $\mathcal{M}$ of size $M$, we minimize the loss

$$\mathcal{L}_{ood}(\theta,\phi_k) = -\frac{1}{M+N}\left(\sum_{(x,y)\in\mathcal{M}} \log p(ood|x,k) + \sum_{(x,y)\in\mathcal{D}^k} \log p(y|x,k)\right) \tag{27}$$

It is the sum of two cross-entropy losses. The first loss is for learning OOD samples while the second loss is for learning the classes from the current task. We optimize the shared parameter $\theta$ in the feature extractor. The task-specific classification parameters $\phi_k$ are independent of other tasks. The learned representation of the current data should be robust to OOD data. The classifier thus can classify both IND and OOD data.

In testing, we perform prediction by comparing the softmax probability output values using all the task classifiers from task 1 to $k$ without the OOD class as

$$\hat{y} = \arg\max \bigoplus_{1 \le j \le k} p(\mathbf{Y}_j | x, j) \tag{28}$$

where $\bigoplus$ is the concatenation over the output space. Fig. 2(a) shows the prediction rule. We are basically choosing the class with the highest softmax probability over all classes from all learned tasks.

#### 5.1.2. Back-updating the previous OOD models

Each task model works better if more diverse OOD data is provided during training. As in a replay-based approach, MORE saves an equal number of samples per class after each task [11]. The saved samples in the memory are used as OOD samples for each new task. Thus, in the beginning of continual learning when the system is trained on only a small number of tasks, the classes of samples in the memory are less diverse than after more tasks are learned. This makes the performance of OOD detection stronger for later tasks, but weaker in earlier tasks. To prevent this asymmetry, we update the model of each previous task so that it can also identify the samples from subsequent classes (which were unseen during the training of the previous task) as OOD samples.

At task $k$, we update the previous task models ($j = 1, \cdots, k-1$) as follows. Denote the samples of task $j$ in memory $\mathcal{M}$ by $\tilde{\mathbf{X}}_j$. We construct a new dataset using the current task dataset and the samples in the memory buffer. We randomly select $|\mathcal{M}|$ samples from the training data $\mathcal{D}_k$ and pool them with the remaining samples in $\mathcal{M}$ after removing the IND samples $\tilde{\mathcal{D}}_j$ of task $j$ from $\mathcal{M}$. We do not use the entire training data $\mathcal{D}_k$ as we do not want a large sample imbalance between IND and OOD. Denote the new dataset by $\tilde{\mathcal{M}}$. Using the data, we update only the parameters $\phi_j$ of the classifier for task $j$ with the feature representations frozen by minimizing the loss

$$\mathcal{L}(\phi_j) = -\frac{1}{2M}\left( \sum_{(x,y)\in\tilde{\mathcal{M}}} \log p(ood|x,j) + \sum_{(x,y)\in\tilde{\mathcal{D}}_j} \log p(y|x,j) \right) \tag{29}$$

We reduce the loss by updating the parameters of classifier $j$ to maximize the probability of the class if the sample belongs to task $j$ and maximize the OOD probability otherwise.

#### 5.1.3. Improving prediction performance by a distance based technique

We further improve the prediction in Eq. (28) by introducing a distance-based factor used as a coefficient to the softmax probabilities in Eq. (28). It is quite intuitive that if a test instance is close to a class, it is more likely to belong to the class. We thus propose to combine this new distance factor and the softmax probability output of the task $j$ model to make the final prediction decision. In some sense, this can be considered as an ensemble of the two methods.

We define the distance-based coefficient $s_j(x)$ of task $j$ for the test instance $x$ by the maximum of inverse Mahalanobis distance [35] between the feature of $x$ and the Gaussian distributions of the classes in task $j$ parameterized by the mean $\mu_j^i$ of the class $i$ in task $j$ and the sample covariance $S_j$. They are estimated by the features of class $i$'s training data for each class $i$ in task $j$. If a test instance is from the task, its feature should be close to the distribution that the instance belongs to. Conversely, if the instance is OOD to the task, its feature should not be close to any of the distributions of the classes in the task. More precisely, for task $j$ with class $y_1, \cdots, y_{|\mathbf{Y}_j|}$ (where $|\mathbf{Y}_j|$ represents the number of classes in task $k$), we define the coefficient $s_j(x)$ as

$$s_j(x) = \max\left[ 1/\mathrm{MD}(x; \mu_j^{y_1}, S_j), \cdots, 1/\mathrm{MD}(x; \mu_j^{y_{|\mathbf{Y}_j|}}, S_j) \right] \tag{30}$$

$\mathrm{MD}(x; \mu_j^i, S_j)$ is the Mahalanobis distance. The coefficient is large if at least one of the Mahalanobis distances is small but the coefficient is small if all the distances are large (i.e. the feature is far from all the distributions of the task). The parameters $\mu_j^i$ and $S_j$ can be computed and saved when each task is learned. The mean $\mu_j^i$ is computed using the training samples $\mathcal{D}_j^i$ of class $i$ as follows,

$$\mu_j^i = \sum_{x\in\mathcal{D}_j^i} h(x,i)/|\mathcal{D}_j^i| \tag{31}$$

and the covariance $S_j$ of task $j$ is the mean of covariances of the classes in task $j$,

$$S_j = \sum_{i\in\mathbf{Y}_j} S_j^i/|\mathbf{Y}_j| \tag{32}$$

where $S_j^i = \sum_{x\in\mathcal{D}_j^i}(x-\mu_j^i)^T(x-\mu_j^i)/|\mathcal{D}_j^i|$ is the sample covariance of class $i$. By multiplying the coefficient $s_j(x)$ to the original softmax probabilities $p(\mathbf{Y}_j|x,j)$, the task output $p(\mathbf{Y}_j|x,j)s_j(x)$ increases if $x$ is from task $j$ and decreases otherwise. The final prediction is made by (which replaces Eq. (28))

$$y = \arg\max \bigoplus_{1\le j\le k} p(\mathbf{Y}_j|x,j)s_j(x), \tag{33}$$

where $k$ is the last task that we have learned.

### 5.2. Experiments

We now report the experiment results of the proposed method MORE. For **experimental datasets**, we use the same three image classification benchmark datasets as in Section 4.2.1. For **baselines**, the same systems are used as well (see Section 4.2.1) except Mnemonics, HyperNet, CCG, Co$^2$L, and PR-Ent. Mnemonics requires optimization of training instances and it is not clear how to implement it for images after interpolation for a given input size of a pre-trained model (see below). For HyperNet, it is due to the reason explained in Training Details in Section 4.2.2. For CCG, Co$^2$L, and PR-Ent, CCG has no code and the codes of Co$^2$L, and PR-Ent do not run in our environment and thus we could not convert their codes to use a pre-trained model. Finally, we are left with 13 baselines. Note that HAT-CSI and Sup+CSI are not included as they are much weaker (up to 15% lower than MORE in accuracy) as CSI's approach of using contrastive learning and data augmentations does not work well with a pre-trained model.

**Evaluation Metrics.** We still use the same evaluation measures as we used in Section 4.2.2. **(1).** *Average classification accuracy* over all classes after learning the last task. **(2).** *Average AUC* (Area Under the ROC Curve) for evaluating OOD detection performance of continual learning in the open world. See Section 5.2.4 for more details.

#### 5.2.1. Pre-trained network

We pre-train a vision transformer [66] using a subset of the ImageNet data [58] and apply the pre-trained network/model to all baselines and our method. To ensure that there is no overlapping of data between ImageNet and our experimental datasets, we manually removed 389 classes from the original 1000 classes in ImageNet that are similar/identical to the classes in CIFAR-10, CIFAR-100, or Tiny-ImageNet. We pre-train the network with the remaining subset of 611 classes of ImageNet.

Using the pre-trained network, both our system and the baselines improve dramatically compared to their versions without using the pre-trained network. For instance, the two best baselines (DER++ and PASS) in our experiments achieved the average classification accuracy of 66.89 and 68.25 (after the final task) with the pre-trained network over 5 experiments while they achieved only 46.88 and 32.42 without using the pre-train network.

We insert an adapter module at each transformer layer to exploit the pre-trained transformer network in continual learning. During training, the adapter module and the layer norm are trained while the transformer parameters are unchanged to prevent forgetting in the pre-trained network.

#### 5.2.2. Training details

For all experiments, we use the same backbone architecture DeiT-S/16 [66] with a 2-layer adapter [23] at each transformer layer, and the same class order for both baselines and our method. The first fully connected layer in the adapter maps from dimension 384 to the bottleneck. The second fully connected layer following ReLU activation function maps from bottleneck to 384. The bottleneck dimension is the same for all adapters in a model. For our method, we use SGD with a momentum value 0.9. The back-updating method in Section 5.1.2 is also a hyper-parameter choice. If we apply it, we train each classifier for 10 epochs by SGD with a learning rate 0.01, batch size 16, and momentum value 0.9. We choose 500 for $s$ in Eq. (11) and 0.75 for $\lambda$ in Eq. (15) as recommended in [61]. We find a good set of learning rates and number of epochs on the validation set made of 10% of the training data. We follow [11] and save an equal number of random samples per class in the replay memory. Following the experiment settings in [56,77], we fix the size of the memory buffer and reduce the saved samples to accommodate a new set of samples after a new task is learned. We use the class order protocol in [56,6] by generating random class orders for the experiments. The baselines and our method use the same class ordering. We also report the size of memory required for each experiment in Appendix I.

For CIFAR-10, we split 10 classes into 5 tasks (2 classes per task). The bottleneck size in each adapter is 64. Following [6], we use the memory size 200, and train for 20 epochs with a learning rate 0.005, and apply the back-updating method in Section 5.1.2.

For CIFAR-100, we conducted 10 tasks and 20 tasks experiments, where each task has 10 classes and 5 classes, respectively. We double the bottleneck size of the adapter to learn more classes. We use the memory size 2000 following [56] and train for 40 epochs with learning rates 0.001 and 0.005 for 10 tasks and 20 tasks, respectively, and apply the back-updating method in Section 5.1.2.

For Tiny-ImageNet, two experiments are conducted. We split 200 classes into 5 and 10 tasks, where each task has 40 classes and 20 classes per task, respectively. We use the bottleneck size 128, and save 2000 samples in memory. We train with the learning rate 0.005 for 15 and 10 epochs for 5 tasks and 10 tasks, respectively. There is no need to use the back-updating method as the earlier tasks already have diverse OOD classes.

#### 5.2.3. Accuracy and forgetting rate results and analysis

**Average Accuracy.** Table 6 shows that our method MORE consistently outperforms the baselines. All the reported results are the averages of 5 runs. The last column gives the average of each row. We compare with the replay-based methods first. The best replay-based method on average over all the datasets is DER++. Our method MORE achieves an accuracy of 71.59, much better than 66.89 of DER++. This demonstrates that the existing replay-based methods utilizing the replay samples to update all learned classes are inferior to our MORE method using samples for OOD learning. The best baseline is the generative method PASS. Its average accuracy over all the datasets is 68.25, which is still poorer than our method's performance of 71.59. The performance of the multi-head method HAT [61] using task-id prediction is only 62.68, which is lower than many other baselines. Its performance is particularly low in experiments where the number of classes per task is small. For instance, its accuracy on C100-20T is 56.72, much lower than our method of 70.53 trained based on OOD detection.

**Accuracy with Smaller Memory Sizes.** For all the datasets, we run additional experiments with half of the original memory size and show that our method is even stronger with a smaller memory. The new memory sizes are 100, 1000, and 1000 for CIFAR-10,

**Table 6**
Average accuracy after the final task. '-XT' means X number of tasks. Our system MORE and all baselines use the pre-trained network. The baselines are grouped into (a), (b), (c), and (d) for projection, regularization, replay, and parameter-isolation methods, respectively. The last column shows the average accuracy of each method over all datasets and experiments. We highlight the best results in each column in bold.

| | Method | C10-5T | C100-10T | C100-20T | T-5T | T-10T | Avg. |
|---|---|---|---|---|---|---|---|
| (a) | OWM | 41.69±6.34 | 21.39±3.18 | 16.98±4.44 | 24.55±2.48 | 17.52±3.45 | 24.43 |
| (b) | MUC | 73.95±7.24 | 57.87±1.11 | 43.98±2.68 | 62.47±0.34 | 55.79±0.49 | 58.81 |
| | PASS | 86.21±1.10 | 68.90±0.94 | 66.77±1.18 | 61.03±0.38 | 58.34±0.42 | 68.25 |
| (c) | LwF | 67.59±4.27 | 66.50±1.93 | 67.54±0.97 | 33.51±4.36 | 36.85±4.46 | 54.40 |
| | iCaRL | 87.55±0.99 | 68.90±0.47 | 69.15±0.99 | 53.13±1.04 | 51.88±2.36 | 66.12 |
| | A-GEM | 56.33±7.77 | 25.21±4.00 | 21.99±4.01 | 30.53±3.99 | 21.90±5.52 | 31.20 |
| | EEIL | 82.34±3.13 | 68.08±0.51 | 63.79±0.66 | 53.34±0.54 | 50.38±0.97 | 63.59 |
| | GD | **89.16**±0.53 | 64.36±0.57 | 60.10±0.74 | 53.01±0.97 | 42.48±2.53 | 61.82 |
| | BiC | 67.44±3.93 | 64.47±1.30 | 67.69±1.97 | 38.78±1.26 | 40.98±2.39 | 55.87 |
| | DER++ | 84.63±2.91 | 69.73±0.99 | 70.03±1.46 | 55.84±2.21 | 54.20±3.28 | 66.89 |
| | HAL | 84.38±2.70 | 67.17±1.50 | 67.37±1.45 | 52.80±2.37 | 55.25±3.60 | 65.39 |
| (d) | HAT | 83.30±1.54 | 62.34±0.93 | 56.72±0.44 | 57.91±0.72 | 53.12±0.94 | 62.68 |
| | Sup | 80.91±2.99 | 62.49±0.49 | 57.32±1.11 | 58.43±0.67 | 54.52±0.45 | 62.74 |
| | MORE | **89.16**±0.96 | **70.23**±2.27 | **70.53**±1.09 | **64.97**±1.28 | **63.06**±1.26 | **71.59** |

**Table 7**
Average accuracy of the baselines and our method MORE with smaller memory sizes. We reduce the size of the memory buffer by half. The new sizes are 100, 1000, and 1000 for CIFAR10, CIFAR100, and Tiny-ImageNet. Numbers in bold are the best results in each column.

| | Method | C10-5T | C100-10T | C100-20T | T-5T | T-10T | Avg. |
|---|---|---|---|---|---|---|---|
| (a) | OWM | 41.69±6.34 | 21.39±3.18 | 16.98±4.44 | 24.55±2.48 | 17.52±3.45 | 24.43 |
| (b) | MUC | 73.95±7.24 | 57.87±1.11 | 43.98±2.68 | 62.47±0.34 | 55.79±0.49 | 58.81 |
| | PASS | 86.21±1.10 | 68.90±0.94 | 66.77±1.18 | 61.03±0.38 | 58.34±0.42 | 68.25 |
| (c) | LwF | 63.01±4.19 | 56.76±3.72 | 63.53±2.86 | 26.79±2.36 | 28.08±4.88 | 47.63 |
| | iCaRL | 86.08±1.19 | 66.96±2.08 | 68.16±0.71 | 47.27±3.22 | 49.51±1.87 | 63.60 |
| | A-GEM | 56.64±4.29 | 23.18±2.54 | 20.76±2.88 | 31.44±3.84 | 23.73±6.27 | 31.15 |
| | EEIL | 77.44±3.04 | 62.95±0.68 | 57.86±0.74 | 48.36±1.38 | 44.59±1.72 | 58.24 |
| | GD | 85.96±1.64 | 57.17±1.06 | 50.30±0.58 | 46.09±1.77 | 32.41±2.75 | 54.39 |
| | BiC | 56.28±3.31 | 58.42±2.48 | 62.19±1.20 | 33.29±2.65 | 28.44±2.41 | 47.72 |
| | DER++ | 80.09±3.00 | 64.89±2.48 | 65.84±1.46 | 50.74±2.41 | 49.24±5.01 | 62.16 |
| | HAL | 79.16±4.56 | 62.65±0.83 | 63.96±1.49 | 48.17±2.94 | 47.11±6.00 | 60.21 |
| (d) | HAT | 83.30±1.54 | 62.34±0.93 | 56.72±0.44 | 57.91±0.72 | 53.12±0.94 | 62.68 |
| | Sup | 80.91±2.99 | 62.49±0.49 | 57.32±1.11 | 58.43±0.67 | 54.52±0.45 | 62.74 |
| | MORE | **88.13**±1.16 | **71.69**±0.11 | **71.29**±0.55 | **64.17**±0.77 | **61.90**±0.90 | **71.44** |

CIFAR-100, and Tiny-ImageNet, respectively. Table 7 shows that MORE has experienced almost no performance drop with the reduced memory size while the memory-based baselines suffer from major performance reduction. The accuracy of the best memory-based baseline (DER++) has decreased the accuracy from 66.89 to 62.16, while MORE only decreases from 71.59 to 71.44, which shows that a small number of OOD samples is enough to enable the system to produce a robust OOD detection model.

**Average Forgetting Rate.** The average forgetting rate is defined as follows [43]: $\mathcal{F}^t = \sum_{k=1}^{t-1}(\mathcal{A}_k^{\text{init}} - \mathcal{A}_k^t)/(t-1)$, where $\mathcal{A}_k^{\text{init}}$ is the classification accuracy on samples of task $k$ right after learning the task $k$. We do not consider the task $t$ as it is the last task.

We compare the forgetting rate of our method MORE against the baselines using C10-5T, C100-10T, and C100-20T. Fig. 3 shows that the performance drop of our method as more tasks are learned is relatively lower than many baselines. MUC, A-GEM, HAT, and Sup achieve lower drops than our method. However, they are not able to adapt to new tasks well as the accuracy values of the four methods on C100-10T are 57.87, 25.21, 62.34, and 62.49, respectively, while our method MORE achieves 70.23. The performance gaps remain consistent in the other dataset. PASS experiences a smaller drop in performance on C100-10T and C100-20T than our method, but its accuracy of 68.90 and 66.77 are significantly lower than 70.23 and 70.53 of our MORE.

### *5.2.4. Out-of-distribution detection results*

As we explained in the introduction section, since our method MORE is based on OOD detection in building each task model, our method can naturally be used to detect test samples that are out-of-distribution for all the classes or tasks learned thus far. We are not aware of any existing continual learning system that has done such an evaluation. We evaluate the performance of the baselines and our system in this out-of-distribution scenario, which is also called the open set setting.

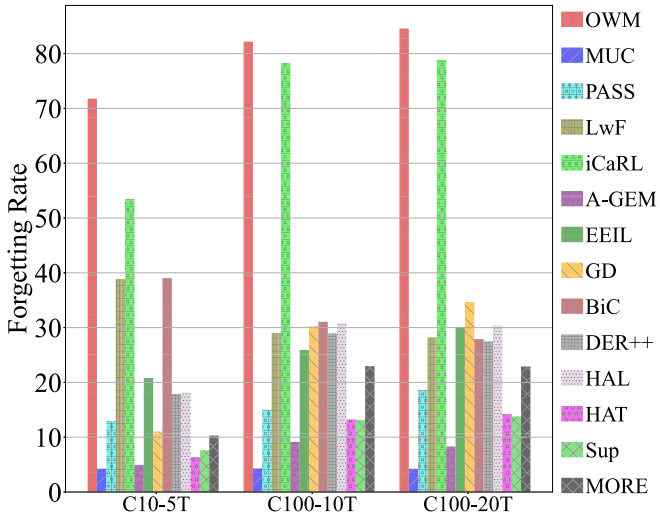


**Fig. 3.** Average forgetting rate (%). The lower the rate, the better the method is.

**Table 8**
AI-AUC results. Numbers in bold are the best results in each column.

| | Method | C10-5T | C100-10T | C100-20T | T-5T | T-10T | Avg. |
|---|---|---|---|---|---|---|---|
| (a) | OWM | 70.02±3.59 | 63.17±1.06 | 59.42±1.26 | 67.24±0.92 | 62.17±0.35 | 64.41 |
| (b) | MUC | 85.47±3.97 | 79.28±1.15 | 74.82±1.91 | **83.91**±0.54 | **81.42**±0.47 | 80.98 |
| | PASS | 84.57±1.54 | 77.74±1.40 | 77.42±1.44 | 77.07±2.14 | 74.79±2.36 | 78.32 |
| (c) | LwF | 72.18±4.15 | 74.95±0.39 | 75.40±0.64 | 66.44±1.14 | 65.52±0.64 | 70.90 |
| | iCaRL | 82.12±5.38 | 77.42±0.45 | 76.91±1.30 | 71.86±1.57 | 74.24±1.66 | 76.06 |
| | A-GEM | 74.92±5.62 | 64.19±0.86 | 60.23±0.95 | 67.88±1.28 | 63.08±1.12 | 66.06 |
| | EEIL | 87.19±2.31 | 78.89±1.32 | 77.69±1.40 | 74.82±0.79 | 73.45±1.33 | 78.39 |
| | GD | **89.71**±1.85 | 77.31±1.03 | 75.19±0.87 | 75.36±0.78 | 70.90±1.75 | 77.69 |
| | BiC | 71.29±3.57 | 74.49±0.72 | 75.71±0.60 | 67.45±0.89 | 66.63±0.77 | 71.11 |
| | DER++ | 84.61±2.64 | 78.42±0.64 | 78.37±0.42 | 74.80±1.72 | 74.86±1.93 | 78.09 |
| | HAL | 84.09±3.30 | 77.37±0.55 | 77.66±0.31 | 74.52±1.93 | 75.47±2.35 | 77.82 |
| (d) | HAT | 87.83±2.44 | 79.57±0.29 | 77.20±0.74 | 79.78±1.59 | 78.25±1.68 | 80.53 |
| | Sup | 87.06±3.68 | 80.54±0.12 | 77.81±0.66 | 80.01±0.71 | 78.96±0.64 | 80.87 |
| | MORE | 88.06±1.84 | **81.67**±1.27 | **80.97**±0.80 | 80.72±3.38 | 79.73±2.97 | **82.23** |

This OOD detection ability is highly desirable for a continual learning system because in a real-life environment in the open world, the system can be exposed to not only seen classes but also unseen classes. When the test sample is from one of the seen classes, the system should be able to predict its class. If the sample does not belong to any of the training classes seen so far (i.e., the sample is out-of-distribution), the system should detect it.

We formulate the performance of OOD detection of a continual learning system as follows. A continual learning system accepts and classifies a test sample after training task $k$ if the test sample is from one of the classes in tasks $1, \cdots, k$. If it is from one of the classes of the future tasks $k+1, \cdots, t$, it should be rejected as OOD (where $t$ is the last task in each evaluation).

We use maximum softmax probability (MSP) [20] as the OOD score of a test sample for the baselines and use maximum output with coefficient in Eq. (33) for our method MORE. We employ Area Under the ROC Curve (AUC) to measure the performance of OOD detection as AUC is the standard metric used in OOD detection papers [73]. We report *average incremental AUC* (AI-AUC) which is the average AUC result at all tasks except the last one as there is no more OOD data after the last task.

Table 8 shows that our method MORE outperforms all baselines consistently except MUC and GD. For MUC, it performs better than MORE on Tiny-ImageNet, but on average, it is poorer (80.98) than MORE (82.23). For GD, it outperforms MORE on C10-5T, but its overall performance is much lower as it achieves an average of only 77.69 over the 5 experiments. The best baselines based on accuracy are DER++ and PASS. Their average AI-AUC scores over the experiments shown in the last column are 78.09 for DER++ and 78.32 for PASS, but our method MORE achieves 82.23. In Appendix J, we report the OOD detection performances of the models on novel classes drawn from datasets that are completely different from the datasets used in the continual learning process.

**Table 9**
Ablation study.

| | C10-5T | C100-10T | C100-20T |
|---|---|---|---|
| Original | 91.01±2.48 | 76.93±1.58 | 75.76±2.35 |
| Coefficient (C) | 93.86±1.12 | 80.31±1.02 | 80.77±1.36 |
| Back (B) | 93.36±0.79 | 80.35±1.08 | 80.32±0.82 |
| C + B | 94.23±0.82 | 81.24±1.24 | 81.59±0.98 |

Performance gains with the proposed techniques. The row Original indicates the method without the coefficient and back-updating and the row Back means the back-updating method.

#### *5.2.5. Ablation study*

We conduct an ablation study to measure the performance gain by each proposed technique, **back-updating** in previous models in Section 5.1.2 and the **distance-based coefficient** in Section 5.1.3, using three experiments. The back-updating by Eq. (29) is to improve the earlier task models as they are trained with less diverse OOD data than later models. The modified output with coefficient in Eq. (30) is to improve the classification accuracy by combining the softmax probability of the task networks and the inverse Mahalanobis distances.

Table 9 compares accuracy obtained after applying each method. Both distance-based coefficient and back-updating show large improvements from the original method without any of the two techniques. Although the performance is already competitive with either technique, the performance improves further after applying them together.

## 6. Conclusion

This paper studied open-world continual learning (OWCL), which is necessary for learning in the open world. An open-world learning algorithm first needs to detect novel or out-of-distribution (OOD) items and then learn them continually or incrementally. This incremental learning of new items or classes is referred to as *class incremental learning* (CIL), which is a challenging setting of *continual learning*. Since the traditional CIL does not do OOD detection, we call it the *closed-world CIL*. In existing research, novelty/OOD detection and CIL are regarded as two completely different problems. This paper theoretically showed that the two problems can be unified. In particular, we first decomposed the closed-world CIL problem into *within-task prediction* (WP) and *task-id prediction* (TP), and then proved that TP is correlated with *closed-world OOD detection*. It then showed that a good performance of the two is both *necessary* and *sufficient* for good CIL performances. We then generalized the traditional closed-world CIL to the open-world CIL (or $\text{CIL}^+$). $\text{CIL}^+$ does both (open-world) OOD detection and CIL. Our theory thus connects and unifies open-world OOD/novelty detection and CIL in continual learning. This combination gives us the paradigm of open-world continual learning or $\text{CIL}^+$. The theoretical result also provides principled guidance for designing better continual learning algorithms. Based on the result, several new CIL methods have been designed. They outperform strong CIL baselines by a large margin and also perform novelty (or OOD) detection well in the open-world continual learning setting.

Several interesting future directions are worth pursuing. The central goal of these directions is to achieve learning autonomy. It means that the agent should discover new tasks to learn by itself and also take the initiative to acquire the ground-truth training data for learning. In the current research, the training data for each task is assumed to be provided by human engineers who have collected a large amount of labeled training data for each task. To achieve autonomy, the agent has to interact with human users, other agents, and/or the application environment. This requires an interactive module. To interact with humans, a natural language dialogue system is needed, which must also improve itself continually. Since new tasks are the discovered unknowns, an effective algorithm is also needed to decide whether and how to interact with human users to acquire the ground truth labels and training data when it is uncertain about an unfamiliar object that it sees. Another interesting direction is few-shot continual learning as human users usually cannot provide a large number of training examples. Few-shot continual learning is highly challenging as it has to deal with the difficulties of few-shot learning, catastrophic forgetting, and inter-task class separation. To address these challenges, a powerful foundation model with strong reasoning capabilities is likely to be necessary.

## CRediT authorship contribution statement

**Gyuhak Kim:** Writing – review & editing, Writing – original draft, Software, Methodology, Formal analysis, Conceptualization. **Changnan Xiao:** Writing – review & editing, Writing – original draft, Methodology, Formal analysis, Conceptualization. **Tatsuya Konishi:** Writing – review & editing, Methodology. **Zixuan Ke:** Writing – review & editing, Methodology. **Bing Liu:** Writing – review & editing, Supervision, Project administration, Methodology, Funding acquisition, Formal analysis, Conceptualization.

## Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Bing Liu reports financial support was provided by KDDI Research Inc (104543). Bing Liu reports financial support was provided by Defense Advanced Research Projects Agency (HR001120C0023).

**Acknowledgements**

We would like to express our sincere gratitude to the reviewers for their valuable and constructive feedback, which has significantly improved the rigor and precision of the theory presented in the paper. The work of Gyuhak Kim, Zixuan Ke, and Bing Liu was supported in part by four NSF grants (IIS-2229876, IIS-1910424, IIS-1838770, and CNS-2225427), a research contract from DARPA (HR001120C0023), and a research contract from KDDI (104543).

## Appendix A. Proof of theorems and corollaries

### A.1. Proof of Theorem 1

**Proof.** Since

$$\begin{aligned} H_{CIL}(x) &= H(y, \{\mathbf{P}(x \in \mathbf{X}_{k,j} | D)\}_{k,j}) \\ &= -\sum_{k,j} y_{k,j} \log \mathbf{P}(x \in \mathbf{X}_{k,j} | D) \\ &= -\log \mathbf{P}(x \in \mathbf{X}_{k_0,j_0} | D), \\ H_{WP}(x) &= H(\tilde{y}, \{\mathbf{P}(x \in \mathbf{X}_{k_0,j} | x \in \mathbf{X}_{k_0}, D)\}_j) \\ &= -\sum_{j} y_{k_0,j} \log \mathbf{P}(x \in \mathbf{X}_{k_0,j} | x \in \mathbf{X}_{k_0}, D) \\ &= -\log \mathbf{P}(x \in \mathbf{X}_{k_0,j_0} | x \in \mathbf{X}_{k_0}, D), \end{aligned}$$

and

$$\begin{aligned} H_{TP}(x) &= H(\bar{y}, \{\mathbf{P}(x \in \mathbf{X}_k | D)\}_k) \\ &= -\sum_{k} \bar{y}_k \log \mathbf{P}(x \in \mathbf{X}_k | D) \\ &= -\log \mathbf{P}(x \in \mathbf{X}_{k_0} | D), \end{aligned}$$

we have

$$\begin{aligned} H_{CIL}(x) &= -\log \mathbf{P}(x \in \mathbf{X}_{k_0,j_0} | D) \\ &= -\log \mathbf{P}(x \in \mathbf{X}_{k_0,j_0} | x \in \mathbf{X}_{k_0}, D) - \log \mathbf{P}(x \in \mathbf{X}_{k_0} | D) \\ &= H_{WP}(x) + H_{TP}(x) \\ &\le \epsilon + \delta. \quad \square \end{aligned}$$

### A.2. Proof of Corollary 1

**Proof.** By proof of Theorem 1, we have

$$H_{CIL}(x) = H_{WP}(x) + H_{TP}(x).$$

Taking expectations on both sides, we have i)

$$\begin{aligned} \mathbb{E}_{x \sim U(\mathbf{X})}[H_{CIL}(x)] &= \mathbb{E}_{x \sim U(\mathbf{X})}[H_{WP}(x)] + \mathbb{E}_{x \sim U(\mathbf{X})}[H_{TP}(x)] \\ &\le \mathbb{E}_{x \sim U(\mathbf{X})}[H_{WP}(x)] + \delta, \end{aligned}$$

and ii)

$$\begin{aligned} \mathbb{E}_{x \sim U(\mathbf{X})}[H_{CIL}(x)] &= \mathbb{E}_{x \sim U(\mathbf{X})}[H_{WP}(x)] + \mathbb{E}_{x \sim U(\mathbf{X})}[H_{TP}(x)] \\ &\le \epsilon + \mathbb{E}_{x \sim U(\mathbf{X})}[H_{TP}(x)]. \quad \square \end{aligned}$$

### A.3. Proof of Theorem 2

**Proof.** i) Assume $x \in \mathbf{X}_{k_0}$. For $k = k_0$, we have

$$\begin{aligned} H_{OOD,k_0}(x) &= -\log \mathbf{P}'_{k_0}(x \in \mathbf{X}_{k_0} | D) \\ &= -\log \mathbf{P}(x \in \mathbf{X}_{k_0} | D) \\ &= H_{TP}(x) \le \delta. \end{aligned}$$

For $k \neq k_0$, we have

$$\begin{aligned} H_{OOD,k}(x) &= -\log \mathbf{P}'_k(x \in \mathbf{X} \backslash \mathbf{X}_k | D) \\ &= -\log(1 - \mathbf{P}'_k(x \in \mathbf{X}_k | D)) \\ &= -\log(1 - \mathbf{P}(x \in \mathbf{X}_k | D)) \\ &= -\log \mathbf{P}(x \in \cup_{k' \neq k} \mathbf{X}_{k'} | D) \\ &\leq -\log \mathbf{P}(x \in \mathbf{X}_{k_0} | D) \\ &= H_{TP}(x) \leq \delta. \end{aligned}$$

ii) Assume $x \in \mathbf{X}_{k_0}$. For $k = k_0$, by $H_{OOD,k_0}(x) \leq \delta_{k_0}$, we have

$$-\log \mathbf{P}'_{k_0}(x \in \mathbf{X}_{k_0} | D) \leq \delta_{k_0},$$

which means

$$\mathbf{P}'_{k_0}(x \in \mathbf{X}_{k_0} | D) \geq e^{-\delta_{k_0}}.$$

For $k \neq k_0$, by $H_{OOD,k}(x) \leq \delta_k$, we have

$$-\log \mathbf{P}'_k(x \in \mathbf{X} \backslash \mathbf{X}_k | D) \leq \delta_k,$$

which means

$$\mathbf{P}'_k(x \in \mathbf{X}_k | D) \leq 1 - e^{-\delta_k}.$$

Therefore, we have

$$\begin{aligned} \mathbf{P}(x \in \mathbf{X}_{k_0} | D) &= \frac{\mathbf{P}'_{k_0}(x \in \mathbf{X}_{k_0} | D)}{\sum_{k'} \mathbf{P}'_{k'}(x \in \mathbf{X}_{k'} | D)} \\ &\geq \frac{e^{-\delta_{k_0}}}{1 + \sum_{k \neq k_0} 1 - e^{-\delta_k}} \\ &= \frac{e^{-\delta_{k_0}}}{e^{-\delta_{k_0}} + \sum_k 1 - e^{-\delta_k}} \\ &= \frac{1}{1 + e^{\delta_{k_0}} \sum_k 1 - e^{-\delta_k}}. \end{aligned}$$

Hence,

$$\begin{aligned} H_{TP}(x) &= -\log \mathbf{P}(x \in \mathbf{X}_{k_0} | D) \\ &\leq -\log \frac{1}{1 + e^{\delta_{k_0}} \sum_k 1 - e^{-\delta_k}} \\ &= \log(1 + e^{\delta_{k_0}} \sum_k 1 - e^{-\delta_k}) \\ &\leq e^{\delta_{k_0}} (\sum_k 1 - e^{-\delta_k}) \\ &= (\sum_k \mathbf{1}_{x \in \mathbf{X}_k} e^{\delta_k})(\sum_k 1 - e^{-\delta_k}). \quad \square \end{aligned}$$

### *A.4. Proof of Corollary 2*

**Proof.** i) Assume $x \in \mathbf{X}_{k_0}$. For $k = k_0$, we have

$$\begin{aligned} H_{OOD^+,k_0}(x) &= -\log \mathbf{P}'_{k_0}(x \in \mathbf{X}_{k_0} | D) \\ &= -\log \mathbf{P}(x \in \mathbf{X}_{k_0} | D) \\ &= H_{TP^+}(x) \leq \delta. \end{aligned}$$

For $k \neq k_0$, we have

$$\begin{aligned}
H_{OOD^+,k}(x) &= -\log \mathbf{P}'_k(x \in (\mathbf{X} \cup \mathbf{X}^+) \backslash \mathbf{X}_k | D) \\
&= -\log(1 - \mathbf{P}'_k(x \in \mathbf{X}_k | D)) \\
&= -\log(1 - \mathbf{P}(x \in \mathbf{X}_k | D)) \\
&= -\log \mathbf{P}(x \in (\cup_{k' \neq k} \mathbf{X}_{k'}) \cup X^+ | D) \\
&\leq -\log \mathbf{P}(x \in \mathbf{X}_{k_0} | D) \\
&= H_{TP^+}(x) \leq \delta.
\end{aligned}$$

ii.a) Assume $x \in \mathbf{X}_{k_0}$. For $k = k_0$, by $H_{OOD^+,k_0}(x) \leq \delta_{k_0}$, we have

$$-\log \mathbf{P}'_{k_0}(x \in \mathbf{X}_{k_0} | D) \leq \delta_{k_0},$$

which means

$$\mathbf{P}'_{k_0}(x \in \mathbf{X}_{k_0} | D) \geq e^{-\delta_{k_0}}.$$

For $k \neq k_0$, by $H_{OOD^+,k}(x) \leq \delta_k$, we have

$$-\log \mathbf{P}'_k(x \in (\mathbf{X} \cup \mathbf{X}^+) \backslash \mathbf{X}_k | D) \leq \delta_k,$$

which means

$$\mathbf{P}'_k(x \in \mathbf{X}_k | D) \leq 1 - e^{-\delta_k}.$$

Therefore, we have

$$\begin{aligned}
\mathbf{P}(x \in \mathbf{X}_{k_0} | D) &= \frac{\mathbf{P}'_{k_0}(x \in \mathbf{X}_{k_0} | D)}{\sum_{k'} \mathbf{P}'_{k'}(x \in \mathbf{X}_{k'} | D) + \prod_{k'} (1 - \mathbf{P}'_{k'}(x \in \mathbf{X}_{k'} | D))} \\
&\geq \frac{e^{-\delta_{k_0}}}{1 + \sum_{k \neq k_0} 1 - e^{-\delta_k} + (1 - e^{-\delta_{k_0}}) \prod_{k' \neq k} 1} \\
&= \frac{e^{-\delta_{k_0}}}{e^{-\delta_{k_0}} + \sum_k 1 - e^{-\delta_k} + 1 - e^{-\delta_{k_0}}} \\
&= \frac{1}{1 + e^{\delta_{k_0}} (1 - e^{-\delta_{k_0}} + \sum_k 1 - e^{-\delta_k})}.
\end{aligned}$$

Hence,

$$\begin{aligned}
H_{TP^+}(x) &= -\log \mathbf{P}(x \in \mathbf{X}_{k_0} | D) \\
&\leq -\log \frac{1}{1 + e^{\delta_{k_0}} (1 - e^{-\delta_{k_0}} + \sum_k 1 - e^{-\delta_k})} \\
&= \log(1 + e^{\delta_{k_0}} (1 - e^{-\delta_{k_0}} + \sum_k 1 - e^{-\delta_k})) \\
&\leq e^{\delta_{k_0}} (1 - e^{-\delta_{k_0}} + \sum_k 1 - e^{-\delta_k}) \\
&= (\sum_k \mathbf{1}_{x \in \mathbf{X}_k} e^{\delta_k})(\sum_k (1 + \mathbf{1}_{x \in \mathbf{X}_k})(1 - e^{-\delta_k})).
\end{aligned}$$

ii.b) Assume $x \in \mathbf{X}^+$. For $k = 1, \ldots, T$, by $H_{OOD^+,k}(x) \leq \delta_k$, we have

$$-\log \mathbf{P}'_k(x \in (\mathbf{X} \cup \mathbf{X}^+) \backslash \mathbf{X}_k | D) \leq \delta_k,$$

which means

$$\mathbf{P}'_k(x \in \mathbf{X}_k | D) \leq 1 - e^{-\delta_k}.$$

By definition, we have

$$\mathbf{P}(x \in \mathbf{X}^+ | D) = \frac{\prod_{k'} (1 - \mathbf{P}'_{k'}(x \in \mathbf{X}_{k'} | D))}{\sum_{k'} \mathbf{P}'_{k'}(x \in \mathbf{X}_{k'} | D) + \prod_{k'} (1 - \mathbf{P}'_{k'}(x \in \mathbf{X}_{k'} | D))}.$$

Hence,

$$
\begin{aligned}
H_{TP^+}(x) &= -\log \mathbf{P}(x \in \mathbf{X}^+ | D) \\
&= \log(1 + \frac{\sum_{k'} \mathbf{P}'_{k'}(x \in \mathbf{X}_{k'} | D)}{\prod_{k'}(1 - \mathbf{P}'_{k'}(x \in \mathbf{X}_{k'} | D))}) \\
&\leq \frac{\sum_{k'} \mathbf{P}'_{k'}(x \in \mathbf{X}_{k'} | D)}{\prod_{k'}(1 - \mathbf{P}'_{k'}(x \in \mathbf{X}_{k'} | D))} \\
&\leq \frac{\sum_{k'} 1 - e^{-\delta_{k'}}}{\prod_{k'} e^{-\delta_{k'}}} \\
&= \prod_k e^{\delta_k} \sum_k 1 - e^{-\delta_k}. \quad \square
\end{aligned}
$$

### A.5. Proof of Theorem 3

**Proof.** Using Theorem 1 and 2,

$$
\begin{aligned}
H_{CIL}(x) &= -\log \mathbf{P}(x \in \mathbf{X}_{k_0,j_0} | D) \\
&= -\log \mathbf{P}(x \in \mathbf{X}_{k_0,j_0} | x \in \mathbf{X}_{k_0}, D) - \log \mathbf{P}(x \in \mathbf{X}_{k_0} | D) \\
&= H_{WP}(x) + H_{TP}(x) \\
&\leq \epsilon + H_{TP}(x) \\
&\leq \epsilon + (\sum_k \mathbf{1}_{x \in \mathbf{X}_k} e^{\delta_k})(\sum_k 1 - e^{-\delta_k}) \quad \square
\end{aligned}
$$

### A.6. Proof of Theorem 4

**Proof.** i) Assume $x \in \mathbf{X}_{k_0,j_0} \subset \mathbf{X}_{k_0}$. Define $\mathbf{P}(x \in \mathbf{X}_{k,j} | x \in \mathbf{X}_k, D) = \mathbf{P}(x \in \mathbf{X}_{k,j} | D)$. According to proof of Theorem 1,

$$
\begin{aligned}
H_{WP}(x) &= -\log \mathbf{P}(x \in \mathbf{X}_{k_0,j_0} | x \in \mathbf{X}_{k_0}, D), \\
H_{CIL}(x) &= -\log \mathbf{P}(x \in \mathbf{X}_{k_0,j_0} | D).
\end{aligned}
$$

Hence, we have

$$
\begin{aligned}
H_{WP}(x) &= -\log \mathbf{P}(x \in \mathbf{X}_{k_0,j_0} | x \in \mathbf{X}_{k_0}, D) \\
&= -\log \mathbf{P}(x \in \mathbf{X}_{k_0,j_0} | D) \\
&= H_{CIL}(x) \leq \eta.
\end{aligned}
$$

ii) Assume $x \in \mathbf{X}_{k_0,j_0} \subset \mathbf{X}_{k_0}$. Define $\mathbf{P}(x \in \mathbf{X}_k | D) = \sum_j \mathbf{P}(x \in \mathbf{X}_{k,j} | D)$. According to proof of Theorem 1,

$$
\begin{aligned}
H_{TP}(x) &= -\log \mathbf{P}(x \in \mathbf{X}_{k_0} | D), \\
H_{CIL}(x) &= -\log \mathbf{P}(x \in \mathbf{X}_{k_0,j_0} | D).
\end{aligned}
$$

Hence, we have

$$
\begin{aligned}
H_{TP}(x) &= -\log \mathbf{P}(x \in \mathbf{X}_{k_0} | D) \\
&= -\log \sum_j \mathbf{P}(x \in \mathbf{X}_{k_0,j} | D) \\
&\leq -\log \mathbf{P}(x \in \mathbf{X}_{k_0,j_0} | D) \\
&= H_{CIL}(x) \leq \eta.
\end{aligned}
$$

iii) Assume $x \in \mathbf{X}_{k_0,j_0} \subset \mathbf{X}_{k_0}$. Define $\mathbf{P}'_i(x \in \mathbf{X}_k | D) = \mathbf{P}(x \in \mathbf{X}_k | D) = \sum_j \mathbf{P}(x \in \mathbf{X}_{k,j} | D)$. According to proof of Theorem 4 ii), we have

$$
H_{TP}(x) \leq \eta.
$$

According to proof of Theorem 2 i), we have

$$
H_{OOD,i}(x) \leq H_{TP}(x).
$$

Therefore,

$$
H_{OOD,i}(x) \leq H_{TP}(x) \leq \eta. \quad \square
$$

**Table B.10**
Performance comparison between the original output and output post-processed with OOD detection technique ODIN. Note that ODIN does not apply to iCaRL and Mnemonics as they are not based on softmax but some distance functions. The results for C100-10T are reported in the main paper.

| | | C10-5T | | C100-20T | | T-5T | | T-10T | |
|---|---|---|---|---|---|---|---|---|---|
| Method | CIL | AUC | CIL | AUC | CIL | AUC | CIL | AUC | CIL |
| OWM | Original | 81.33 | 51.79 | 71.90 | 24.15 | 58.49 | 10.00 | 59.48 | 8.57 |
| | ODIN | 71.72 | 40.65 | 68.52 | 23.05 | 58.46 | 10.77 | 59.38 | 9.52 |
| MUC | Original | 79.49 | 52.85 | 66.20 | 14.19 | 68.42 | 33.57 | 62.63 | 17.39 |
| | ODIN | 79.54 | 53.22 | 65.72 | 14.11 | 68.32 | 33.45 | 62.17 | 17.27 |
| PASS | Original | 66.51 | 47.34 | 70.26 | 24.99 | 65.18 | 28.40 | 63.27 | 19.07 |
| | ODIN | 63.08 | 35.20 | 69.81 | 21.83 | 65.93 | 29.03 | 62.73 | 17.78 |
| LwF | Original | 89.39 | 54.67 | 89.84 | 44.33 | 78.20 | 32.17 | 79.43 | 24.28 |
| | ODIN | 88.94 | 63.04 | 88.68 | 47.56 | 76.83 | 36.20 | 77.02 | 28.29 |
| A-GEM | Original | 85.93 | 20.03 | 74.48 | 4.14 | 72.33 | 13.52 | 76.42 | 7.66 |
| | ODIN | 86.43 | 34.03 | 75.12 | 6.99 | 72.46 | 14.69 | 76.75 | 8.50 |
| EEIL | Original | 89.72 | 57.09 | 85.96 | 33.46 | 64.82 | 14.67 | 64.87 | 9.79 |
| | ODIN | 89.20 | 59.47 | 85.46 | 35.16 | 57.01 | 11.92 | 55.42 | 6.88 |
| GD | Original | 91.23 | 58.69 | 86.76 | 38.83 | 68.63 | 16.36 | 69.61 | 11.73 |
| | ODIN | 90.39 | 60.53 | 86.64 | 42.33 | 60.75 | 13.43 | 63.92 | 11.83 |
| BiC | Original | 90.89 | 61.41 | 89.46 | 48.92 | 80.17 | 41.75 | 80.37 | 33.77 |
| | ODIN | 91.86 | 64.29 | 87.89 | 47.40 | 74.54 | 37.40 | 76.27 | 29.06 |
| DER++ | Original | 90.16 | 66.04 | 85.44 | 46.59 | 71.80 | 35.80 | 72.41 | 30.49 |
| | ODIN | 87.08 | 63.07 | 87.72 | 49.26 | 73.92 | 37.87 | 72.91 | 32.52 |
| HAL | Original | 86.16 | 32.82 | 65.59 | 13.51 | 53.00 | 3.42 | 57.87 | 3.36 |
| | ODIN | 76.27 | 44.75 | 64.46 | 17.40 | 53.26 | 4.80 | 58.13 | 4.74 |
| HAT | Original | 82.47 | 62.67 | 75.35 | 25.64 | 72.28 | 38.46 | 71.82 | 29.78 |
| | ODIN | 82.45 | 62.60 | 75.36 | 25.84 | 72.31 | 38.61 | 71.83 | 30.01 |
| HyperNet | Original | 78.54 | 53.40 | 72.04 | 18.67 | 54.58 | 7.91 | 55.37 | 5.32 |
| | ODIN | 79.39 | 56.72 | 73.89 | 23.8 | 54.60 | 8.64 | 55.53 | 6.91 |
| Sup | Original | 79.16 | 62.37 | 81.14 | 34.70 | 74.13 | 41.82 | 74.59 | 36.46 |
| | ODIN | 82.38 | 62.63 | 81.48 | 36.35 | 73.96 | 41.10 | 74.61 | 36.46 |

## Appendix B. Additional results and explanation regarding Table 1 in the main paper

In Section 4.2.3, we showed that a better OOD detection improves CIL performance. For the post-processing method ODIN, we only reported the results on C100-10T. Table B.10 shows the results on the other datasets.

A continual learning method with a better AUC shows a better CIL performance than other methods with lower AUC. For instance, original HAT achieves AUC of 82.47 while HyperNet achieves 78.54 on C10-5T. The CIL for HAT is 62.67 while it is 53.40 for HyperNet. However, there are some exceptions that this comparison does not hold. An example is LwF. Its AUC and CIL are 89.39 and 54.67 on C10-5T. Although its AUC is better than HAT, the CIL is lower. This is due to the fact that CIL improves with WP and TP according to Theorem 1. The contraposition of Theorem 4 also says if the cross-entropy of TIL is large, that of CIL is also large. Indeed, the average within-task prediction (WP) accuracy for LwF on C10-5T is 95.2 while the same for HAT is 96.7. Improving WP is also important in achieving good CIL performances.

For PASS, we had to tune $\tau_k$ using a validation set. This is because the softmax in Eq. (25) improves AUC by making the IND (in-distribution) and OOD scores more separable within a task, but deteriorates the final scores across tasks. To be specific, the test instances are predicted as one of the classes in the first task after softmax because the relative values between classes in task 1 are larger than the other tasks in PASS. Therefore, larger $\tau_1$ and smaller $\tau_k$, for $k > 1$, are chosen to compensate for the relative values.

## Appendix C. Definitions of TP

As noted in the main paper, the class prediction in Eq. (2) varies by the definition of WP and TP. The precise definition of WP and TP depends on implementation. Due to this subjectivity, we follow the prediction method in the existing approaches in continual learning, which is the $\arg\max$ over the output. In this section, we show that the $\arg\max$ over output is a special case of Eq. (2). We also provide CIL results using different definitions of TP.

We first establish another theorem. This is an extension of Theorem 2 and connects the standard prediction method to our analysis.

**Table C.11**
Average accuracy with a different prediction method.

| Method | C10-5T | C100-10T | C100-20T | T-5T | T-10T |
|---|---|---|---|---|---|
| *OWM* | 40.6±0.47 | 28.6±0.82 | 22.9±0.32 | 10.4±0.54 | 9.2±0.35 |
| *MUC* | 53.2±1.32 | 30.6±1.21 | 14.0±0.12 | 33.1±0.18 | 17.2±0.13 |
| *PASS*† | 33.6±0.71 | 18.5±1.85 | 20.8±0.85 | 21.4±0.44 | 13.0±0.55 |
| LwF | 63.0±0.34 | 51.9±0.88 | 47.5±0.62 | 35.9±0.32 | 27.8±0.29 |
| iCaRL* | 65.3±0.83 | 52.9±0.39 | 48.2±0.70 | 34.8±0.34 | 27.3±0.17 |
| A-GEM | 34.0±1.86 | 14.5±0.55 | 7.3±1.78 | 15.4±0.24 | 9.0±0.30 |
| EEIL | 59.5±0.41 | 41.8±0.78 | 37.9±6.11 | 15.1±0.00 | 7.5±0.19 |
| GD | 68.0±0.75 | 47.2±0.33 | 41.8±0.25 | 15.7±2.08 | 12.2±0.14 |
| Mnemonics†* | 65.6±1.55 | 50.7±0.72 | 47.9±0.71 | 36.3±0.30 | 27.7±0.78 |
| BiC | 65.5±0.81 | 50.8±0.69 | 47.2±0.71 | 37.0±0.58 | 29.1±0.34 |
| DER++ | 63.1±1.12 | 54.6±1.21 | 48.9±1.18 | 37.4±0.72 | 32.1±0.44 |
| HAL | 43.0±3.10 | 20.0±1.15 | 17.0±0.83 | 4.6±0.58 | 4.8±0.50 |
| *HAT* | 62.6±1.31 | 41.5±0.80 | 25.9±0.56 | 38.9±1.62 | 30.1±0.52 |
| *HyperNet* | 56.7±1.23 | 32.4±1.07 | 24.5±1.12 | 8.9±0.58 | 7.0±0.52 |
| *Sup* | 62.6±1.11 | 46.8±0.34 | 36.0±0.32 | 41.5±1.17 | 35.7±0.40 |
| *HAT+CSI* | 85.2±0.92 | 62.9±1.07 | 53.6±0.84 | 47.0±0.38 | 46.2±0.30 |
| *Sup+CSI* | 87.4±0.40 | 66.6±0.23 | 60.5±0.89 | 47.7±0.30 | 46.3±0.30 |
| HAT+CSI+c | 85.2±0.94 | 63.6±0.69 | 55.4±0.79 | 51.4±0.38 | 46.5±0.26 |
| Sup+CSI+c | 86.2±0.79 | 67.0±0.14 | 60.4±1.04 | 48.2±0.35 | 46.1±0.32 |

Average classification accuracy. The results are based on the class prediction method defined with WP and TP in Eq. (C.1) and Eq. (C.2), respectively. The results can be improved by finding optimal temperature scaling parameters.

**Theorem 5** *(Extension of Theorem 2). i) If $H_{TP}(x) \le \delta$, let $\mathbf{P}'_k(x \in \mathbf{X}_k|D) = \mathbf{P}(x \in \mathbf{X}_k|D)^{1/\tau_k}$, $\forall \tau_k > 0$, then $H_{OOD,k}(x) \le \max(\delta/\tau_k, -\log(1-(1-e^{-\delta})^{1/\tau_k}), \forall k = 1, \dots, T$.*

*ii) If $H_{OOD,k}(x) \le \delta_k, k = 1, \dots, T$, let $\mathbf{P}(x \in \mathbf{X}_k|D) = \frac{\mathbf{P}'_k(x \in \mathbf{X}_k|D)^{1/\tau_k}}{\sum_j \mathbf{P}'_j(x \in \mathbf{X}_j|D)^{1/\tau_j}}$, $\forall \tau_k > 0$, then $H_{TP}(x) \le \sum_k \frac{\mathbf{1}_{x \in \mathbf{X}_k}\delta_k}{\tau_k} + \frac{\sum_k (1-e^{-\delta_k})^{1/\tau_k}}{\sum_k \mathbf{1}_{x \in \mathbf{X}_k}(1-(1-e^{-\delta_k})^{1/\tau_k})}$, where $\mathbf{1}_{x \in \mathbf{X}_k}$ is an indicator function.*

In Theorem 5 (proof appears later), we can observe that $\delta/\tau_k$ decreases with the increase of $\tau_k$, while $-\log(1-(1-e^{-\delta})^{1/\tau_k})$ increases. Hence, when TP is given, let $\delta = H_{TP}(x)$, we can find the optimal $\tau_i$ to define OOD by solving $\delta/\tau_k = -\log(1-(1-e^{-\delta})^{1/\tau_k})$. Similarly, given OOD, let $\delta_k = H_{OOD,k}(x)$, we can find the optimal $\tau_1, \dots, \tau_T$ to define TP by finding the global minima of $\sum_k \frac{\mathbf{1}_{x \in \mathbf{X}_k}\delta_k}{\tau_k} + \frac{\sum_k (1-e^{-\delta_k})^{1/\tau_k}}{\sum_k \mathbf{1}_{x \in \mathbf{X}_k}(1-(1-e^{-\delta_k})^{1/\tau_k})}$. The optimal $\tau_k$ can be found using a memory buffer to save a small number of previous data like that in a replay-based continual learning method.

In Theorem 5 (ii), let $\mathbf{P}'_k(x \in \mathbf{X}_k|D) = \sigma(\max f(x)_k)$, where $\sigma$ is the sigmoid and $f(x)_k$ is the output of task $k$ and choose $\tau_k \approx 0$ for each $k$. Then $\mathbf{P}(x \in \mathbf{X}_k|D)$ becomes approximately 1 for the task $k$ where the maximum logit value appears and 0 for the rest tasks. Therefore, Eq. (2) in the paper

$$\mathbf{P}(x \in \mathbf{X}_{k,j}|D) = \mathbf{P}(x \in \mathbf{X}_{k,j}|x \in \mathbf{X}_k, D)\mathbf{P}(x \in \mathbf{X}_k|D)$$

is zero for all classes in tasks $k' \neq k$. Since only the probabilities of classes in task $k$ are non-zero, taking $\arg\max$ over all class probabilities gives the same class as $\arg\max$ over output logits.

We have also tried another definition of WP and TP. The considered WP is

$$\mathbf{P}(x \in \mathbf{X}_{k,j}|x \in \mathbf{X}_k, D) = \frac{e^{f(x)_{kj}/\nu_k}}{\sum_j e^{f(x)_{kj}/\nu_k}}, \tag{C.1}$$

where $\nu_k$ is a temperature scaling parameter for task $k$, and the TP is

$$\mathbf{P}(x \in \mathbf{X}_k|D) = \frac{\mathbf{P}'_k(x \in \mathbf{X}_k|D)}{\sum_k \mathbf{P}'_k(x \in \mathbf{X}_k|D)}, \tag{C.2}$$

where $\mathbf{P}'_k(x \in \mathbf{X}_k|D) = \max_j e^{f(x)_{kj}/\tau_k} / \sum_j e^{f(x)_{kj}/\tau_k}$ and $\tau_k$ is a temperature scaling parameter. This is the maximum softmax of task $k$. We choose $\nu_k = 0.1$ and $\tau_k = 5$ for all $k$. A good $\tau$ and $\nu$ can be found using grid search on a validation set. However, one can also find the optimal values by optimization using some past data saved for the memory buffer. The CIL results for the new prediction method are in Table C.11.

**Proof of Theorem 5.** i) Assume $x \in \mathbf{X}_{k_0}$.

For $k = k_0$, we have

$$H_{OOD,k_0}(x) = -\log \mathbf{P}'_{k_0}(x \in \mathbf{X}_{k_0}|D)$$

$$= -\frac{1}{\tau_{k_0}} \log \mathbf{P}(x \in \mathbf{X}_{k_0} | D)$$
$$= \frac{1}{\tau_{k_0}} H_{TP}(x) \le \frac{\delta}{\tau_{k_0}}.$$

For $k \neq k_0$, we have

$$\begin{aligned}
H_{OOD,k}(x) &= -\log \mathbf{P}'_k(x \notin \mathbf{X}_k | D) \\
&= -\log(1 - \mathbf{P}'_k(x \in \mathbf{X}_k | D)) \\
&= -\log(1 - \mathbf{P}(x \in \mathbf{X}_k | D)^{1/\tau_k}) \\
&= -\log(1 - (1 - \mathbf{P}(x \in \cup_{k' \neq k} \mathbf{X}_{k'} | D))^{1/\tau_k}) \\
&\le -\log(1 - (1 - \mathbf{P}(x \in \mathbf{X}_{k_0} | D))^{1/\tau_k}) \\
&= -\log(1 - (1 - e^{-H_{TP}(x)})^{1/\tau_k}) \\
&\le -\log(1 - (1 - e^{-\delta})^{1/\tau_k}).
\end{aligned}$$

ii) Assume $x \in \mathbf{X}_{k_0}$.
For $k = k_0$, by $H_{OOD,k_0}(x) \le \delta_{k_0}$, we have

$$-\log \mathbf{P}'_{k_0}(x \in \mathbf{X}_{k_0} | D) \le \delta_{k_0},$$

which means

$$\mathbf{P}'_{k_0}(x \in \mathbf{X}_{k_0} | D) \ge e^{-\delta_{k_0}}.$$

For $k \neq k_0$, by $H_{OOD,k}(x) \le \delta_k$, we have

$$-\log \mathbf{P}'_k(x \notin \mathbf{X}_k | D) \le \delta_k,$$

which means

$$\mathbf{P}'_k(x \in \mathbf{X}_k | D) \le 1 - e^{-\delta_k}.$$

Therefore, we have

$$\begin{aligned}
\mathbf{P}(x \in \mathbf{X}_{k_0} | D) &= \frac{\mathbf{P}'_{k_0}(x \in \mathbf{X}_{k_0} | D)^{1/\tau_{k_0}}}{\sum_k \mathbf{P}'_k(x \in \mathbf{X}_k | D)^{1/\tau_k}} \\
&\ge \frac{e^{-\delta_{k_0}/\tau_{k_0}}}{1 + \sum_{k \neq k_0} (1 - e^{-\delta_k})^{1/\tau_k}} \\
&= \frac{e^{-\delta_{k_0}/\tau_{k_0}}}{1 - (1 - e^{-\delta_{k_0}})^{1/\tau_{k_0}} + \sum_k (1 - e^{-\delta_k})^{1/\tau_k}} \\
&= \frac{e^{-\delta_{k_0}/\tau_{k_0}}}{1 - (1 - e^{-\delta_{k_0}})^{1/\tau_{k_0}}} \cdot \frac{1}{1 + \frac{\sum_k (1 - e^{-\delta_k})^{1/\tau_k}}{1 - (1 - e^{-\delta_{k_0}})^{1/\tau_{k_0}}}}.
\end{aligned}$$

Hence,

$$\begin{aligned}
H_{TP}(x) &= -\log \mathbf{P}(x \in \mathbf{X}_{k_0} | D) \\
&\le -\log \frac{e^{-\delta_{k_0}/\tau_{k_0}}}{1 - (1 - e^{-\delta_{k_0}})^{1/\tau_{k_0}}} \cdot \frac{1}{1 + \frac{\sum_k (1 - e^{-\delta_k})^{1/\tau_k}}{1 - (1 - e^{-\delta_{k_0}})^{1/\tau_{k_0}}}} \\
&= \frac{\delta_{k_0}}{\tau_{k_0}} + \log[1 - (1 - e^{-\delta_{k_0}})^{1/\tau_{k_0}}] + \log \left[ 1 + \frac{\sum_k (1 - e^{-\delta_k})^{1/\tau_k}}{1 - (1 - e^{-\delta_{k_0}})^{1/\tau_{k_0}}} \right] \\
&\le \frac{\delta_{k_0}}{\tau_{k_0}} + \frac{\sum_k (1 - e^{-\delta_k})^{1/\tau_k}}{1 - (1 - e^{-\delta_{k_0}})^{1/\tau_{k_0}}} \\
&= \sum_k \frac{\mathbf{1}_{x \in \mathbf{X}_k} \delta_k}{\tau_k} + \frac{\sum_k (1 - e^{-\delta_k})^{1/\tau_k}}{\sum_k \mathbf{1}_{x \in \mathbf{X}_k} (1 - (1 - e^{-\delta_k})^{1/\tau_k})}. \quad \square
\end{aligned}$$

**Table E.12**
The TIL results of all the systems.

| Method | C10-5T | C100-10T | C100-20T | T-5T | T-10T | Avg. |
|---|---|---|---|---|---|---|
| *OWM* | 85.0±0.07 | 59.6±0.83 | 65.4±0.48 | 22.4±0.87 | 28.1±0.55 | 52.1 |
| *MUC* | 95.1±0.10 | 77.3±0.83 | 73.4±9.16 | 55.9±0.26 | 47.2±0.22 | 69.8 |
| *PASS*† | 83.8±0.68 | 72.1±0.70 | 76.8±0.32 | 49.9±0.56 | 46.5±0.39 | 65.8 |
| LwF | 95.2±0.30 | 86.2±1.00 | 89.0±0.45 | 56.4±0.48 | 55.3±0.35 | 76.4 |
| iCaRL | 94.9±0.34 | 84.2±1.04 | 85.7±0.68 | 54.5±0.29 | 52.7±0.37 | 74.4 |
| A-GEM | 82.5±4.19 | 58.9±2.14 | 56.4±7.03 | 32.1±0.90 | 30.1±0.29 | 52.0 |
| EEIL | 93.4±0.02 | 83.1±3.13 | 88.4±2.07 | 30.3±0.89 | 25.9±0.04 | 64.2 |
| GD | 94.4±0.09 | 82.2±0.18 | 85.7±0.20 | 30.7±1.79 | 32.2±0.37 | 65.0 |
| Mnemonics†* | 94.5±0.46 | 82.3±0.30 | 86.2±0.46 | 54.8±0.16 | 52.9±0.66 | 74.1 |
| BiC | 95.4±0.35 | 84.6±0.48 | 88.7±0.19 | 61.5±0.60 | 62.2±0.45 | 78.5 |
| DER++ | 92.0±0.54 | 84.0±9.43 | 86.6±9.44 | 57.4±1.31 | 60.0±0.74 | 76.0 |
| HAL | 82.8±1.94 | 49.5±1.51 | 61.1±1.43 | 13.2±0.77 | 21.2±0.41 | 26.2 |
| *HAT* | 96.7±0.18 | 84.0±0.23 | 85.0±0.98 | 61.2±0.72 | 63.8±0.41 | 78.1 |
| *HyperNet* | 94.6±0.37 | 76.8±1.22 | 83.5±0.98 | 23.9±0.60 | 28.0±0.69 | 61.4 |
| *Sup* | 96.6±0.21 | 87.9±0.27 | 91.6±0.15 | 64.3±0.24 | 68.4±0.22 | 81.8 |
| *HAT+CSI* | 98.7±0.06 | 92.0±0.37 | 94.3±0.06 | 68.4±0.16 | 72.4±0.21 | 85.2 |
| *Sup+CSI* | 98.7±0.07 | 93.0±0.13 | 95.3±0.20 | 65.9±0.25 | 74.1±0.28 | 85.4 |

The calibrated versions (+c) of our methods are omitted as calibration does not affect TIL performance. Exemplar-free methods are italicized. The last column Avg. shows the average TIL accuracy of each method over all datasets.

## Appendix D. Output calibration

In this section, we discuss the output calibration technique used in Section 4.2.4 to improve the final prediction accuracy. Even if an OOD detection of each task was perfect (i.e. the model accepts and rejects IND and OOD samples perfectly), the system could make an incorrect class prediction if the magnitudes of outputs across different tasks are different. To ensure that the output values are comparable, we calibrate the outputs by scaling $\alpha_k$ and shifting $\beta_k$ for each task. The optimal parameters $(\alpha_k, \beta_k) \in R \times R$ can be found by solving the optimization problem using samples in the memory buffer. More precisely, denote the memory buffer $\mathcal{M}$ and calibration parameters $(\alpha, \beta) \in R^T \times R^T$, where $T$ is the number of learned tasks. After training $T$th task, we find optimal calibration parameters by minimizing the cross-entropy loss,

$$\mathcal{L} = -\frac{1}{|\mathcal{M}|} \sum_{(x,y)\in\mathcal{M}} \log p(y|x) \tag{D.1}$$

where $p(c|x)$ is computed using the softmax,

$$\text{softmax} \bigoplus [\alpha_k f(x)_k + \beta_k] \tag{D.2}$$

where $\bigoplus$ indicates the concatenation and $f(x)_k$ is the output of task $k$ as Eq. (24). Given the optimal parameters $(\alpha^*, \beta^*)$, we make final prediction as

$$\hat{y} = \arg\max \bigoplus [\alpha_k^* f(x)_k + \beta_k^*] \tag{D.3}$$

If we use $OOD_k = \sigma(\alpha_k^* f(x)_k + \beta_k^*)$, where $\sigma$ is the sigmoid, and $TP_k = OOD_k / \sum_{k'} OOD_{k'}$, the theoretical results in Section 3 hold.

## Appendix E. TIL (WP) results

The TIL (WP) results of all the systems are reported in Table E.12. HAT and Sup show strong performances compared to the other baselines as they leverage task-specific parameters. However, as shown in Theorem 1, the CIL depends on TP (or OOD). Without an OOD detection mechanism in HAT or Sup, they perform poorly in CIL as shown in the main paper. The contrastive learning in CSI also improves the IND prediction (i.e., WP), and this along with OOD detection results in strong CIL performance.

## Appendix F. Hyper-parameters

Here we report the hyper-parameters that we did not report in the main paper due to space limitations. We mainly report the hyper-parameters of the proposed methods, HAT+CSI, Sup+CSI, and their calibrated versions. For all the experiments of the proposed methods, we use the values chosen by the original CSI [64]. We use LARS [74] optimization with a learning rate 0.1 for training the feature extractor. We linearly increase the learning rate by 0.1 per epoch for the first 10 epochs. After that, we use cosine scheduler [45] without restart as in [64,12]. After training the feature extractor, we train the linear classifier for 100 epochs with SGD with a learning rate 0.1 and reduce the rate by 0.1 at 60, 75, and 90 epochs. For all the experiments except MNIST, we train the feature extractor for 700 epochs with batch size 128.

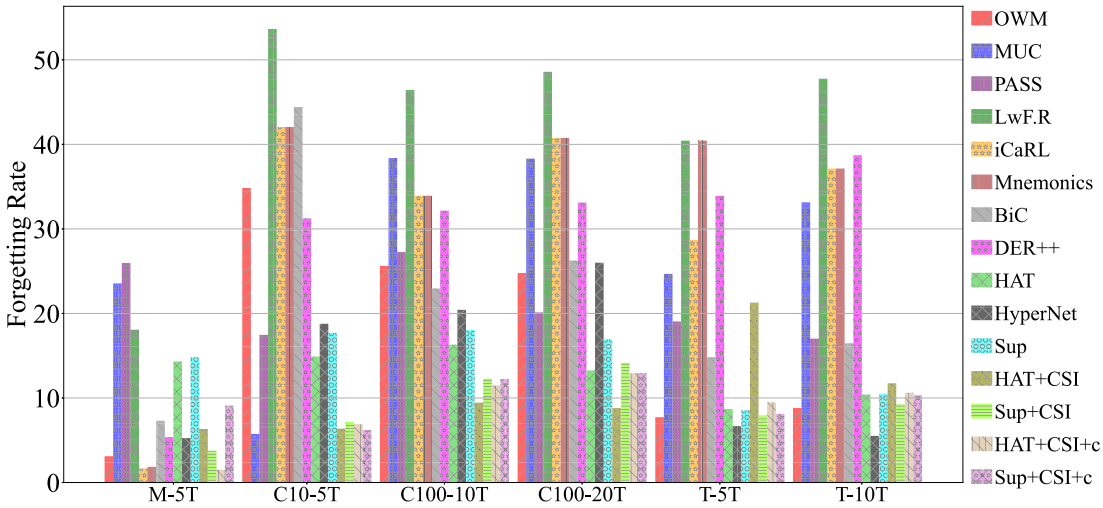


**Fig. G.4.** Average forgetting rate (%). The lower the value, the better the method is on forgetting.

For the following hyper-parameters, we use 10% of training data for validation to find a good set of values. For the number of epochs and batch size for MNIST, Sup+CSI trains for 1000 epochs with a batch size of 32 while HAT+CSI trains for 700 epochs with a batch size of 256. The hard attention regularization penalty $\lambda_i$ in HAT is different by experiments and task $i$. For MNIST, we use $\lambda_1 = 0.25$, and $\lambda_2 = \cdots = \lambda_5 = 0.1$. For C10-5T, we use $\lambda_1 = 1.0$, and $\lambda_2 = \cdots = \lambda_5 = 0.75$. For C100-10T, $\lambda_1 = 1.5$, and $\lambda_2 = \cdots = \lambda_{10} = 1.0$ are used. For C100-20T, $\lambda_1 = 3.5$, and $\lambda_2 = \cdots = \lambda_{20} = 2.5$ are used. For T-5T, $\lambda_i = 0.75$ for all tasks, and lastly, for T-10T, $\lambda_1 = 1.0$, and $\lambda_2 = \cdots = \lambda_{10} = 0.75$ are used. We use larger $\lambda_1$ for the first task than the later tasks as we have found that the larger regularization on the first task results in better accuracy. This is by the definition of regularization in HAT. The earlier task gives a lower penalty than the later tasks. We manually give a larger penalty to the first task. We did not search hyper-parameter $\lambda_t$ for tasks $t \geq 2$. For sparsity in Sup+CSI, we simply choose the least sparsity value of 32 used in the original Sup paper without parameter search.

Calibration methods (HAT+CSI+c and Sup+CSI+c) are based on its memory-free versions (i.e. HAT+CSI and Sup+CSI). Therefore, the model training part uses the same hyper-parameters as their calibration-free counterparts. For calibration training, we use SGD with a learning rate 0.01, 160 training iterations, and a batch size of 15 for HAT+CSI+c for all experiments. For Sup+CSI+c, we use the same values for all the experiments except for MNIST. For MNIST, we use a learning rate 0.05, batch size of 8, and run 280 iterations.

For the baselines, we use the hyper-parameters reported in the original papers or their code. If the hyper-parameters are unknown or the code does not reproduce the result (e.g., the baseline did not implement a particular dataset or the code had undergone significant version change), we search for the hyper-parameters as we did for HAT+CSI and Sup+CSI.

## Appendix G. Forgetting rate

We discuss forgetting rate (i.e., backward transfer) [44], which is defined for task $t$ as

$$\mathcal{F}^t = \frac{1}{t-1} \sum_{k=1}^{t-1} \mathcal{A}_k^{\text{init}} - \mathcal{A}_k^t, \tag{G.1}$$

where $\mathcal{A}_k^{\text{init}}$ is the classification accuracy of task $k$'s data after learning it for the first time and $\mathcal{A}_k^t$ is the accuracy of task $k$'s data after learning task $t$. We report the forgetting rate after learning the last task.

Fig. G.4 shows the forgetting rates of each method. Some methods (e.g., OWM, iCaRL) experience less forgetting than the proposed methods HAT+CSI and Sup+CSI on M-5T. On this dataset, all the systems performed well. For instance, OWM and iCaRL achieve 95.8% and 96.0% accuracy while HAT+CSI and HAT+CSI+c achieve 94.4 and 96.9% accuracy. As we have noted in the main paper, Sup+CSI and Sup+CSI+c achieve only 80.7 and 81.0 on M-5T although they have improved drastically from 70.1% of the base method Sup.

OWM and HyperNet show lower forgetting rates than HAT+CSI+c and Sup+CSI+c on T-5T and T-10T. However, they are not able to adapt to new classes as OWM and HyperNet achieve the classification accuracy of only 10.0% and 7.9%, respectively, on T-5T and 8.6% and 5.3% on T-10T. HAT+CSI+c and Sup+CSI+c achieves 51.7% and 49.2%, respectively, on T-5T and 47.6% and 46.2% on T-10T.

In fact, the performance reduction (i.e., forgetting) in our proposed methods occurs not because the systems forget the previous task knowledge, but because the systems learn more classes and the classification naturally becomes harder. The continual learning mechanisms (HAT and Sup) used in the proposed methods experience little or no forgetting because they find an independent subset

of parameters for each task, and the learned parameters are not interfered with during training. For the forgetting rate results in the TIL setting, refer to our earlier workshop paper [30].

## Appendix H. Pseudo-code

For task $k$, Let $p(y|\boldsymbol{x}, k) = \text{softmax} f(h(\boldsymbol{x}, k; \theta, \boldsymbol{e}^k); \phi_k)$, where $\theta$ is the parameters for the adapter, $\boldsymbol{e}^k$ is the trainable embedding for hard attentions, and $\phi_k$ is the set of parameters of the classification head of task $k$. Algorithm 1 and Algorithm 2 describe the training and testing processes, respectively. We add comments with the symbol "//".

**Algorithm 1** Training MORE.

**Require:** Memory $\mathcal{M}$, learning rate $\lambda$, a sequence of tasks $\mathcal{D} = \{\mathcal{D}^k\}_{k=1}$, and parameters $\{\theta, \boldsymbol{e}, \phi\}$, where $\boldsymbol{e}$ and $\phi$ are collections of task embeddings $\boldsymbol{e}^k$ and task heads $\phi_k$

// CL starts

**for** each task data $\mathcal{D}^k \in \mathcal{D}$ **do**

  // Model training

  **for** a batch $(\boldsymbol{X}_i^k, \boldsymbol{y})$ in $\mathcal{D}^k$, until converge **do**

    $\boldsymbol{X}_s = sample(\mathcal{M})$

    Compute loss (Eq. (27)+Eq. (16)) and gradients of parameters

    Modify the model parameters $\nabla\theta \leftarrow \nabla\theta'$ using Eq. (14)

    Update parameters as $\theta \leftarrow \theta - \lambda\nabla\theta,\ \boldsymbol{e}^k \leftarrow \boldsymbol{e}^k - \lambda\partial\mathcal{L},\ \phi_k \leftarrow \phi_k - \lambda\partial\mathcal{L}$

  **end for**

  // Back-updating in Section 5.1.2

  Randomly select $\tilde{\mathcal{D}} \subset \mathcal{D}^k$, where $|\tilde{\mathcal{D}}| = |\mathcal{M}|$

  **for** each task $j$, until converge **do**

    minimize $\mathcal{L}(\phi_j)$ of Eq. (29)

  **end for**

  // Obtain statistics in Section 5.1.3

  Compute $\boldsymbol{\mu}_j^k$ using Eq. (31) and $\boldsymbol{S}^k$ using Eq. (32)

**end for**

**Algorithm 2** MORE prediction.

**Require:** Test instance $\boldsymbol{x}$ and parameters $\{\theta, \boldsymbol{e}, \phi\}$

**for** each task $k$ **do**

  Obtain $p(\mathcal{Y}^k|\boldsymbol{x}, k)$

  Obtain $s^k(\boldsymbol{x})$ using Eq. (30)

**end for**

// Concatenate outputs for final prediction $y$ and OOD score $s$

$y = \arg\max \bigoplus_{1\le k\le t} p(\mathcal{Y}^k|\boldsymbol{x}, k)s^k(\boldsymbol{x})$ (i.e. Eq. (33))

$s = \max \bigoplus_{1\le k\le t} p(\mathcal{Y}^k|\boldsymbol{x}, k)s^k(\boldsymbol{x})$

## Appendix I. Size of memory required

In this section, we report the memory size required by each method in Section 5.2. The sizes include network size, replay buffer, and all other parameters or examples kept in memory simultaneously for a model to be functional.

We use an 'entry' to refer to a parameter or element in a vector or matrix to calculate the total memory required to train and test. The pre-trained backbone uses 21.6 million (M) entries (parameters). The adapter modules use 1.2M entries for CIFAR10 and 2.4M for other datasets. The baselines and our method use 22.9M and 24.1M entries for the model on CIFAR10 and other datasets, respectively. The unique technique of each method may add additional entries for training and test/inference.

The total memory required for each method without considering the replay memory buffer is reported in Table I.13. Our method is competitive in memory consumption. Baselines such as OWM and A-GEM take considerably more memory than our system. iCaRL and DER++ take the least amount of memory, but the differences between our method and theirs are only 0.8M, 1.8M, 3.6M, 1.0M, and 1.8M for C10-5T, C100-10T, C100-20T, T-5T, and T-10T.

Many replay-based methods (e.g., iCaRL, HAL) need to save the previous network for distillation during training. This requires an additional 1.2M or 2.4M entries for CIFAR10 or other datasets. Our method does not save the previous model as we do not use distillation.

Note that a large memory consumption usually comes from the memory buffer as the raw data is of size 32*32*3 or 64*64*3 for CIFAR and T-ImageNet. For a memory buffer of size 2000, a system needs 6.1M or 24.6M entries for CIFAR or T-ImageNet. Therefore, saving a smaller number of samples is important for reducing memory consumption. As we demonstrated in Table 6 and Table 7 in the main paper, our method performs better than the baselines even with a smaller memory buffer. In Table 6, we use large memory sizes (e.g., 200 and 2000 for CIFAR10 and other datasets). In Table 7, we reduce the memory size by half. When we compare the accuracy of our method in Table 7 to those of the baselines in Table 6, our method still outperforms them on all datasets. Our method with a smaller memory buffer achieves average classification accuracy of 88.13, 71.69, 71.29, 64.17, 61.90 on C10-5T, C100-10T,

**Table I.13**
Required memory.

| Method | C10-5T | C100-10T | C100-20T | T-5T | T-10T |
|---|---|---|---|---|---|
| OWM | 26.6M | 28.1M | 28.1M | 28.2M | 28.2M |
| MUC | 22.9M | 24.1M | 24.1M | 24.1M | 24.1M |
| PASS | 22.9M | 24.2M | 24.2M | 24.3M | 24.4M |
| LwF | 22.9M | 24.1M | 24.1M | 24.1M | 24.1M |
| iCaRL | 22.9M | 24.1M | 24.1M | 24.1M | 24.1M |
| A-GEM | 26.5M | 31.4M | 31.4M | 31.5M | 31.5M |
| EEIL | 22.9M | 24.1M | 24.1M | 24.1M | 24.1M |
| GD | 22.9M | 24.1M | 24.1M | 24.1M | 24.1M |
| BiC | 22.9M | 24.1M | 24.1M | 24.1M | 24.1M |
| DER++ | 22.9M | 24.1M | 24.1M | 24.1M | 24.1M |
| HAL | 22.9M | 24.1M | 24.1M | 24.1M | 24.1M |
| HAT | 23.0M | 24.7M | 25.4M | 24.6M | 25.1M |
| Sup | 24.7M | 33.7M | 45.7M | 27.7M | 33.7M |
| MORE | 23.7M | 25.9M | 27.7M | 25.1M | 25.9M |

Total memory (in entries) required for each method without the replay memory buffer.

**Table J.14**
AUC of the continually trained models following the final task.

| | MUC | Sup | HAT | MORE |
|---|---|---|---|---|
| LSUN | 92.60 | 93.60 | 92.88 | 95.61 |
| CIFAR-100 | 84.23 | 86.64 | 87.88 | 89.56 |
| Tiny-ImageNet | 93.43 | 94.78 | 91.90 | 95.45 |

(a)

| | MUC | Sup | HAT | MORE |
|---|---|---|---|---|
| LSUN | 82.33 | 80.26 | 76.82 | 82.96 |
| CIFAR-10 | 75.52 | 73.86 | 78.98 | 79.76 |
| CIFAR-100 | 78.58 | 75.98 | 77.62 | 80.21 |

(b)

(a) We use all 10 classes learned from 5 tasks of CIFAR-10 as IND and consider LSUN, CIFAR-10, and CIFAR-100 as OOD. (b) We use all 200 classes learned from 10 tasks of Tiny-ImageNet as IND and consider LSUN, CIFAR-10, and CIFAR-100 as OOD.

C100-20T, T-5T, and T-10T. On the other hand, the best baselines achieve 88.98, 69.73, 70.03, 61.03, 58.34 on the same experiments with a larger memory buffer.

## Appendix J. OOD detection on different datasets

In Section 5.2.4 of the main text, we used the classes from unseen tasks of the same dataset used in continual learning as OOD. Here, we also use classes drawn from a dataset that is completely different from the datasets used in the continual learning process. We use the continual learning models after the final task of C10-5T (with the least number of classes, 10) and T-ImageNet-10T (with the largest number of classes, 200) to detect novel samples from completely different datasets. We compare the novelty/OOD detection performance with the three best-performing baseline methods, MUC, HAT, and SupSup. Table J.14 shows that our method, MORE, outperforms the baselines on the completely different OOD classes from different datasets.

## Data availability

We use public data.

## References

[1] D. Abati, J. Tomczak, T. Blankevoort, S. Calderara, R. Cucchiara, E. Bejnordi, Conditional channel gated networks for task-aware continual learning, in: CVPR, 2020, pp. 3931–3940.
[2] R. Aljundi, E. Belilovsky, T. Tuytelaars, L. Charlin, M. Caccia, M. Lin, L. Caccia, Online continual learning with maximal interfered retrieval, in: NeurIPS, 2019.
[3] A. Ayub, A. Wagner, {EEC}: learning to encode and regenerate images for continual learning, in: International Conference on Learning Representations, 2021, https://openreview.net/forum?id=lWaz5a9lcFU.
[4] J. Bang, H. Kim, Y. Yoo, J.W. Ha, J. Choi, Rainbow memory: continual learning with a memory of diverse samples, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2021, pp. 8218–8227.

[5] A. Bendale, T. Boult, Towards open world recognition, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2015, pp. 1893–1902.
[6] P. Buzzega, M. Boschini, A. Porrello, D. Abati, S. Calderara, Dark experience for general continual learning: a strong, simple baseline, in: NeurIPS, 2020.
[7] F.M. Castro, M.J. Marín-Jiménez, N. Guil, C. Schmid, K. Alahari, End-to-end incremental learning, in: Proceedings of the European Conference on Computer Vision (ECCV), 2018.
[8] H. Cha, J. Lee, J. Shin, Co2l: contrastive continual learning, in: ICCV, 2021.
[9] A. Chaudhry, A. Gordo, P. Dokania, P. Torr, D. Lopez-Paz, Using hindsight to anchor past knowledge in continual learning, Proc. AAAI Conf. Artif. Intell. 35 (2021) 6993–7001, https://ojs.aaai.org/index.php/AAAI/article/view/16861.
[10] A. Chaudhry, M. Ranzato, M. Rohrbach, M. Elhoseiny, Efficient lifelong learning with a-gem, arXiv:1812.00420, 2018.
[11] A. Chaudhry, M. Rohrbach, M. Elhoseiny, T. Ajanthan, P. Dokania, P. Torr, M. Ranzato, Continual learning with tiny episodic memories, in: Workshop on Multi-Task and Lifelong Reinforcement Learning, 2019.
[12] T. Chen, S. Kornblith, M. Norouzi, G. Hinton, A simple framework for contrastive learning of visual representations, in: ICML, 2020.
[13] Z. Chen, B. Liu, Lifelong Machine Learning, Morgan & Claypool Publishers, 2018.
[14] G. Fei, S. Wang, B. Liu, Learning cumulatively to become more knowledgeable, in: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2016, pp. 1565–1574.
[15] I.J. Goodfellow, J. Shlens, C. Szegedy, Explaining and harnessing adversarial examples, in: ICLR, 2015.
[16] M. Gummadi, D. Kent, J.A. Mendez, E. Eaton, Shels: exclusive feature sets for novelty detection and continual learning without class boundaries, in: Conference on Lifelong Learning Agents, PMLR, 2022, pp. 1065–1085.
[17] K. Han, A. Vedaldi, A. Zisserman, Learning to discover novel visual categories via deep transfer clustering, in: International Conference on Computer Vision (ICCV), 2019.
[18] J. He, F. Zhu, Out-of-distribution detection in unsupervised continual learning, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2022, pp. 3850–3855.
[19] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: CVPR, 2016.
[20] D. Hendrycks, K. Gimpel, A baseline for detecting misclassified and out-of-distribution examples in neural networks, arXiv preprint, arXiv:1610.02136, 2016.
[21] D. Hendrycks, M. Mazeika, S. Kadavath, D. Song, Using self-supervised learning can improve model robustness and uncertainty, in: NeurIPS, 2019, pp. 15663–15674.
[22] C. Henning, M. Cervera, F. D'Angelo, J. Von Oswald, R. Traber, B. Ehret, S. Kobayashi, B.F. Grewe, J. Sacramento, Posterior meta-replay for continual learning, Adv. Neural Inf. Process. Syst. 34 (2021).
[23] N. Houlsby, A. Giurgiu, S. Jastrzebski, B. Morrone, Q. De Laroussilhe, A. Gesmundo, M. Attariyan, S. Gelly, Parameter-efficient transfer learning for nlp, in: International Conference on Machine Learning, PMLR, 2019, pp. 2790–2799.
[24] W. Hu, Q. Qin, M. Wang, J. Ma, B. Liu, Continual learning by using information of each class holistically, in: Proceedings of the AAAI Conference on Artificial Intelligence, 2021, pp. 7797–7805.
[25] M. Jafarzadeh, A.R. Dhamija, S. Cruz, C. Li, T. Ahmad, T.E. Boult, A review of open-world learning and steps toward open-world learning without labels, 2021, arXiv e-prints, arXiv–2011.
[26] R. Karakida, S. Akaho, Learning curves for continual learning in neural networks: self-knowledge transfer and forgetting, in: International Conference on Learning Representations, 2022.
[27] Z. Ke, B. Liu, Continual learning of natural language processing tasks: a survey, arXiv preprint, arXiv:2211.12701, 2022.
[28] Z. Ke, B. Liu, N. Ma, H. Xu, L. Shu, Achieving forgetting prevention and knowledge transfer in continual learning, Adv. Neural Inf. Process. Syst. 34 (2021).
[29] P. Khosla, P. Teterwak, C. Wang, A. Sarna, Y. Tian, P. Isola, A. Maschinot, C. Liu, D. Krishnan, Supervised contrastive learning, arXiv preprint, arXiv:2004.11362, 2020.
[30] G. Kim, S. Esmaeilpour, C. Xiao, B. Liu, Continual learning based on ood detection and task masking, in: CVPR 2022 Workshop on Continual Learning, 2022.
[31] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A.A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska, et al., Overcoming catastrophic forgetting in neural networks, Proc. Natl. Acad. Sci. 114 (2017) 3521–3526.
[32] A. Krizhevsky, G. Hinton, Learning multiple layers of features from tiny images, Technical Report TR-2009, University of Toronto, Toronto, 2009.
[33] P. Langley, Open-world learning for radically autonomous agents, in: Proceedings of the AAAI Conference on Artificial Intelligence, 2020, pp. 13539–13543.
[34] Y. Le, X. Yang, Tiny Imagenet Visual Recognition Challenge, 2015.
[35] K. Lee, K. Lee, H. Lee, J. Shin, A simple unified framework for detecting out-of-distribution samples and adversarial attacks, Adv. Neural Inf. Process. Syst. 31 (2018).
[36] K. Lee, K. Lee, J. Shin, H. Lee, Overcoming catastrophic forgetting with unlabeled data in the wild, in: CVPR, 2019.
[37] Z. Li, D. Hoiem, Learning without forgetting, in: ECCV, Springer, 2016, pp. 614–629.
[38] S. Liang, Y. Li, R. Srikant, Enhancing the reliability of out-of-distribution image detection in neural networks, in: ICLR, 2018, https://openreview.net/forum?id=H1VGkIxRZ.
[39] B. Liu, Learning on the job: online lifelong and continual learning, in: AAAI, 2020.
[40] B. Liu, S. Mazumder, E. Robertson, S. Grigsby, Ai autonomy: self-initiated open-world continual learning and adaptation, AI Mag. 44 (2023) 185–199.
[41] B. Liu, E. Robertson, S. Grigsby, S. Mazumder, Self-initiated open world learning for autonomous ai agents, in: Proceedings of AAAI Symposium on 'Designing Artificial Intelligence for Open Worlds', 2022.
[42] Y. Liu, S. Parisot, G. Slabaugh, X. Jia, A. Leonardis, T. Tuytelaars, More classifiers, less forgetting: a generic multi-classifier paradigm for incremental learning, in: ECCV, Springer International Publishing, 2020, pp. 699–716.
[43] Y. Liu, Y. Su, A.A. Liu, B. Schiele, Q. Sun, Mnemonics training: multi-class incremental learning without forgetting, in: CVPR, 2020.
[44] D. Lopez-Paz, M. Ranzato, Gradient episodic memory for continual learning, in: NeurIPS, 2017, pp. 6470–6479.
[45] I. Loshchilov, F. Hutter, Sgdr: stochastic gradient descent with warm restarts, arXiv preprint, arXiv:1608.03983, 2016.
[46] B. Loyall, A. Pfeffer, J. Niehaus, T. Mayer, P. Rizzo, A. Gee, S. Cvijic, W. Manning, M.K. Skitka, M. Becker, et al., An integrated architecture for online adaptation to novelty in open worlds using probabilistic programming and novelty-aware planning, in: Proceedings of the AAAI Spring Symposium on Designing AI for Open-World Novelty, 2022.
[47] M. McCloskey, N.J. Cohen, Catastrophic interference in connectionist networks: the sequential learning problem, in: Psychology of Learning and Motivation, vol. 24, Elsevier, 1989, pp. 109–165.
[48] O. Ostapenko, M. Puscas, T. Klein, P. Jahnichen, M. Nabi, Learning to remember: a synaptic plasticity driven framework for continual learning, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2019, pp. 11321–11329.
[49] J. von Oswald, C. Henning, J. Sacramento, B.F. Grewe, Continual learning with hypernetworks, in: ICLR, 2020.
[50] M. Palash, B. Bhargava, Continuous learning based novelty aware emotion recognition system, in: AAAI 2022 Spring Symposium on Designing Artificial Intelligence for Open Worlds, 2022.
[51] G. Pang, C. Shen, L. Cao, A.V.D. Hengel, Deep learning for anomaly detection: a review, ACM Comput. Surv. 54 (2021) 1–38.
[52] J. Parmar, S.S. Chouhan, V. Raychoudhury, S.S. Rathore, Open-world machine learning: applications, challenges, and opportunities, arXiv preprint, arXiv: 2105.13448, 2021.
[53] A. Pentina, C. Lampert, A pac-Bayesian bound for lifelong learning, in: International Conference on Machine Learning, PMLR, 2014, pp. 991–999.

[54] J. Rajasegaran, S. Khan, M. Hayat, F.S. Khan, M. Shah, itaml: an incremental task-agnostic meta-learning approach, in: CVPR, 2020.
[55] V. Ramanujan, M. Wortsman, A. Kembhavi, A. Farhadi, M. Rastegari, What's hidden in a randomly weighted neural network?, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020, pp. 11893–11902.
[56] S.A. Rebuffi, A. Kolesnikov, C.H. Lampert, iCaRL: incremental classifier and representation learning, in: CVPR, 2017, pp. 5533–5542.
[57] A. Rios, N. Ahuja, I. Ndiour, U. Genc, L. Itti, O. Tickoo, incdfm: incremental deep feature modeling for continual novelty detection, in: European Conference on Computer Vision, Springer, 2022, pp. 588–604.
[58] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al., Imagenet large scale visual recognition challenge, Int. J. Comput. Vis. 115 (2015) 211–252.
[59] W.J. Scheirer, L.P. Jain, T.E. Boult, Probability models for open set recognition, IEEE Trans. Pattern Anal. Mach. Intell. 36 (2014) 2317–2324.
[60] B. Schölkopf, R.C. Williamson, A.J. Smola, J. Shawe-Taylor, J.C. Platt, et al., Support vector method for novelty detection, in: NIPS, Citeseer, 1999, pp. 582–588.
[61] J. Serra, D. Suris, M. Miron, A. Karatzoglou, Overcoming catastrophic forgetting with hard attention to the task, in: International Conference on Machine Learning, PMLR, 2018, pp. 4548–4557.
[62] H. Shin, J.K. Lee, J. Kim, J. Kim, Continual learning with deep generative replay, in: NIPS, 2017, pp. 2994–3003.
[63] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, A. Rabinovich, Going deeper with convolutions, in: CVPR, 2015, http://arxiv.org/abs/1409.4842.
[64] J. Tack, S. Mo, J. Jeong, J. Shin, Csi: novelty detection via contrastive learning on distributionally shifted instances, in: NeurIPS, 2020.
[65] T. Thai, M. Shen, N. Varshney, S. Gopalakrishnan, U. Soni, C. Baral, M. Scheutz, J. Sinapov, An architecture for novelty handling in a multi-agent stochastic environment: case study in open-world monopoly, in: Designing Artificial Intelligence for Open Worlds: Papers from the 2022 Spring Symposium, Virtual, AAAI Press, 2022.
[66] H. Touvron, M. Cord, M. Douze, F. Massa, A. Sablayrolles, H. Jégou, Training data-efficient image transformers & distillation through attention, in: International Conference on Machine Learning, PMLR, 2021, pp. 10347–10357.
[67] G.M. Van De Ven, Z. Li, A.S. Tolias, Class-incremental learning with generative classifiers, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2021, pp. 3611–3620.
[68] G.M. van de Ven, A.S. Tolias, Three scenarios for continual learning, arXiv preprint, arXiv:1904.07734, 2019.
[69] M. Wortsman, V. Ramanujan, R. Liu, A. Kembhavi, M. Rastegari, J. Yosinski, A. Farhadi, Supermasks in superposition, in: NeurIPS, 2020.
[70] Y. Wu, Y. Chen, L. Wang, Y. Ye, Z. Liu, Y. Guo, Y. Fu, Large scale incremental learning, in: CVPR, 2019.
[71] H. Xu, B. Liu, L. Shu, P. Yu, Open-world learning and application to product classification, in: The World Wide Web Conference, 2019, pp. 3413–3419.
[72] S. Yan, J. Xie, X. He, Der: dynamically expandable representation for class incremental learning, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2021, pp. 3014–3023.
[73] J. Yang, K. Zhou, Y. Li, Z. Liu, Generalized out-of-distribution detection: a survey, arXiv preprint, arXiv:2110.11334, 2021.
[74] Y. You, I. Gitman, B. Ginsburg, Large batch training of convolutional networks, arXiv preprint, arXiv:1708.03888, 2017.
[75] G. Zeng, Y. Chen, B. Cui, S. Yu, Continuous learning of context-dependent processing in neural networks, Nat. Mach. Intell. (2019).
[76] F. Zenke, B. Poole, S. Ganguli, Continual learning through synaptic intelligence, in: ICML, 2017, pp. 3987–3995.
[77] F. Zhu, X.Y. Zhang, C. Wang, F. Yin, C.L. Liu, Prototype augmentation and self-supervision for incremental learning, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2021, pp. 5871–5880.