Learnability and Algorithm for Continual Learning

Gyuhak Kim [*1] **Changnan Xiao** [*2] **Tatsuya Konishi** [3] **Bing Liu** [1]

Abstract

This paper studies the challenging *continual learning* (CL) setting of *Class Incremental Learning* (CIL). CIL learns a sequence of tasks consisting of disjoint sets of concepts or classes. At any time, a single model is built that can be applied to predict/classify test instances of any classes learned thus far without providing any task related information for each test instance. Although many techniques have been proposed for CIL, they are mostly empirical. It has been shown recently that a strong CIL system needs a strong within-task prediction (WP) and a strong out-of-distribution (OOD) detection for each task. However, it is still not known whether CIL is actually learnable. This paper shows that CIL is learnable. Based on the theory, a new CIL algorithm is also proposed. Experimental results demonstrate its effectiveness.

1. Introduction

Learning a sequence of tasks incrementally, called *continual learning*, has attracted a great deal of attention recently (Chen & Liu, 2018). In the supervised learning context, each task consists of a set of concepts or classes to be learned. It is assumed that all tasks are learned in one neural network, which results in the key challenge of *catastrophic forgetting* (CF) because when learning a new task, the system has to modify the network parameters learned from old tasks in order to learn the new task, which may cause performance degradation for the old tasks (McCloskey & Cohen, 1989). Two continual learning settings have been popularly studied: *task incremental learning* (TIL) (van de Ven & Tolias, 2019) and *class incremental learning* (CIL).

In TIL, each task is an independent classification problem and has a separate model (the tasks may overlap). At test time, the task-id of each test instance is provided to locate the task-specific model to classify the test instance.

Definition 1.1 (Task Incremental Learning (TIL))**.** TIL learns a sequence of tasks, $1, 2, ..., T$. The training set of task k is $\mathcal{D}_k = \{((x_k^i, k), y_k^i)_{i=1}^{n_k}\}$, where n_k is the number of samples in task $k \in \mathcal{T} = \{1, 2, ..., T\}$, and $x_k^i \in \mathcal{X}$ is an input sample and $y_k^i \in Y_k \subseteq \mathcal{Y}$ $(= \bigcup_{k=1}^{T} Y_k)$ is its label. A TIL system learns a function $f : \mathcal{X} \times \mathcal{T} \rightarrow \mathcal{Y}$ to assign a class label $y \in Y_k$ to (x, k) (a test instance x from task k).

For CIL, a single model is built for all tasks/classes learned thus far (the classes in each task are distinct). At test time, no task-id is provided for a test instance.

Definition 1.2 (Class Incremental Learning (CIL))**.** CIL learns a sequence of tasks, $1, 2, ..., T$. The training set of task k is $\mathcal{D}_k = \{(x_k^i, y_k^i)_{i=1}^{n_k}\}$, where n_k is the number of samples in task $k \in \mathcal{T} = \{1, 2, ..., T\}$, and $x_k^i \in \mathcal{X}$ is an input sample and $y_k^i \in Y_k \subset \mathcal{Y}$ $(= \bigcup_{k=1}^{T} Y_k)$ is its label. All Y_k's are disjoint ($Y_k \cap Y_{k'} = \emptyset$, $\forall k \neq k'$). A CIL system learns a function (predictor or classifier) $f : \mathcal{X} \rightarrow \mathcal{Y}$ that assigns a class label y to a test instance x.

CIL is a more challenging setting because in addition to CF, it has the *inter-task class separation* (ICS) (Kim et al., 2022b) problem. ICS refers to the situation that since the learner has no access to the training data of the old tasks when learning a new task, then the learner has no way to establish the decision boundaries between the classes of the old tasks and the classes of the new task, which results in poor classification accuracy. Kim et al. (2022b) showed that a good within-task prediction (WP) and a good *out-of-distribution* (OOD) detection for each task are *necessary* and *sufficient* conditions for a strong CIL model.

Definition 1.3 (out-of-distribution (OOD) detection)**.** Given the training set $\mathcal{D} = \{(x^i, y^i)_{i=1}^{n}\}$, where n is the number of data samples, $x^i \in \mathcal{X}$ is an input sample and $y^i \in \mathcal{Y}$ is its class label. $\mathcal{Y}$ is the set of all classes in $\mathcal{D}$ and called the *in-distribution* (IND) classes. Our goal is to learn a function $f : \mathcal{X} \rightarrow \mathcal{Y} \cup \{O\}$ that can detect test instances that do not belong to any classes in $\mathcal{Y}$ (OOD)), which are assigned to class O, denoting *out-of-distribution* (OOD).

The intuition of the theory in (Kim et al., 2022b) is that if OOD detection is perfect for each task, then a test instance will be assigned to the correct task model to which the test

[*]Equal contribution [1]Department of Computer Science, University of Illinois at Chicago. [2]Work done at ByteDance. [3]KDDI Research (work done when this author was visiting Bing Liu's group). Correspondence to: Bing Liu <liub@uic.edu>.

Proceedings of the 40^{th} International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023.

instance belongs for classification, i.e., within-task prediction (WP). However, (Kim et al., 2022b) does not prove that CIL is learnable. To our knowledge, no existing work has reported a learnability study for CIL (see Sec. 2). This paper performs the **CIL learnability** study.

The proposed learnability proof requires two assumptions: **(1)** OOD detection is learnable. Fortunately, this has been proven in a recent paper (Fang et al., 2022). **(2)** There is a mechanism that can completely overcome forgetting (CF) for the model of each task. Again, fortunately, there are many existing TIL methods that can eliminate forgetting, e.g., parameter-isolation methods such as HAT (Serrà et al., 2018) and SupSup (Wortsman et al., 2020), which work by learning a sub-network in a shared network for each task. The sub-networks of all old tasks are protected when training a new task. Orthogonal projection methods such as PCP (Kim & Liu, 2020) and CUBER (Lin et al., 2022) can also overcome forgetting in the TIL setting.

CIL can be solved by a combination of **[a]** a *TIL method* that is able to protect each task model with no CF, and **[b]** a *normal supervised learning method for WP* and **[c]** an *OOD detection* method. **[b]** and **[c]** can be easily combined either **(i)** with an OOD detection model since it also learns the IND classes (see **Definition** 1.3) or **(ii)** a WP model that can also perform OOD detection. That is, for CIL, we simply replace the classification model built for each task in HAT/SupSup with a combined WP and OOD detection model.

Based on the theory, we propose a new replay-based CIL method that uses the combination of **[a]** and **(ii)** (two separate heads for each task, one for WP and the other for OOD detection based on the same feature extractor). This paper thus makes two main contributions:

(1). It performs **the first learnability study of CIL**. To the best of knowledge, no such a study has been reported so far.

(2). Based on the theory, **a new CIL method**, called ROW (*Replay, OOD, and WP* for CIL), is proposed. Experimental results show that it outperforms existing strong baselines.

It is interesting to note that our theory, including our earlier work in (Kim et al., 2022b), in fact, unifies OOD detection and continual learning as it covers both (Kim et al., 2023). Additionally, the theory is also applicable to ***open world learning*** because OOD detection and class incremental learning are two critical components of an open world learning system (Liu et al., 2023).

2. Related Work

To our knowledge, we are not aware of any paper that studies the learnability of CIL. Below, we survey the existing CL literature on both the theoretical and empirical sides.

On the **theoretical side**. Pentina & Lampert (2014) proposes a PAC-Bayesian framework to provide a learning bound on expected error by the average loss on the observed tasks. However, this work is not about CIL but about TIL. It focuses on knowledge transfer and assumes that all the tasks have the same input space and the same label space and the tasks are very similar. However, in CIL, every task has a distinct set of class labels. Furthermore, this work is not concerned with CF as earlier research in lifelong learning builds a separate model for each task. Lee et al. (2021) studied the generalization error by task similarity. It is again about TIL. Bennani et al. (2020) showed that a specific method called orthogonal gradient descent (OGD) gives a tighter generalization bound than SGD. As noted in Sec. 1, empirically, the CF problem for TIL has been solved (Serrà et al., 2018; Kim et al., 2022b). Several techniques have also been proposed to carry out knowledge transfer (Ke et al., 2020; 2021; Lin et al., 2022). Our work is entirely different as we study the learnability of CIL, which is a more challenging setting than TIL because of the additional difficulty of ICS (Kim et al., 2022b) in CIL. In this work, we are not concerned with knowledge transfer, which is mainly studied for the TIL setting. Recently, Kim et al. (2022b) showed that a good within-task prediction (WP) and a good OOD detection for each task are necessary and sufficient conditions for a strong CIL model. However, Kim et al. (2022b) did not show that CIL is learnable. This paper performs this study. It also proposes a new CIL algorithm.

On the **empirical side**, a large number of algorithms have been proposed. They belong to several families. **(1).** *Regularization-based methods* mitigate CF by restricting the learned parameters for old tasks from being updated significantly in a new task learning using regularizations (Kirkpatrick et al., 2017; Zenke et al., 2017) or knowledge distillation (Li & Hoiem, 2016; Zhu et al., 2021). Many existing approaches have used similar approaches (Ritter et al., 2018; Schwarz et al., 2018; Xu & Zhu, 2018; Castro et al., 2018; Dhar et al., 2019; Lee et al., 2019; Ahn et al., 2019; Liu et al., 2020a). **(2).** *Replay-based methods* alleviate CF by saving a small amount of training data from old tasks in a memory buffer and jointly train the model using the current data and the saved buffer data (Rusu et al., 2016; Lopez-Paz & Ranzato, 2017; Chaudhry et al., 2019a; Hou et al., 2019; Wu et al., 2019; Rolnick et al., 2019; Buzzega et al., 2020; Rajasegaran et al., 2020a; Liu et al., 2021; Cha et al., 2021; Yan et al., 2021; Wang et al., 2022b; Guo et al., 2022; Kim et al., 2022a). Some methods in this family also study which samples in memory should be used in replay (Aljundi et al., 2019) or which samples in the training data should be saved (Rebuffi et al., 2017; Liu et al., 2020b). **(3).** *Pseudo-replay methods* generate pseudo replay data for old tasks to serve as the replay data (Kamra et al., 2017; Shin et al., 2017; Wu et al., 2018; Seff et al., 2017; Kemker & Kanan,

2018; Hu et al., 2019; Rostami et al., 2019; Ostapenko et al., 2019). (Zhu et al., 2021) generates features instead of raw data. **(4).** *Parameter-isolation methods* train and protect a sub-network for each task (Mallya & Lazebnik, 2017; Abati et al., 2020; von Oswald et al., 2020; Rajasegaran et al., 2020b; Hung et al., 2019; Henning et al., 2021). Several systems, e.g., HAT (Serrà et al., 2018) and SupSup (Wortsman et al., 2020), have largely eliminated CF. A limitation is that the task-id of each test instance must be provided. These methods are thus mainly used for TIL. **(5).** *Orthogonal projection* methods learn each task in an orthogonal space to other tasks to reduce task interference or CF (Zeng et al., 2019; Kim & Liu, 2020; Chaudhry et al., 2020; Lin et al., 2022).

Our empirical part of the work is related to but also very different from the above methods. We use the replay data as OOD training data to fine-tune an OOD detection head for each task based on the features learned for the WP head and uses the TIL method HAT to overcome CF. Some existing methods have used a TIL method for CIL with an additional task-id prediction technique. iTAML (Rajasegaran et al., 2020b)'s task-id prediction needs the test data come in batches and each batch must be from the same task, which is unrealistic as the test sample usually comes one after another. CCG (Abati et al., 2020), Expert Gate (Aljundi et al., 2017), HyperNet (von Oswald et al., 2020) and PR-Ent (Henning et al., 2021) either build a separate network or use entropy to predict the task-id. LMC (Ostapenko et al., 2021) learns task specific knowledge via local modules capable of task-id prediction. However, they all perform poorly because none of the systems deal with the ICS problem, which is the key and is what our OOD detection is trying to tackle. In this line of work, the most closely related work to ours is MORE (Kim et al., 2022a), which builds a model for each task treating the replay data as OOD data. However, in inference, it considers only the IND classes of each task, but not OOD detector. Our method is more principled and outperforms MORE. The methods in (Kim et al., 2022b) do not use replay data and perform worse.

3. Learnability of CIL

Before going to the learnability analysis, we first describe the intuition behind. Kim et al. (2022b) showed that given a test sample, the CIL prediction probability for each class in a task is the product of two prediction probabilities: *within-task prediction* (WP) and *task-id prediction* (TP),

$$\mathbf{P}(X_{k,j}|x) = \mathbf{P}(X_{k,j}|x,k)\mathbf{P}(X_k|x), \tag{1}$$

where $X_{k,j}$ is the domain of task k and class j of the task and x is an instance. The first probability on the right-hand-side (RHS) is WP and the second probability on the RHS is TP. However, as mentioned earlier, Kim et al. (2022b) did not study whether CIL is learnable. We also note that (Kim et al., 2022b) proved that TP and OOD are correlated and only differ by a constant factor. Based on the definition of OOD detection (**Definition** 1.3), the OOD detection model can also perform WP. In the recent work in (Fang et al., 2022), it is proven that OOD detection is learnable.

Table 1. Notations used in Sec. 3.

Notation	Description
$\mathcal{X}$	feature space
$\mathcal{Y}$	label space
$\mathcal{H}$	hypothesis space
X_k	random variable in $\mathcal{X}$ of task k
Y_k	random variable in $\mathcal{Y}$ of task k
$D_{(X_k,Y_k)}$	distribution of task k
l	loss function
h	hypothesis function in $\mathcal{H}$
$\mathbf{R}_{D_{(X,Y)}}$	risk function, expectation of loss function on $D_{(X,Y)}$
$\mathcal{D}$	set of all distributions
S	set of samples
$D_{[1:k]}$	weighted mixture of the first k distributions
π_k	mixture weight
D_k	equivalent to $D_{[k:k]}$ and $D_{(X_k,Y_k)}$
$S\|_{[k1:k2]}$	set of support samples for $D_{[k1:k2]}$
$S\|_k$	equivalent to $S\|_{[k:k]}$
$\mathbf{A}_k$	algorithm after training the k-th task
$z_k^{i,j}$	score function of the j-th class of the i-th task for task k
O	distribution of OOD
α	constant in $[0,1)$
D^α	mixture of D and O with weight α
z_k^o	score for the OOD class
$\emptyset$	empty set
ϵ_n	error rate with total number of samples n

We show that if the learning of each task does not cause catastrophic forgetting (CF) for previous tasks, then CIL is learnable.[1] Fortunately, CF can be prevented for each task as several existing *task incremental learning* (TIL) methods including but not limited to HAT (Serrà et al., 2018) and SupSup (Wortsman et al., 2020) in the parameter-isolation family and PCP (Kim & Liu, 2020) and CUBER (Lin et al., 2022) in the orthogonal projection family can ensure no CF (Kim et al., 2022b). HAT/SupSup basically trains a sub-network as the model for each task. In learning each new task, all the sub-networks for the previous tasks are protected with masks so that their parameters will not be modified in backpropagation. Thus, in this section, we assume that all tasks are learned without *catastrophic forgetting* (CF).

We now discuss the learnability of class incremental learning (CIL). The notations for the following discussion are described in Tab. 1. Let $\mathcal{X}$ be a feature space, $\mathcal{Y}$ a label space, and $\mathcal{H}$ a hypothesis function space. Assume $\mathcal{H}$ is a *ring*, because we construct hypothesis functions by addition and multiplication in the proof of **Theorem** 3.3 and **Theorem** 3.7. We use $D_{(X_k,Y_k)}$ to denote the distribution

[1]The current learnability result only applies to offline CIL, but not to online CL, where the task boundary may be blurry.

of task k. $X_k \in \mathcal{X}$ and $Y_k \in \mathcal{Y}$ are random variables following $D_{(X_k,Y_k)}$. $\mathcal{D} = \{D_{(X,Y)}\}$ denotes the set of all distributions. $l(y_1, y_2) \geq 0$ denote a loss function. Denote $h \in \mathcal{H}$ as a hypothesis function. For any $X \in \mathcal{X}$, $Y \in \mathcal{Y}$, the risk function $\mathbf{R}_{D_{(X,Y)}}(h) \overset{def}{=} \mathbb{E}_{(x,y)\sim D_{(X,Y)}}[l(h(x), y)]$. $S \overset{def}{=} \{(x, y) \sim D_{(X,Y)}\}$ is sampled from $D_{(X,Y)}$, denoted as $S \sim D_{(X,Y)}$.

For a series of distributions $D_{(X_1,Y_1)}, \ldots, D_{(X_T,Y_T)}$, we denote the mixture of the first k distributions as $D_{[1:k]} = \frac{\sum_{i=1}^{k} \pi_i D_{(X_i,Y_i)}}{\sum_{i=1}^{k} \pi_i}$, where the mixture weights $\pi_1, \ldots, \pi_T > 0$ with $\sum_k \pi_k = 1$. For brevity, $D_k = D_{[k:k]} = D_{(X_k,Y_k)}$.

Denote $S|_{[k1:k2]} \overset{def}{=} \{s \in S | s \in supp\, D_{[k1:k2]}\}$. For simplicity, $S|_k = S|_{[k:k]}$.

Since continual learning tasks come one by one sequentially, we denote the hypothesis function that is found by an algorithm $\mathbf{A}$ after training the k-th task as $\mathbf{A}_k(S)$ with $S \sim D_{[1:k]}$. Strictly speaking, $h_k(x) = \mathbf{A}_k(S)(x)$ is only well-defined for $(x, y) \sim D_{[1:k]}$, and is not well-defined for $(x', y') \sim D_{k'}$, $k' > k$. Even if some implementation may predict a real value by $h_k(x')$, we regard it as non-sense at time k and only make sense until time k'.

For the risk function, we will meet $\mathbf{R}_{D_{[k_1:k_2]}}(h_k)$ and we guarantee that $k_1 \leq k_2 \leq k$. Denote

$$h_k = \arg\max_{1\leq i\leq k, j\in\{1,\ldots\}} \{\ldots, z_k^{i,j}, \ldots\},$$

where $z_k^{i,j}$ is the score function of the j-th class of the i-th task. The score function is any function that indicates which class the sample belongs to. For example, the score function could be the predicted logit of each class for a classification algorithm. We calculate

$$\mathbf{R}_{D_{[k_1:k_2]}}(h_k) = \mathbb{E}_{(x,y)\sim D_{[k_1:k_2]}} [l(\arg\max_{k_1\leq i\leq k_2, j\in\{1,\ldots\}} \{\ldots, z_k^{i,j}(x), \ldots\}, y)].$$

When we write $\mathbf{R}_{D^\alpha}(h)$ with $D^\alpha = (1-\alpha)D + \alpha O$ (where O denotes OOD and $\alpha \in [0, 1)$), we require h to predict one additional OOD class as

$$h_k = \arg\max\{\ldots, z_k^{i,j}, \ldots; z_k^o\},$$

where z_k^o is the score function of the OOD class.

Definition 3.1 (Fully-Observable Separated-Task Closed–World Learnability)**.** Given a set of distributions $\mathcal{D}$, a hypothesis function space $\mathcal{H}$, we say CIL is learnable if there exists an algorithm $\mathbf{A}$ and a sequence $\{\epsilon_n | \lim_{n\to+\infty} \epsilon_n = 0\}$ s.t. **(i)** for any $D_1, \ldots, D_T \in \mathcal{D}$ with $supp\, D_k \cap supp\, D_{k'} = \emptyset, k \neq k'$, and **(ii)** for any $\pi_1, \ldots, \pi_T > 0$ with $\sum_k \pi_k = 1$,

$$\max_{k=1,\ldots,T} \mathbb{E}_{S\sim D_{[1:k]}} [\mathbf{R}_{D_{[1:k]}}(\mathbf{A}_k(S)) - \inf_{h\in\mathcal{H}} \mathbf{R}_{D_{[1:k]}}(h)] < \epsilon_n.$$

We use ϵ to represent the error rate, where the index n of ϵ_n represents the total number of samples. The equation $\lim_{n\to+\infty} \epsilon_n = 0$ means that the error rate decreases to 0 as n goes to $+\infty$. **Definition** 3.1, the risk function is calculated over $D_{[1:k]}$ at task k, which means the data of all the past tasks and the current task are observable for optimization. It is a desirable property for CIL to take expectation over $D_{[1:k]}$ as it constructs a model that is equivalent to the model built with the full training data of all tasks. Generally, when an algorithm satisfies **Definition** 3.1, the system is already learnable because this is just the traditional supervised learning which can see/observe all the training data of all tasks and there is no OOD data involved (which means the *closed-world*). However, when we apply the algorithm $\mathbf{A}$ to solve for $\mathbf{A}(S)$ in practice, we usually cannot access all samples in S, which is partially-observable instead of fully-observable. That is the case for continual learning as it assumes that in learning the new/current task, the training data of the previous/past tasks is not accessible, at least a major part of it.

Due to the lack of full observations, we have to define $\mathbf{A}_k^r$ recursively. For any $S \sim D_{[1:k]}$, we define

$$\mathbf{A}_k^r(S) = \mathbf{A}_k^r(S|_k, \mathbf{A}_{k-1}^r(S|_{[1:k-1]})). \tag{2}$$

The algorithm depends on implementation. In the following discussion, we assume that learning a new task does not interrupt the error bound of previous tasks. This is a valid assumption as existing algorithms (Serrà et al., 2018; Wortsman et al., 2020) achieve little or no forgetting. The version of **Definition** 3.1 for partial observations is as follows.

Definition 3.2 (Partially-Observable Separated-Task Closed–World Learnability)**.** Given a set of distributions $\mathcal{D}$, a hypothesis function space $\mathcal{H}$, we say CIL is learnable if there exists an algorithm $\mathbf{A}$ and a sequence $\{\epsilon_n | \lim_{n\to+\infty} \epsilon_n = 0\}$ s.t. **(i)** for any $D_1, \ldots, D_T \in \mathcal{D}$ with $supp\, D_k \cap supp\, D_{k'} = \emptyset, k \neq k'$, **(ii)** for any $\pi_1, \ldots, \pi_T > 0$ with $\sum_k \pi_k = 1$,

$$\max_{k=1,\ldots,T} \mathbb{E}_{S\sim D_{[1:k]}} [\mathbf{R}_{D_k}(\mathbf{A}_k^r(S)) - \inf_{h\in\mathcal{H}} \mathbf{R}_{D_k}(h)] < \epsilon_n.$$

In **Definition** 3.4, the risk function is calculated over D_k alone as only the current task data D_k is observable while the past tasks are not. It is desirable that **Definition** 3.2 implies **Definition** 3.1, which transforms the learnability of a CIL problem into the learnability of a supervised problem. Unfortunately, **Definition** 3.2 does not imply **Definition** 3.1. **Theorem** 3.3 shows that there exists a trivial hypothesis function that satisfies **Definition** 3.2 but doesn't satisfy **Definition** 3.1.

Theorem 3.3 (**Definition** 3.2 does not imply **Definition** 3.1)**.** *For a set of distributions $\mathcal{D}$ and a hypothesis function space $\mathcal{H}$, if **Definition** 3.2 holds and $\mathcal{H}$ has the capacity to learn*

more than one task, then there exists $h \in \mathcal{H}$ s.t. ***Definition*** *3.2 holds but* ***Definition*** *3.1 doesn't hold.*

The proof is given in Appendix A. The main reason here is that only the samples of the current task are observable, while samples of both past and future tasks are hard to be exploited. From the perspective of forward looking, when training the current task, we have no access to any information of future tasks, where samples of future tasks are regarded as *out-of-distribution* (OOD) samples with respect to the current and past tasks. Inspired by **Theorem** 3.3, we include OOD detection into consideration and generalize **Definition** 3.1 to the open-world setting.

Definition 3.4 (Fully-Observable Separated-Task Open–World Learnability)**.** Given a set of distributions $\mathcal{D}$, a hypothesis function space $\mathcal{H}$, we say CIL is learnable if there exists an algorithm $\mathbf{A}$ and a sequence $\{\epsilon_n | \lim_{n\to+\infty} \epsilon_n = 0\}$ s.t. **(i)** for any $D_1, \ldots, D_T \in \mathcal{D}$ with $supp\, D_k \cap supp\, D_{k'} = \emptyset, k \neq k'$, **(ii)** for any $\pi_1, \ldots, \pi_T > 0$ with $\sum_k \pi_k = 1$, and **(iii)** for any $O_{(X_1,Y_1)}, \ldots, O_{(X_T,Y_T)} \in \mathcal{D}$, any $\alpha_1, \ldots, \alpha_T \in [0, 1)$,

$$\max_{k=1,\ldots,T} \mathbb{E}_{S\sim D_{[1:k]}}[\mathbf{R}_{D_{[1:k]}^{\alpha_{[1:k]}}}(\mathbf{A}_k(S)) - \inf_{h\in\mathcal{H}} \mathbf{R}_{D_{[1:k]}^{\alpha_{[1:k]}}}(h)] < \epsilon_n,$$

where $D_{[1:k]}^{\alpha_{[1:k]}} = \sum_{i=1}^{k}(1-\alpha_i)D_i + \alpha_i O_{(X_i,Y_i)}$.

The proof of **Definition** 3.4 is guaranteed by previous work (Fang et al., 2022), which studies the learnablity of OOD detection. It's obvious that when **Definition** 3.4 is satisfied, **Definition** 3.1 is satisfied, which is shown in **Theorem** 3.5.

Theorem 3.5 (**Definition** 3.4 implies **Definition** 3.1)**.** *For a set of distributions $\mathcal{D}$ and a hypothesis function space $\mathcal{H}$, if* ***Definition*** *3.4 holds, then* ***Definition*** *3.1 holds.*

The proof is given in Appendix A. When we have no access to samples of past tasks in practice, we define $\mathbf{A}_k$ recursively as in Eq. 2. The partial observable version of **Definition** 3.4 is stated below. In **Definition** 3.6, the risk function is over D_k instead of $D_{[1:k]}$ because it's the partial observable case.

Definition 3.6 (Partially-Observable Separated-Task Open–World Learnability)**.** Given a set of distributions $\mathcal{D}$, a hypothesis function space $\mathcal{H}$, we say CIL is learnable if there exists an algorithm $\mathbf{A}$ and a sequence $\{\epsilon_n | \lim_{n\to+\infty} \epsilon_n = 0\}$ s.t. **(i)** for any $D_1, \ldots, D_T \in \mathcal{D}$ with $supp\, D_k \cap supp\, D_{k'} = \emptyset, k \neq k'$, **(ii)** for any $\pi_1, \ldots, \pi_T > 0$ with $\sum_k \pi_k = 1$, and **(iii)** for any $O_{(X_1,Y_1)}, \ldots, O_{(X_T,Y_T)} \in \mathcal{D}$, any $\alpha_1, \ldots, \alpha_T \in [0, 1)$,

$$\max_{k=1,\ldots,T} \mathbb{E}_{S\sim D_{[1:k]}}[\mathbf{R}_{D_k^{\alpha_k}}(\mathbf{A}_k^r(S)) - \inf_{h\in\mathcal{H}} \mathbf{R}_{D_k^{\alpha_k}}(h)] < \epsilon_n,$$

where $D_k^{\alpha_k} = (1-\alpha_k)D_k + \alpha_k O_{(X_k,Y_k)}$.

Note that Fang et al. (2022) showed that OOD detection is learnable under a compatibility condition for a single OOD detection problem and **Definition** 3.6 is about learnability with respect to an ensemble of multiple OOD detection problems. It is obvious that once each OOD detection problem is learnable, the ensemble of them is also learnable. With this definition, we derive that CIL is learnable as OOD detection is learnable. Different from **Theorem** 3.3 that partially-observable learnability does not imply fully-observable learnability for the closed-world setting, **Theorem** 3.7 shows that the learnability of a CIL system can be converted to a series of OOD learnability problems for the open-world setting (meaning there are OOD data).

Theorem 3.7 (**Definition** 3.6 implies **Definition** 3.4)**.** *For a set of distributions $\mathcal{D}$ and a hypothesis function space $\mathcal{H}$, if* ***Definition*** *3.6 holds, $\mathcal{H}$ enjoys enough capacity i.e. $\inf_{h\in\mathcal{H}} \mathbf{R}_{D_{[1:k]}^{\alpha_{[1:k]}}}(h) = 0$, and the loss function on all tasks is bounded by summation of loss function on each task i.e., Eq. 8 in Appendix, then* ***Definition*** *3.4 holds and the upper bound ϵ_n is multiplied by $\max_{k=1,\ldots,T} \sum_{t=1}^{k} \frac{\pi_{[t:T]}}{\pi_{[1:k]}}$.*

The proof is given in Appendix A. **Theorem** 3.7 connects **Definitions** 3.6 to 3.4 and **Theorem** 3.5 connects **Definitions** 3.4 to 3.1, which is the desirable property of CIL. When all tasks have the same weight $\pi_1 = \cdots = \pi_T = 1/T$, the multiplier $\max_{k=1,\ldots,T} \sum_{t=1}^{k} \frac{\pi_{[t:T]}}{\pi_{[1:k]}} = T$, which is positively correlated with the number of tasks.

Though **Theorem** 3.7 gives an upper bound to induce **Definition** 3.4 from **Definition** 3.6, the hypothesis function that satisfies **Definition** 3.4 is recursively derived from the previous tasks (see the proof). We can also observe that when tasks have different weights, the multiplier $\max_{k=1,\ldots,T} \sum_{t=1}^{k} \frac{\pi_{[t:T]}}{\pi_{[1:k]}}$ depends on the order of tasks. It is undesirable that the hypothesis function depends on the order of tasks. When we can acquire some replay data of past tasks and treat them as OOD data, we have the following corollary that gives an order-free hypothesis function.

Corollary 3.8. *For a set of distributions $\mathcal{D}$ and a hypothesis function space $\mathcal{H}$, if* ***Definition*** *3.6 holds, $\mathcal{H}$ enjoys enough capacity i.e. $\inf_{h\in\mathcal{H}} \mathbf{R}_{D_{[1:k]}^{\alpha_{[1:k]}}}(h) = 0$, and the loss function on all tasks is bounded by summation of the loss functions on every task i.e. Eq. 10 in Appendix, if we treat data of past tasks as OOD data, then* ***Definition*** *3.4 holds and the upper bound ϵ_n is multiplied by $\max_{k=1,\ldots,T} \frac{k\pi_{[1:T]}}{\pi_{[1:k]}}$.*

The proof is given in Appendix A.

4. Proposed Method

The learnability in **Definition** 3.6 is defined over the OOD function of each task. By **Definition** 1.3 of OOD, an OOD function is capable of classification (i.e., WP) for IND in-

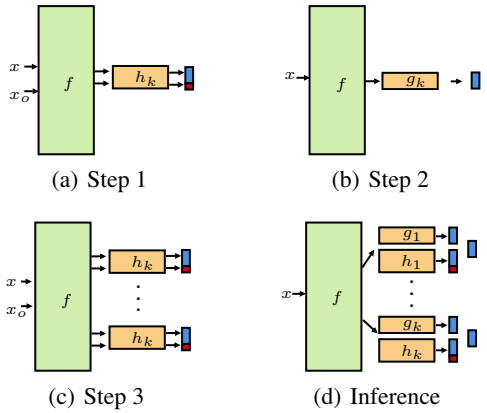
stances and rejection for OOD instances (or TP as it can be defined using OOD and such TP differs from OOD by a constant factor (Kim et al., 2022b)). As discussed early, we use the masks in HAT (Serrà et al., 2018) to protect each OOD model to ensure there is no forgetting. Following exactly this theoretical framework, an algorithm can be designed, which works quite well (see ROW (-WP) in Tab. 4). However, it is possible to do better by introducing a WP head so that we can use the OOD head for estimating only TP rather than for handling both WP and TP.

The proposed method ROW is a replay-based method. At each task k, the system receives dataset D_k and leverages the replay data saved from previous tasks in the replay memory buffer $\mathcal{M}$ as the OOD data of the task to train an OOD detection head and also to fine-tune the WP head. Specifically, the model of each task has two heads: one for OOD (for computing TP) and one for WP. That is, we optimize the set of parameters $(\Psi_k, \theta_k, \phi_k)$, where Ψ_k is the parameter set of the feature extractor f_k, θ_k is the parameter set of OOD head h_k, and ϕ_k is the parameter set of the WP head (i.e., classifier) g_k. The two task specific heads h_k and g_k receive feature u from the shared feature extractor f_k and produce WP and TP probabilities, respectively. The training consists of three steps: 1) training the feature extractor f_k and the OOD head h_k using both IND instances in D_k and OOD instances in $\mathcal{M}$ (i.e., the replay data), 2) fine-tuning a WP head g_k for the task using D_k based on only the fixed feature extractor f_k, and 3) fine-tuning the OOD heads of all tasks that have been learned so far. Training steps 2 and 3 are fast as both are simply fine-tuning the single layer of the classifiers (details below). The outputs from the two heads are used to compute the final CIL prediction probability in Eq. 1. An overview of the training and prediction process is illustrated in Fig. 1.

1) Training Feature Extractor and OOD Head. This step trains the OOD head h_k for task k. Its feature extractor f_k is also shared by the WP head (see below). An illustration of the training process is given in Fig. 1(a). Since OOD instances are any instances whose labels do not belong to task k, we leverage the task data D_k as IND instances and the saved replay instances of tasks $k' \neq k$ in the memory buffer $\mathcal{M}$ as OOD instances represented by an OOD class (in red) in the OOD head. The network $h_k \circ f_k$ is trained to maximize the probability $p(y|x,k) = \text{softmax} h_k(f(x,k;\Psi_k);\theta_k)_y$ for an IND instance $x \in D_k$ and maximize the probability $p(ood|x,k)$ for OOD instance $x \in \mathcal{M}$. Formally, this is achieved by minimizing the sum of cross-entropy losses

$$\mathcal{L}_{ood}(\Psi_t, \theta_k) = -\frac{1}{2N}\Big(\sum_{(x,y)\in D_k} \log p(y|x,k) + \sum_{(x,y)\in \mathcal{M}} \log p(ood|x,k)\Big), \tag{3}$$

where N is the number of instances in D_k. We utilize upsampling with the replay instances to achieve an equal number of samples as the current task data $\mathcal{D}_k$. The first loss is to discriminate the IND instances while the second loss is introduced to distinguish between IND and OOD instances.



(a) Step 1 (b) Step 2 (c) Step 3 (d) Inference

Figure 1. Overview of training steps at task k and inference. (a): the first step of training the feature extractor and OOD head for task k. The system receives both IND instance $x \in D_k$ and OOD instance $x_o \in \mathcal{M}$. The output has IND classes (in blue) and the OOD class or label (in red). (b): the second step of fine-tuning the WP head using the IND training data only. (c): fine-tuning all OOD heads using both IND and OOD instances. (d): inference/prediction. For a test instance x, obtain TP and WP probabilities, and compute the CIL probability as in Eq. 1.

To deal with forgetting, we use the HAT method (Serrà et al., 2018) (see Appendix B).

2) Fine-Tuning the WP Head. Given the feature extractor trained in the first step, we fix the feature extractor and fine-tune the WP head g_k (i.e., the WP classifier) using only D_k by adjusting the parameters ϕ_k. This is achieved by minimizing the cross-entropy loss

$$\mathcal{L}_{WP}(\phi_k) = -\frac{1}{N}\sum_{(x,y)\in D_k} \log p(y|x,k). \tag{4}$$

WP probabilities for the classes of task k are just the output softmax probabilities.

3) Fine-Tuning the OOD Heads of All Tasks. The OOD head h_k built in step 1) is biased because for early tasks, where the instances in $\mathcal{M}$ are less diverse, the OOD heads for them will be weaker than the OOD heads of later tasks when the instances in $\mathcal{M}$ are more diverse. To mitigate this bias, we fine-tune all OOD heads of all tasks after training each task using only the replay data in $\mathcal{M}$. After training task k, we have $\mathcal{M}$ with replay instances of classes from task 1 to k. For each task $k' \leq k$, reconstruct a new IND data $\tilde{D}_{k'}$ by selecting instances corresponding to task k' from $\mathcal{M}$, and a new pseudo memory buffer $\tilde{\mathcal{M}}$ after removing the instances in $\tilde{D}_{k'}$. We then fine-tune every OOD head by

minimizing the loss function

$$\mathcal{L}_{TP}(\theta_{k'}) - \frac{1}{M}\Big(\sum_{(x,y)\in\tilde{\mathcal{M}}} \log p(ood|x,k') + \sum_{(x,y)\in\tilde{D}_{k'}} \log p(y|x,k')\Big) \quad (5)$$

where M is $|\tilde{D}_k| + |\tilde{\mathcal{M}}|$. Although the TP probability can be defined simply using the fine-tuned OOD heads, it can be further improved, which we discuss next.

4.1. Distance-Based Coefficient

We can further improve the performance by incorporating a distance-based coefficient defined at the feature level into the output from the OOD head. The intuition is based on the observation that samples identified as OOD using a score function defined at the feature level are not recognized with a score function defined in the output level, and vice versa (Wang et al., 2022a). Their combination usually produces a better OOD detector.

After training task k, compute the means of the feature vectors per class of the task and the variance of the features. Denote the mean of class y by μ_y and the variance by $\Sigma_k = \sum_y \Sigma_y$, where Σ_y is the variance of features of class y. Using Mahalanobis distance (MD), the coefficient of an instance x for task k is

$$c_k(x) = \max_y 1/MD(x;\mu_y,\Sigma_k). \quad (6)$$

The coefficient is large if the feature of a test instance x is close to one of the sample means of the task and small otherwise.

We finally define the **TP probability** for task k as

$$\mathbf{P}(X_k|x) = c_k(x)\max_j \text{softmax}(h_k(x))_j/Z, \quad (7)$$

where Z is the normalizing factor and the maximum is taken over the softmax outputs of the IND classes j obtained by the OOD head h_k. The $\max_j$ operation can also be seen as the maximum softmax probability score (Hendrycks & Gimpel, 2016).

With the WP and TP probabilities, we now make a CIL prediction based on Eq. 1.

5. Empirical Evaluation

Baselines. We compare the proposed ROW[2] with 12 baselines. Five are exemplar-free (i.e., saving no previous data) methods and seven are replay-based methods. The exemplar-free methods are: **HAT** (Serrà et al., 2018), **OWM** (Zeng et al., 2019), **SLDA** (Hayes & Kanan, 2020), **PASS** (Zhu et al., 2021), and **L2P** (Wang et al., 2022c). For the multi-head method HAT, we make prediction by taking $\arg\max$ over the concatenated logits from each task model as (Kim et al., 2022b). The replay methods are: **iCaRL** (Rebuffi et al., 2017), **A-GEM** (Chaudhry et al., 2019a), **EEIL** (Castro et al., 2018), **GD** (Lee et al., 2019) without external data, **DER++** (Buzzega et al., 2020), **HAL** (Chaudhry et al., 2021), and **MORE** (Kim et al., 2022a).

We could not make the recent system in (Wu et al., 2022) using a pre-trained model as no code is released. We also do not include the existing parameter isolation methods that deal with CIL problems as they are very weak. HyperNet (von Oswald et al., 2020) and PR (Henning et al., 2021) find the task-id via an entropy function and SupSup (Wortsman et al., 2020) finds it via gradient update. They then make a within-task prediction. SupSup, PR, and iTAML (Rajasegaran et al., 2020b) assume the test instances come in batches and all samples in a batch belong to one task. When tested per sample, HyperNet, SupSup, PR and iTAML achieve 22.4, 11.8, 45.2 and 33.5 on 10 tasks of CIFAR100, respectively, which are much lower than 51.4 of iCaRL. CCG (Abati et al., 2020) has no code. The systems in (Kim et al., 2022b) are also not included because they are quite weak as their contrastive learning does not work well with a pre-trained model. The results reported in their paper based on ResNet-18 are also weaker than ROW.

Datasets. We use three popular continual learning benchmark datasets. **1). CIFAR10** (Krizhevsky & Hinton, 2009). This is an image classification dataset consisting of 60,000 color images of size 32x32, among which 50,000 are training data and 10,000 are testing data. It has 10 different classes. **2). CIFAR100** (Krizhevsky & Hinton, 2009). This dataset consists of 50,000 training images and 10,000 testing images with 100 classes. Each image is colored and of size 32x32. **3). Tiny-ImageNet** (Le & Yang, 2015). This dataset has 200 classes with 500 training images of size 64x64 per class. The validation data has 50 samples per class. Since no label is provided for the test data, we use the validation set for testing as in (Zhu et al., 2021).

Backbone Architecture and Pre-Training. We use the backbone architecture of transformer DeiT-S/16 (Touvron et al., 2021). As pre-training models or feature extractors are increasingly used in all types of applications, including continual learning (Wang et al., 2022c; Kim et al., 2022a; Wu et al., 2022), we also take this approach. Following (Kim et al., 2022a), to ensure there is no information leak from pre-training to continual learning, the pre-trained model/network is trained using 611 classes of ImageNet after removing 389 classes which are similar or identical to the classes of CIFAR and Tiny-ImageNet. To leverage the strong performance of the pre-trained model while adapting to new knowledge, we

[2] https://github.com/k-gyuhak/CLOOD

fix the feature extractor and append trainable **adapter modules** of fully-connected networks with one hidden layer at each transformer layer (Houlsby et al., 2019) except SLDA and L2P.[3] The number of neurons in each hidden layer is 64 for CIFAR10 and 128 for other datasets. Note that ***all baselines and ROW use the same architecture and the same pre-training model for fairness*** as using a pre-trained model improves the performance (Kim et al., 2022a; Ostapenko et al., 2022).

Note that we do not use the pre-trained models like CLIP (Radford et al., 2021) or others trained using the full ImageNet data due to **information leak** both in terms of features and class labels because our experiment data have been used in training these pre-trained models. This leakage can seriously affect the results. For example, the L2P system using the pre-training model trained using the full ImageNet data performs extremely well, but after those overlapping classes are removed in pre-training, its performances drop greatly. In Tab. 2, we can see that it is in fact quite weak.

Training Details. For the replay-based methods, we follow the budget sizes of (Rebuffi et al., 2017; Buzzega et al., 2020). For our method, we use the memory budget strategy (Chaudhry et al., 2019b) to save equal number of samples per class. Denote the budget size by $|\mathcal{M}|$.

For CIFAR10, we split the 10 classes into 5 tasks with 2 classes per task. We refer the experiment as C10-5T. The memory budget size $|\mathcal{M}|$ is 200.

For CIFAR100, we conduct two experiments. We split the 100 classes into 10 and 20 tasks, where each task has 10 classes and 5 classes, respectively. We refer the experiments as C100-10T and C100-20T. We choose $|\mathcal{M}| = 2{,}000$.

For Tiny-ImageNet, we conduct two experiments. We split the 200 classes into 5 tasks with 40 classes per task and 10 tasks with 20 classes per task. We refer the experiments as T-5T and T-10T, respectively. We save 2,000 samples in total for both experiments.

Following the random class order protocol of the existing methods (Rebuffi et al., 2017; Lee et al., 2019), we randomly generate 5 different class orders for each experiment and report the average accuracy over the 5 random orders.

For all the experiments of our system, we find a good set of learning rates and the number of epochs via validation data made of 10% of the training data. The hyper-parameters of our system is reported in Appendix C. For the baselines, we use the experiment setups as reported in their official papers. If we could not reproduce the result, we find the hyper-parameters via the validation set.

[3]For SLDA and L2P, we follow the original papers. SLDA fine-tunes only the classifier with a fixed feature extractor and L2P trains learnable prompts.

Table 2. Average classification accuracy after the final task. '-XT' means X number of tasks. Our system ROW and all baselines use the pre-trained network. The last 7 baselines are replay-based systems. The last column shows the average of each row. We highlight the best results in each column in bold.

Method	C10-5T	C100-10T	C100-20T	T-5T	T-10T	Average
HAT	79.36$_{\pm 5.16}$	68.99$_{\pm 0.21}$	61.83$_{\pm 0.62}$	**65.85**$_{\pm 0.60}$	62.05$_{\pm 0.55}$	67.62
OWM	41.69$_{\pm 6.34}$	21.39$_{\pm 3.18}$	16.98$_{\pm 4.44}$	24.55$_{\pm 2.48}$	17.52$_{\pm 3.45}$	24.43
SLDA	88.64$_{\pm 0.05}$	67.82$_{\pm 0.05}$	67.80$_{\pm 0.05}$	57.93$_{\pm 0.05}$	57.93$_{\pm 0.06}$	68.02
PASS	86.21$_{\pm 1.10}$	68.90$_{\pm 0.94}$	66.77$_{\pm 1.18}$	61.03$_{\pm 0.38}$	58.34$_{\pm 0.42}$	68.25
L2P	73.59$_{\pm 4.15}$	61.72$_{\pm 0.81}$	53.84$_{\pm 1.59}$	59.12$_{\pm 0.96}$	54.09$_{\pm 1.14}$	60.47
iCaRL	87.55$_{\pm 0.99}$	68.90$_{\pm 0.47}$	69.15$_{\pm 0.99}$	53.13$_{\pm 1.04}$	51.88$_{\pm 2.36}$	66.12
A-GEM	56.33$_{\pm 7.77}$	25.21$_{\pm 4.00}$	21.99$_{\pm 4.01}$	30.53$_{\pm 3.99}$	21.90$_{\pm 5.52}$	36.89
EEIL	82.34$_{\pm 3.13}$	68.08$_{\pm 0.51}$	63.79$_{\pm 0.66}$	53.34$_{\pm 0.54}$	50.38$_{\pm 0.97}$	63.59
GD	89.16$_{\pm 0.53}$	64.36$_{\pm 0.57}$	60.10$_{\pm 0.74}$	53.01$_{\pm 0.97}$	42.48$_{\pm 2.53}$	61.82
DER++	84.63$_{\pm 2.91}$	69.73$_{\pm 0.99}$	70.03$_{\pm 1.46}$	55.84$_{\pm 2.21}$	54.20$_{\pm 3.28}$	66.89
HAL	84.38$_{\pm 2.70}$	67.17$_{\pm 1.50}$	67.37$_{\pm 1.45}$	52.80$_{\pm 2.37}$	55.25$_{\pm 3.60}$	65.39
MORE	89.16$_{\pm 0.96}$	70.23$_{\pm 2.27}$	70.53$_{\pm 1.09}$	64.97$_{\pm 1.28}$	63.06$_{\pm 1.26}$	71.59
ROW	**90.97**$_{\pm 0.19}$	**74.72**$_{\pm 0.48}$	**74.60**$_{\pm 0.12}$	65.11$_{\pm 1.97}$	**63.21**$_{\pm 2.53}$	**73.72**

Evaluation Metrics. We use two metrics: average classification accuracy (ACA) and average forgetting rate. ACA after the last task t is $\mathcal{A}_t = \sum_{i=1}^{t} A_i^t/t$, where A_i is the accuracy of the model on task ith data after learning task t. The average forgetting rate after task t is $\mathcal{F}_t = \sum_{i=1}^{t-1} A_i^i - A_i^t$ (Liu et al., 2020b). This is also referred as backward transfer in other literature (Lopez-Paz & Ranzato, 2017).

5.1. Results and Comparison

Average Classification Accuracy. Tab. 2 shows the average classification accuracy after the final task. The last column Average indicates the average performance of each method over the 5 experiments. Our proposed method ROW performs the best consistently. On average, ROW achieves 73.72% while the best replay baseline (MORE) achieves 71.59%. We observe that MORE is significantly better than the other baselines. This is because MORE actually builds an OOD model for each task, which is close to the proposed theory but less principled than ROW.

The baselines SLDA and L2P are proposed to leverage a strong pre-trained feature extractor in the original papers. SLDA freezes the feature extractor and only fine-tunes the classifier. It performs well for the simple experiment C10-5T, but is significantly poorer than ROW on all experiments. This is because the fixed feature extractor does not adapt to new knowledge. Our method updates the feature extractor via adapter modules to new knowledge and it is able to learn more complex problems. L2P trains a set of prompt embeddings. In the original paper, it uses a feature extractor that was pre-trained with ImageNet-21k which already includes the classes of the continual learning (CL) evaluation datasets (information leak). After we remove the classes similar to the datasets used in CL, its performance drops dramatically (60.47% on average over the 5 experiments) and much poorer than our ROW (73.72% on average).

We conduct additional experiments with the size of mem-

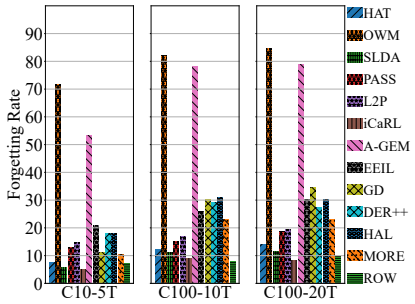


Figure 2. Average forgetting rate (CIL). The lower the rate, the better the method is.

ory buffer reduced by half to show the effectiveness of our method. The new memory buffer sizes for CIFAR10, CIFAR100, and Tiny-ImageNet are 100, 1,000, and 1,000, respectively. Tab. 3 shows that our method ROW experiences little reduction in performance whereas the other replay-based baselines suffer from significant performance reduction. On average over the 5 experiments, ROW achieves 72.70% while previously with the original memory buffer, it achieved 73.72. In contrast, the second best baseline DER++ reduces from 66.89 to 62.16. MORE is also robust with small memory sizes, but its average accuracy is 71.44 which is still lower than ROW.

Average Forgetting Rate. We compare the forgetting rate of each system after learning the last task in Fig. 2. The forgetting rates of the proposed method ROW are 7.05, 7.99, and 9.72 on C10-5T, C100-10T and C100-20T, respectively. iCaRL forgets less than ours on C10-5T and C100-20T as it achieves 4.95 and 8.31, respectively. However, iCaRL was not able to adapt to new knowledge effectively as its accuracies are much lower than ROW on the same experiments. The forgetting rate of SLDA on C10-5T 5.74, but a similar observation can be made as iCaRL. The average accuracy over the 5 experiments of ROW is 73.72 while that of iCaRL and SLDA are only 66.12 and 68.02, respectively. According to the forgetting rates, the best baseline (MORE) adapts to the new knowledge well, but it was not able to retain the knowledge as effectively as ROW. Its forgetting rates are 10.30, 22.96, and 22.90 on C10-5T, C100-10T, and C100-20T, respectively, and are much larger than ours. This results in lower performance of MORE than ROW.

It is **important to note** that our system actually has no forgetting due to the CF prevention by HAT. The '*forgetting*' occurs not because it forgets the task knowledge, but because the classification becomes harder with more classes.

5.2. Ablation Experiments

We conduct an ablation study to measure the performance after each component is removed from ROW. We consider removing two components: WP head and the distance-based coefficient (MD) in Sec. 4.1. The method without WP head (ROW (-WP)) simply uses the OOD head obtained from step 1) with Eq. 3. This method makes the final prediction by taking $\arg\max$ over the concatenated logit values without the OOD label from each task network (i.e. OOD head).

Tab. 4 shows the average classification accuracy. The model after removing WP also works greatly as it already outperforms most of the baselines on C10-5T and outperforms the baselines on C100-10T and 20T. In other words, using OOD head constructed following the theoretical framework is effective. The model is still functional after removing both components (WP and the distance-based coefficient by MD) as shown in the last row of the table (ROW (-WP-MD)).

Table 3. The classification accuracy of the replay-based baselines and our method ROW with smaller memory buffer sizes. The buffer sizes are reduced by half and the new sizes are: 100 for CIFAR10 and 1,000 for 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.
iCaRL	86.08$_{\pm 1.19}$	66.96$_{\pm 2.08}$	68.16$_{\pm 0.71}$	47.27$_{\pm 3.22}$	49.51$_{\pm 1.87}$	63.60
A-GEM	56.64$_{\pm 4.29}$	23.18$_{\pm 2.54}$	20.76$_{\pm 2.88}$	31.44$_{\pm 3.84}$	23.73$_{\pm 6.27}$	31.15
EEIL	77.44$_{\pm 3.04}$	62.95$_{\pm 0.68}$	57.86$_{\pm 0.74}$	48.36$_{\pm 1.38}$	44.59$_{\pm 1.72}$	58.24
GD	85.96$_{\pm 1.64}$	57.17$_{\pm 1.06}$	50.30$_{\pm 0.58}$	46.09$_{\pm 1.77}$	32.41$_{\pm 2.75}$	54.39
DER++	80.09$_{\pm 3.00}$	64.89$_{\pm 2.48}$	65.84$_{\pm 1.46}$	50.74$_{\pm 2.41}$	49.24$_{\pm 5.01}$	62.16
HAL	79.16$_{\pm 4.56}$	62.65$_{\pm 0.83}$	63.96$_{\pm 1.49}$	48.17$_{\pm 2.94}$	47.11$_{\pm 6.00}$	60.21
MORE	88.13$_{\pm 1.16}$	71.69$_{\pm 0.11}$	71.29$_{\pm 0.55}$	64.17$_{\pm 0.77}$	61.90$_{\pm 0.90}$	71.44
ROW	**89.70**$_{\pm 1.54}$	**73.63**$_{\pm 0.12}$	**71.86**$_{\pm 0.07}$	**65.42**$_{\pm 0.55}$	**62.87**$_{\pm 0.53}$	**72.70**

Table 4. Performance gains with the proposed techniques. The method -WP indicates removing WP head and using only OOD head obtained in step 1). The method -MD indicates removing the distance-based coefficient.

	C10-5T	C100-10T	C100-20T
ROW	90.97$_{\pm 0.19}$	74.72$_{\pm 0.48}$	74.60$_{\pm 0.12}$
ROW (-WP)	88.50$_{\pm 1.32}$	72.29$_{\pm 0.90}$	71.97$_{\pm 0.77}$
ROW (-WP-MD)	84.06$_{\pm 3.38}$	67.53$_{\pm 1.73}$	65.85$_{\pm 0.95}$

6. Conclusion

To the best of our knowledge, there is still no reported study of learnability of class incremental learning (CIL). This paper performed such a study and showed that the CIL is learnable with some practically reasonable assumptions. A new CIL algorithm (called ROW) was also proposed based on the theory. Experimental results demonstrated that ROW outperforms strong baselines.

Acknowledgements

The work of Gyuhak Kim and Bing Liu was supported in part by a research contract from KDDI and three NSF grants (IIS-1910424, IIS-1838770, and CNS-2225427).

References

Abati, D., Tomczak, J., Blankevoort, T., Calderara, S., Cucchiara, R., and Bejnordi, E. Conditional channel gated networks for task-aware continual learning. In *CVPR*, pp. 3931–3940, 2020.

Ahn, H., Cha, S., Lee, D., and Moon, T. Uncertainty-based continual learning with adaptive regularization. In *NeurIPS*, 2019.

Aljundi, R., Chakravarty, P., and Tuytelaars, T. Expert gate: Lifelong learning with a network of experts. In *CVPR*, 2017.

Aljundi, R., Belilovsky, E., Tuytelaars, T., Charlin, L., Caccia, M., Lin, M., and Caccia, L. Online continual learning with maximal interfered retrieval. In *NeurIPS*, 2019.

Bennani, M. A., Doan, T., and Sugiyama, M. Generalisation guarantees for continual learning with orthogonal gradient descent. *Lifelong Learning Workshop at the ICML*, 2020.

Buzzega, P., Boschini, M., Porrello, A., Abati, D., and Calderara, S. Dark experience for general continual learning: a strong, simple baseline. In *NeurIPS*, 2020.

Castro, F. M., Marín-Jiménez, M. J., Guil, N., Schmid, C., and Alahari, K. End-to-end incremental learning. In *ECCV*, pp. 233–248, 2018.

Cha, H., Lee, J., and Shin, J. Co2l: Contrastive continual learning. In *ICCV*, 2021.

Chaudhry, A., Ranzato, M., Rohrbach, M., and Elhoseiny, M. Efficient lifelong learning with a-gem. In *ICLR*, 2019a.

Chaudhry, A., Rohrbach, M., Elhoseiny, M., Ajanthan, T., Dokania, P. K., Torr, P. H., and Ranzato, M. Continual learning with tiny episodic memories. 2019b.

Chaudhry, A., Khan, N., Dokania, P. K., and Torr, P. H. S. Continual learning in low-rank orthogonal subspaces, 2020.

Chaudhry, A., Gordo, A., Dokania, P., Torr, P., and Lopez-Paz, D. Using hindsight to anchor past knowledge in continual learning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(8):6993–7001, May 2021. URL https://ojs.aaai.org/index.php/AAAI/article/view/16861.

Chen, Z. and Liu, B. Lifelong machine learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 12(3):1–207, 2018.

Dhar, P., Singh, R. V., Peng, K., Wu, Z., and Chellappa, R. Learning without memorizing. In *CVPR*, 2019.

Fang, Z., Li, Y., Lu, J., Dong, J., Han, B., and Liu, F. Is out-of-distribution detection learnable? *aNeurIPS-2022*, 2022.

Guo, Y., Liu, B., and Zhao, D. Online continual learning through mutual information maximization. In *International Conference on Machine Learning*, pp. 8109–8126. PMLR, 2022.

Hayes, T. L. and Kanan, C. Lifelong machine learning with deep streaming linear discriminant analysis. In *CVPR Workshop on Continual Learning*, 2020.

Hendrycks, D. and Gimpel, K. A baseline for detecting misclassified and out-of-distribution examples in neural networks. *arXiv preprint arXiv:1610.02136*, 2016.

Henning, C., Cervera, M., D'Angelo, F., Von Oswald, J., Traber, R., Ehret, B., Kobayashi, S., Grewe, B. F., and Sacramento, J. Posterior meta-replay for continual learning. *NeurIPS*, 34, 2021.

Hou, S., Pan, X., Loy, C. C., Wang, Z., and Lin, D. Learning a unified classifier incrementally via rebalancing. In *CVPR*, pp. 831–839, 2019.

Houlsby, N., Giurgiu, A., Jastrzebski, S., Morrone, B., De Laroussilhe, Q., Gesmundo, A., Attariyan, M., and Gelly, S. Parameter-efficient transfer learning for nlp. In *International Conference on Machine Learning*, pp. 2790–2799. PMLR, 2019.

Hu, W., Lin, Z., Liu, B., Tao, C., Tao, Z., Ma, J., Zhao, D., and Yan, R. Overcoming catastrophic forgetting for continual learning via model adaptation. In *ICLR*, 2019.

Hung, C.-Y., Tu, C.-H., Wu, C.-E., Chen, C.-H., Chan, Y.-M., and Chen, C.-S. Compacting, picking and growing for unforgetting continual learning. In *NeurIPS*, volume 32, 2019.

Kamra, N., Gupta, U., and Liu, Y. Deep Generative Dual Memory Network for Continual Learning. *arXiv preprint arXiv:1710.10368*, 2017.

Ke, Z., Liu, B., and Huang, X. Continual learning of a mixed sequence of similar and dissimilar tasks. In *NeurIPS*, 2020.

Ke, Z., Liu, B., Ma, N., Xu, H., and Shu, L. Achieving forgetting prevention and knowledge transfer in continual learning. *NeurIPS*, 2021.

Kemker, R. and Kanan, C. FearNet: Brain-Inspired Model for Incremental Learning. In *ICLR*, 2018.

Kim, G. and Liu, B. Continual learning via principal components projection, 2020. URL https://openreview.net/forum?id=SkxlElBYDS.

Kim, G., Ke, Z., and Liu, B. A multi-head model for continual learning via out-of-distribution replay. In *Conference on Lifelong Learning Agents*, pp. 548–563. PMLR, 2022a.

Kim, G., Xiao, C., Konishi, T., Ke, Z., and Liu, B. A theoretical study on solving continual learning. *NeurIPS-2022*, 2022b.

Kim, G., Xiao, C., Konishi, T., Ke, Z., and Liu, B. Open-world continual learning: Unifying novelty detection and continual learning. *arXiv:2304.10038 [cs.LG]*, 2023.

Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu, A. A., Milan, K., Quan, J., Ramalho, T., Grabska-Barwinska, A., et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526, 2017.

Krizhevsky, A. and Hinton, G. Learning multiple layers of features from tiny images. *Technical Report TR-2009, University of Toronto, Toronto.*, 2009.

Le, Y. and Yang, X. Tiny imagenet visual recognition challenge, 2015.

Lee, K., Lee, K., Shin, J., and Lee, H. Overcoming catastrophic forgetting with unlabeled data in the wild. In *CVPR*, 2019.

Lee, S., Goldt, S., and Saxe, A. Continual learning in the teacher-student setup: Impact of task similarity. In *International Conference on Machine Learning*, pp. 6109–6119. PMLR, 2021.

Li, Z. and Hoiem, D. Learning Without Forgetting. In *ECCV*, pp. 614–629. Springer, 2016.

Lin, S., Yang, L., Fan, D., and Zhang, J. Beyond not-forgetting: Continual learning with backward knowledge transfer. *NeurIPS-2022*, 2022.

Liu, B., Mazumder, S., Robertson, E., and Grigsby, S. Ai autonomy: Self-initiated open-world continual learning and adaptation. *AI Magazine*, 2023.

Liu, Y., Parisot, S., Slabaugh, G., Jia, X., Leonardis, A., and Tuytelaars, T. More classifiers, less forgetting: A generic multi-classifier paradigm for incremental learning. In *ECCV*, pp. 699–716. Springer International Publishing, 2020a. doi: 10.1007/978-3-030-58574-7_42. URL https://doi.org/10.1007/978-3-030-58574-7_42.

Liu, Y., Su, Y., Liu, A.-A., Schiele, B., and Sun, Q. Mnemonics training: Multi-class incremental learning without forgetting. In *CVPR*, 2020b.

Liu, Y., Schiele, B., and Sun, Q. Adaptive aggregation networks for class-incremental learning. In *CVPR*, 2021.

Lopez-Paz, D. and Ranzato, M. Gradient Episodic Memory for Continual Learning. In *NeurIPS*, pp. 6470–6479, 2017.

Mallya, A. and Lazebnik, S. PackNet: Adding Multiple Tasks to a Single Network by Iterative Pruning. *arXiv preprint arXiv:1711.05769*, 2017.

McCloskey, M. and Cohen, N. J. Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of learning and motivation*, volume 24, pp. 109–165. Elsevier, 1989.

Ostapenko, O., Puscas, M., Klein, T., Jahnichen, P., and Nabi, M. Learning to remember: A synaptic plasticity driven framework for continual learning. In *CVPR*, pp. 11321–11329, 2019.

Ostapenko, O., Rodriguez, P., Caccia, M., and Charlin, L. Continual learning via local module composition. *Advances in Neural Information Processing Systems*, 34: 30298–30312, 2021.

Ostapenko, O., Lesort, T., Rodríguez, P., Arefin, M. R., Douillard, A., Rish, I., and Charlin, L. Continual learning with foundation models: An empirical study of latent replay. *Conference on Lifelong Learning Agents*, 2022.

Pentina, A. and Lampert, C. A pac-bayesian bound for lifelong learning. In *International Conference on Machine Learning*, pp. 991–999. PMLR, 2014.

Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., et al. Learning transferable visual models from natural language supervision. *arXiv preprint arXiv:2103.00020*, 2021.

Rajasegaran, J., Hayat, M., Khan, S., Khan, F. S., Shao, L., and Yang, M.-H. An adaptive random path selection approach for incremental learning, 2020a.

Rajasegaran, J., Khan, S., Hayat, M., Khan, F. S., and Shah, M. itaml: An incremental task-agnostic meta-learning approach. In *CVPR*, 2020b.

Rebuffi, S.-A., Kolesnikov, A., and Lampert, C. H. iCaRL: Incremental classifier and representation learning. In *CVPR*, pp. 5533–5542, 2017.

Ritter, H., Botev, A., and Barber, D. Online structured laplace approximations for overcoming catastrophic forgetting. In *NeurIPS*, 2018.

Rolnick, D., Ahuja, A., Schwarz, J., Lillicrap, T. P., and Wayne, G. Experience replay for continual learning. In *NeurIPS*, 2019.

Rostami, M., Kolouri, S., and Pilly, P. K. Complementary learning for overcoming catastrophic forgetting using experience replay. In *IJCAI*, 2019.

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

Schwarz, J., Luketina, J., Czarnecki, W. M., Grabska-Barwinska, A., Teh, Y. W., Pascanu, R., and Hadsell, R. Progress & compress: A scalable framework for continual learning. *arXiv preprint arXiv:1805.06370*, 2018.

Seff, A., Beatson, A., Suo, D., and Liu, H. Continual learning in generative adversarial nets. *arXiv preprint arXiv:1705.08395*, 2017.

Serrà, J., Surís, D., Miron, M., and Karatzoglou, A. Overcoming catastrophic forgetting with hard attention to the task. In *ICML*, 2018.

Shin, H., Lee, J. K., Kim, J., and Kim, J. Continual learning with deep generative replay. In *NIPS*, pp. 2994–3003, 2017.

Touvron, H., Cord, M., Douze, M., Massa, F., Sablayrolles, A., and Jégou, H. Training data-efficient image transformers & distillation through attention. In *International Conference on Machine Learning*, pp. 10347–10357. PMLR, 2021.

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

von Oswald, J., Henning, C., Sacramento, J., and Grewe, B. F. Continual learning with hypernetworks. *ICLR*, 2020.

Wang, H., Li, Z., Feng, L., and Zhang, W. Vim: Out-of-distribution with virtual-logit matching. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4921–4930, 2022a.

Wang, L., Zhang, X., Yang, K., Yu, L., Li, C., Hong, L., Zhang, S., Li, Z., Zhong, Y., and Zhu, J. Memory replay with data compression for continual learning. *Proceedings of International Conference on Learning Representations (ICLR)*, 2022b.

Wang, Z., Zhang, Z., Lee, C.-Y., Zhang, H., Sun, R., Ren, X., Su, G., Perot, V., Dy, J., and Pfister, T. Learning to prompt for continual learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 139–149, 2022c.

Wortsman, M., Ramanujan, V., Liu, R., Kembhavi, A., Rastegari, M., Yosinski, J., and Farhadi, A. Supermasks in superposition. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M. F., and Lin, H. (eds.), *NeurIPS*, 2020.

Wu, C., Herranz, L., Liu, X., van de Weijer, J., Raducanu, B., et al. Memory replay gans: Learning to generate new categories without forgetting. In *NeurIPS*, 2018.

Wu, T.-Y., Swaminathan, G., Li, Z., Ravichandran, A., Vasconcelos, N., Bhotika, R., and Soatto, S. Class-incremental learning with strong pre-trained models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 9601–9610, June 2022.

Wu, Y., Chen, Y., Wang, L., Ye, Y., Liu, Z., Guo, Y., and Fu, Y. Large scale incremental learning. In *CVPR*, 2019.

Xu, J. and Zhu, Z. Reinforced continual learning. In *NeurIPS*, 2018.

Yan, S., Xie, J., and He, X. Der: Dynamically expandable representation for class incremental learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 3014–3023, 2021.

Zeng, G., Chen, Y., Cui, B., and Yu, S. Continuous learning of context-dependent processing in neural networks. *Nature Machine Intelligence*, 2019.

Zenke, F., Poole, B., and Ganguli, S. Continual learning through synaptic intelligence. In *ICML*, pp. 3987–3995, 2017.

Zhu, F., Zhang, X.-Y., Wang, C., Yin, F., and Liu, C.-L. Prototype augmentation and self-supervision for incremental learning. In *CVPR*, 2021.

APPENDIX

A. Proof

*Proof of **Theorem** 3.3.* Denote the algorithm that satisfies **Definition** 3.2 as $\mathbf{A}_k^r$. Given any ϵ_n and $S \sim D_{[1:k]}$, denote $h_k = \mathbf{A}_k^r(S)$. Based on h_k, we construct a $\tilde{h}_k$ that satisfies **Definition** 3.2 but doesn't satisfy **Definition** 3.1.

We define $\mathcal{H}$ has the *capacity* to learn more than one task as follows. For any $h_0 \in \mathcal{H}$ that could only make correct predictions on a single task, but wrong predictions on all the other tasks, there exists $\delta > 0$ s.t.

$$\inf_{h\in\mathcal{H}} \mathbf{R}_{D_{[1:k]}}(h) < \mathbf{R}_{D_{[1:k]}}(h_0) - \delta.$$

Denote $h_k = \arg\max_{i,j}\{\dots, z_k^{i,j}, \dots\}$, where $z_k^{i,j}$ is the score function of the j-th class of the i-th task. Let $\sigma(z) = 1/(1+e^{-z})$ to be the sigmoid function. Define

$$\tilde{z}_k^{i,j}(x) = \sigma(z_k^{i,j}(x)) + i$$

and

$$\tilde{h}_k = \arg\max_{i,j}\{\dots, \tilde{z}_k^{i,j}, \dots\}.$$

(i) Since $0 < \sigma < 1$, we have $\tilde{z}_k^{i,j} < \tilde{z}_k^{i',j}$, $\forall\, i < i'$. Therefore,

$$\tilde{h}_k = \arg\max_{i,j}\{\dots, \tilde{z}_k^{i,j}, \dots\} = \arg\max_{i=k,j}\{\dots, \tilde{z}_k^{i,j}, \dots\}.$$

Since σ is monotonic increasing, we have

$$\begin{aligned}\mathbf{R}_{D_k}(h_k) &= \mathbb{E}_{(x,y)\sim D_k}[l(\arg\max_{i=k,j}\{\dots, z_k^{i,j}(x), \dots\}, y)] \\ &= \mathbb{E}_{(x,y)\sim D_k}[l(\arg\max_{i=k,j}\{\dots, \tilde{z}_k^{i,j}(x), \dots\}, y)] = \mathbf{R}_{D_k}(\tilde{h}_k).\end{aligned}$$

Plugging $\mathbf{R}_{D_k}(h_k) = \mathbf{R}_{D_k}(\tilde{h}_k)$ into **Definition** 3.2, we have

$$\max_{k=1,\dots,T} \mathbb{E}_{S\sim D_{[1:k]}}[\mathbf{R}_{D_k}(\tilde{h}_k) - \inf_{h\in\mathcal{H}} \mathbf{R}_{D_k}(h)] < \epsilon_n.$$

Therefore, $\tilde{h}_k$ also satisfies **Definition** 3.2.

(ii) Since $\tilde{h}_k$ always predicts the class of the k-th task, all predicted labels of samples from $D_{[1:k-1]}$ are wrong. Therefore, we have

$$\max_{k=1,\dots,T} \mathbb{E}_{S\sim D_{[1:k]}}[\mathbf{R}_{D_{[1:k]}}(\tilde{h}_k) - \inf_{h\in\mathcal{H}} \mathbf{R}_{D_{[1:k]}}(h)] > \delta > 0.$$

Because δ is a constant that is irrelevant to $S \sim D_{[1:k]}$, it cannot be reduced by increasing samples. Therefore, $\tilde{h}_k$ doesn't satisfy **Definition** 3.1. □

*Proof of **Theorem** 3.5.* Denote the algorithm that satisfies **Definition** 3.4 as $\mathbf{A}_k$. Define $h_k = \mathbf{A}_k(S)$. Let $\alpha_1 = \cdots = \alpha_k = 0$, then we have $D_{[1:k]}^{\alpha_{[1:k]}} = D_{[1:k]}$. It's obvious that h_k satisfies **Definition** 3.1 because

$$\mathbf{R}_{D_{[1:k]}}(h_k) - \inf_{h\in\mathcal{H}} \mathbf{R}_{D_{[1:k]}}(h) = \mathbf{R}_{D_{[1:k]}^{\alpha_{[1:k]}}}(h_k) - \inf_{h\in\mathcal{H}} \mathbf{R}_{D_{[1:k]}^{\alpha_{[1:k]}}}(h).$$

□

*Proof of **Theorem** 3.7.* Denote the algorithm that satisfies **Definition** 3.6 as $\mathbf{A}_k^r$. Given any ϵ_n and $S \sim D_{[1:k]}$, denote $h_k = \mathbf{A}_k^r(S)$. Based on h_k, we construct a $\tilde{h}_k$ that satisfies **Definition** 3.4.

For simplicity, we denote $\pi_{[k:k']} = \sum_{i=k}^{k'} \pi_i$.

For any $O_{(X_1,Y_1)}, \ldots, O_{(X_T,Y_T)} \in \mathcal{D}$ and any $\alpha_1, \ldots, \alpha_T \in [0,1)$, let

$$\alpha_k' = 1 - \frac{\pi_k}{\pi_{[k:T]}}(1-\alpha_k)$$

and

$$O'_{(X_k,Y_k)} = \frac{\pi_k}{\pi_{[k:T]}}\alpha_k O_{(X_k,Y_k)} + \sum_{i=k+1}^{T} \frac{\pi_i}{\pi_{[k:T]}}[(1-\alpha_i)D_i + \alpha_i O_{(X_i,Y_i)}].$$

Defining

$$D_k^{\alpha_k'} \overset{def}{=} (1-\alpha_k')D_k + \alpha_k' O'_{(X_k,Y_k)}$$

and plugging α_k' and $O'_{(X_k,Y_k)}$ into **Definition** 3.6, we have

$$\max_{k=1,\ldots,T} \mathbb{E}_{S\sim D_{[1:k]}}[\mathbf{R}_{D_k^{\alpha_k'}}(h_k) - \inf_{h\in\mathcal{H}} \mathbf{R}_{D_k^{\alpha_k'}}(h)] < \epsilon_n.$$

Denote $h_k = \arg\max\{\ldots, z_k^j, \ldots; z_k^o\}$, where z_k^j is the score function of the j-th class of the k-th task, and z_k^o is the score function of the OOD class of the k-th task. Denote the label of the j-th class of the i-th task as $y^{i,j}$. Denote the label of OOD class of the i-th task as $y^{i,o}$. We define

$$\tilde{h}_k(x) = \begin{cases} y^{i,j} \text{ if } h_{i'}(x) = y^{i',o}, \forall\, i' < i, \text{ and } h_i(x) = y^{i,j}; \\ y^{k,o} \text{ if } h_{i'}(x) = y^{i',o}, \forall\, i' \le k. \end{cases}$$

By definition of $\tilde{h}_k$, when $\tilde{h}_k$ makes a wrong prediction, there exists a $h_{i'}, i' \le k$ that makes a mistake. We assume that the loss function satisfies the following inequality

$$l(\tilde{h}_k(x), y) \le \begin{cases} \sum_{t=1}^{i-1} l(h_t(x), y^{t,o}) + l(h_i(x), y^{i,j}), \\ \qquad\qquad \text{if } h_{i'}(x) = y^{i',o}, \forall\, i' < i, \text{ and } h_i(x) = y^{i,j}; \\ \sum_{t=1}^{k} l(h_t(x), y^{t,o}), \\ \qquad\qquad \text{if } h_{i'}(x) = y^{i',o}, \forall\, i' \le k. \end{cases} \tag{8}$$

Then we can decompose the risk function of $\tilde{h}_k$,

$$\begin{aligned} \mathbf{R}_{D_{[1:k]}^{\alpha_{[1:k]}}}(\tilde{h}_k) &= \mathbb{E}_{(x,y)\sim D_{[1:k]}^{\alpha_{[1:k]}}} l(\tilde{h}_k(x), y) \\ &= \sum_{i=1}^{k} \mathbb{E}_{(x,y)\sim D_{[1:k]}^{\alpha_{[1:k]}}} l(\tilde{h}_k(x), y) 1_{(x,y)\sim D_i} \\ &\quad + \sum_{i=1}^{k} \mathbb{E}_{(x,y)\sim D_{[1:k]}^{\alpha_{[1:k]}}} l(\tilde{h}_k(x), y) 1_{(x,y)\sim O_i} \\ &\le \sum_{i=1}^{k} \mathbb{E}_{(x,y)\sim D_{[1:k]}^{\alpha_{[1:k]}}} [\sum_{t=1}^{i-1} l(h_t(x), y^{t,o}) + l(h_i(x), y^{i,j})] 1_{(x,y)\sim D_i} \\ &\quad + \sum_{i=1}^{k} \mathbb{E}_{(x,y)\sim D_{[1:k]}^{\alpha_{[1:k]}}} [\sum_{t=1}^{i} l(h_t(x), y^{t,o})] 1_{(x,y)\sim O_i}. \end{aligned} \tag{9}$$

With the fact that for any density function $p(x)$ defined on A and any $B \subset A$,

$$\int_A \frac{p(x)}{\int_A p(x)dx} f(x) 1_{x \in B} dx = \frac{\int_B p(x)dx}{\int_A p(x)dx} \int_B \frac{p(x)}{\int_B p(x)dx} f(x)dx,$$

by definition of α'_k and $O'_{(X_k, Y_k)}$, we have that for $t < i \leq k$,

$$\begin{aligned}
&\mathbb{E}_{(x,y)\sim D_{[1:k]}^{\alpha_{[1:k]}}} l(h_t(x), y^{t,o}) 1_{(x,y)\sim D_i} \\
&= \frac{(1-\alpha_i)\pi_i}{\pi_{[1:k]}} \mathbb{E}_{(x,y)\sim D_i} l(h_t(x), y^{t,o}) \\
&= \frac{(1-\alpha_i)\pi_i}{\pi_{[1:k]}} \frac{\pi_{[t:T]}}{(1-\alpha_i)\pi_i} \mathbb{E}_{(x,y)\sim D_t^{\alpha'_t}} l(h_t(x), y^{t,o}) 1_{(x,y)\sim D_i} \\
&= \frac{\pi_{[t:T]}}{\pi_{[1:k]}} \mathbb{E}_{(x,y)\sim D_t^{\alpha'_t}} l(h_t(x), y^{t,o}) 1_{(x,y)\sim D_i},
\end{aligned}$$

$$\begin{aligned}
&\mathbb{E}_{(x,y)\sim D_{[1:k]}^{\alpha_{[1:k]}}} l(h_i(x), y^{i,j}) 1_{(x,y)\sim D_i} \\
&= \frac{(1-\alpha_i)\pi_i}{\pi_{[1:k]}} \mathbb{E}_{(x,y)\sim D_i} l(h_i(x), y^{i,j}) \\
&= \frac{(1-\alpha_i)\pi_i}{\pi_{[1:k]}} \frac{\pi_{[i:T]}}{(1-\alpha_i)\pi_i} \mathbb{E}_{(x,y)\sim D_i^{\alpha'_i}} l(h_i(x), y^{i,j}) 1_{(x,y)\sim D_i} \\
&= \frac{\pi_{[i:T]}}{\pi_{[1:k]}} \mathbb{E}_{(x,y)\sim D_i^{\alpha'_i}} l(h_i(x), y^{i,j}) 1_{(x,y)\sim D_i},
\end{aligned}$$

and for $t \leq i \leq k$,

$$\begin{aligned}
&\mathbb{E}_{(x,y)\sim D_{[1:k]}^{\alpha_{[1:k]}}} l(h_t(x), y^{t,o}) 1_{(x,y)\sim O_i} \\
&= \frac{\alpha_i \pi_i}{\pi_{[1:k]}} \mathbb{E}_{(x,y)\sim O_i} l(h_t(x), y^{t,o}) \\
&= \frac{\alpha_i \pi_i}{\pi_{[1:k]}} \frac{\pi_{[t:T]}}{\alpha_i \pi_i} \mathbb{E}_{(x,y)\sim D_t^{\alpha'_t}} l(h_t(x), y^{t,o}) 1_{(x,y)\sim O_i} \\
&= \frac{\pi_{[t:T]}}{\pi_{[1:k]}} \mathbb{E}_{(x,y)\sim D_t^{\alpha'_t}} l(h_t(x), y^{t,o}) 1_{(x,y)\sim O_i}.
\end{aligned}$$

Plugging the above three equations into Eq. 9, we have

$$\begin{aligned}
\mathbf{R}_{D_{[1:k]}^{\alpha_k}}(\tilde{h}_k) \leq & \sum_{i=1}^{k}\sum_{t=1}^{i-1}\frac{\pi_{[t:T]}}{\pi_{[1:k]}}\mathbb{E}_{(x,y)\sim D_t^{\alpha_t'}} l(h_t(x), y^{t,o})1_{(x,y)\sim D_i} \\
& + \sum_{i=1}^{k}\frac{\pi_{[i:T]}}{\pi_{[1:k]}}\mathbb{E}_{(x,y)\sim D_i^{\alpha_i'}} l(h_i(x), y^{i,j})1_{(x,y)\sim D_i} \\
& + \sum_{i=1}^{k}\sum_{t=1}^{i}\frac{\pi_{[t:T]}}{\pi_{[1:k]}}\mathbb{E}_{(x,y)\sim D_t^{\alpha_t'}} l(h_t(x), y^{t,o})1_{(x,y)\sim O_i} \\
= & \sum_{t=1}^{k}\sum_{i=t+1}^{k}\frac{\pi_{[t:T]}}{\pi_{[1:k]}}\mathbb{E}_{(x,y)\sim D_t^{\alpha_t'}} l(h_t(x), y^{t,o})1_{(x,y)\sim D_i} \\
& + \sum_{t=1}^{k}\frac{\pi_{[t:T]}}{\pi_{[1:k]}}\mathbb{E}_{(x,y)\sim D_t^{\alpha_t'}} l(h_t(x), y^{t,j})1_{(x,y)\sim D_t} \\
& + \sum_{t=1}^{k}\sum_{i=t}^{k}\frac{\pi_{[t:T]}}{\pi_{[1:k]}}\mathbb{E}_{(x,y)\sim D_t^{\alpha_t'}} l(h_t(x), y^{t,o})1_{(x,y)\sim O_i} \\
= & \sum_{t=1}^{k}\frac{\pi_{[t:T]}}{\pi_{[1:k]}}\mathbb{E}_{(x,y)\sim D_t^{\alpha_t'}} l(h_t(x), y^{t,*})1_{(x,y)\sim \cup_{i=t}^{k}(D_i\cup O_i)} \\
= & \sum_{t=1}^{k}\frac{\pi_{[t:T]}}{\pi_{[1:k]}}\mathbf{R}_{D_t^{\alpha_t'}}(h_t).
\end{aligned}$$

By assumption that $\inf_{h\in\mathcal{H}} \mathbf{R}_{D_{[1:k]}^{\alpha_{[1:k]}}}(h) = 0$, it's obvious that $\inf_{h\in\mathcal{H}} \mathbf{R}_{D_k^{\alpha_k'}}(h) = 0$, which means that

$$\max_{k=1,\ldots,T} \mathbb{E}_{S\sim D_{[1:k]}}[\mathbf{R}_{D_k^{\alpha_k'}}(h_k)] < \epsilon_n.$$

Therefore, we have

$$\begin{aligned}
& \max_{k=1,\ldots,T} \mathbb{E}_{S\sim D_{[1:k]}}[\mathbf{R}_{D_{[1:k]}^{\alpha_{[1:k]}}}(\tilde{h}_k) - \inf_{h\in\mathcal{H}} \mathbf{R}_{D_{[1:k]}^{\alpha_{[1:k]}}}(h)] \\
\leq & \max_{k=1,\ldots,T} \mathbb{E}_{S\sim D_{[1:k]}}[\sum_{t=1}^{k}\frac{\pi_{[t:T]}}{\pi_{[1:k]}}\mathbf{R}_{D_t^{\alpha_t'}}(h_t)] \\
< & \epsilon_n \cdot \max_{k=1,\ldots,T}\sum_{t=1}^{k}\frac{\pi_{[t:T]}}{\pi_{[1:k]}}.
\end{aligned}$$

□

*Proof of **Corollary** 3.8.* Denote the algorithm that satisfies **Definition** 3.6 as $\mathbf{A}_k^r$. Given any ϵ_n and $S \sim D_{[1:k]}$, denote $h_k = \mathbf{A}_k^r(S)$. Based on h_k, we construct a $\tilde{h}_k$ that satisfies **Definition** 3.4.

For simplicity, we denote $\pi_{[k:k']} = \sum_{i=k}^{k'} \pi_i$.

Different from proof of **Theorem** 3.7, when we can acquire replay data and therefore treat them as OOD data, we let

$$\alpha_k' = 1 - \frac{\pi_k}{\pi_{[1:T]}}(1-\alpha_k)$$

and

$$O'_{(X_k,Y_k)} = \frac{\pi_k}{\pi_{[1:T]}}\alpha_k O_{(X_k,Y_k)} + \sum_{i\neq k}\frac{\pi_i}{\pi_{[1:T]}}[(1-\alpha_i)D_i + \alpha_i O_{(X_i,Y_i)}].$$

Defining

$$D_k^{\alpha_k'} \overset{def}{=} (1-\alpha_k')D_k + \alpha_k' O'_{(X_k,Y_k)}$$

and plugging α_k' and $O'_{(X_k,Y_k)}$ into **Definition** 3.6, we have

$$\max_{k=1,\dots,T} \mathbb{E}_{S\sim D_{[1:k]}}[\mathbf{R}_{D_k^{\alpha_k'}}(h_k) - \inf_{h\in\mathcal{H}} \mathbf{R}_{D_k^{\alpha_k'}}(h)] < \epsilon_n.$$

Denote $h_k = \arg\max\{\dots, z_k^j, \dots; z_k^o\}$, where z_k^j is the score function of the j-th class of the k-th task, and z_k^o is the score function of the OOD class of the k-th task. Denote the label of the j-th class of the i-th task as $y^{i,j}$. Denote the label of OOD class the the i-th task as $y^{i,o}$. We define

$$\tilde{h}_k(x) = \begin{cases} y^{i,j} \text{ if } h_i(x) = y^{i,j}, \exists\, i \le k; \\ y^{k,o} \text{ if } h_{i'}(x) = y^{i',o}, \forall\, i' \le k. \end{cases}$$

It's ideal that $h_{i'}(x) = y^{i',o}, \forall\, i' \neq i$ when $h_i(x) = y^{i,j}, \exists\, i \le k$. But when $\tilde{h}_k$ makes a wrong prediction, there exists a $h_{i'}, i' \le i$ that makes a mistake. We assume that the loss function satisfies the following inequality

$$l(\tilde{h}_k(x), y) \le \begin{cases} \sum_{t\neq i} l(h_t(x), y^{t,o}) + l(h_i(x), y^{i,j}), \\ \qquad\qquad \text{if } h_i(x) = y^{i,j}, \exists\, i \le k; \\ \sum_{t=1}^{k} l(h_t(x), y^{t,o}), \\ \qquad\qquad \text{if } h_{i'}(x) = y^{i',o}, \forall\, i' \le k. \end{cases} \tag{10}$$

All the same as the proof of **Theorem** 3.7, we have

$$\begin{aligned} \mathbf{R}_{D_{[1:k]}^{\alpha_{[1:k]}}}(\tilde{h}_k) &= \mathbb{E}_{(x,y)\sim D_{[1:k]}^{\alpha_{[1:k]}}} l(\tilde{h}_k(x), y) \\ &\le \sum_{i=1}^{k} \mathbb{E}_{(x,y)\sim D_{[1:k]}^{\alpha_{[1:k]}}}[\sum_{t\neq i} l(h_t(x), y^{t,o}) + l(h_i(x), y^{i,j})]1_{(x,y)\sim D_i} \\ &\quad + \sum_{i=1}^{k} \mathbb{E}_{(x,y)\sim D_{[1:k]}^{\alpha_{[1:k]}}}[\sum_{t=1}^{i} l(h_t(x), y^{t,o})]1_{(x,y)\sim O_i} \\ &\le \epsilon_n \cdot \max_{k=1,\dots,T} \sum_{t=1}^{k} \frac{\pi_{[1:T]}}{\pi_{[1:k]}} = \epsilon_n \cdot \max_{k=1,\dots,T} \frac{k\pi_{[1:T]}}{\pi_{[1:k]}}. \end{aligned}$$

□

B. Hard Attention to the Task (HAT)

In training the network $h_k \circ f_k$ using the data of task k and the generated pseudo feature vectors, we employ the hard attention mask (Serrà et al., 2018) to prevent forgetting in the feature extractor.

The hard attention mask a_l^k is a trainable pseudo binary 0-1 vector at each layer l of task k. It is element-wise multiplied to the output of the layer as $a_l^k \otimes h_l$ and blocks (for value of 0) or unblocks (for value of 1) the information flow from neurons of adjacent layers. Neurons with value 1 are important for the task and thus need to be protected while neurons with value 0 are not necessary for the task and can be freely modifed without affecting other tasks.

More specifically, we modify the gradients of parameters that are important in performing the previous tasks $(1, \cdots, k-1)$ during training task k so the important parameters for previous tasks are unaffected. The gradient of parameter $w_{ij,l}$ at ith row and jth column of layer l is modified as

$$\nabla w'_{ij,l} = \left(1 - \min\left(a_{i,l}^{<k}, a_{j,l-1}^{<k}\right)\right) \nabla w_{ij,l}, \tag{11}$$

where $a_{i,l}^{<k}$ is an accumulated attentions over previous tasks and is 1 if the hard attention of neuron i at layer l is ever used by any previous task $< k$ (see (Serrà et al., 2018) for details).

To encourage parameter sharing and sparsity in the number of activated masks, a regularization is introduced as $\mathcal{L}_r = \sum_{l,i} a_{i,l}^k (1 - a_{i,l}^{<k}) / \sum_{l,i} (1 - a_{i,l}^{<k})$. The final objective to train a comprehensive task network without forgetting is

$$\mathcal{L} = \mathcal{L}_{ood} + \mathcal{L}_r, \tag{12}$$

where $\mathcal{L}_{ood}$ is the cross-entropy loss in Eq. 3.

C. Hyper-Parameters

For all the experiments, we use SGD with momentum value of 0.9 with batch size of 64. For C10-5T, we use learning rate 0.005 and train for 20 epochs. For C100-10T and 20T, we train for 40 epochs with learning rate 0.001 and 0.005 for 10T and 20T, respectively. For T-5T and 10T, we use the same learning rate 0.005, but train for 15 and 10 epochs for 5T and 10T, respectively. For fine-tuning WP and OOD head, we use batch size of 32 and use the same learning rate used for training the feature extractor. For fine-tuning WP and TP, we train for 5 epochs and 10 epochs, respectively.

D. Required Memory

Table 5. The size of the model (in entries) required for each method without the memory buffer.

Method	C10-5T	C100-10T	C100-20T	T-5T	T-10T
HAT	23.0M	24.7M	25.4M	24.6M	25.1M
OWM	26.6M	28.1M	28.1M	28.2M	28.2M
SLDA	21.6M	21.6M	21.6M	21.7M	21.7M
PASS	22.9M	24.2M	24.2M	24.3M	24.4M
L2P	21.7M	21.7M	21.7M	21.8M	21.8M
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
DER++	22.9M	24.1M	24.1M	24.1M	24.1M
HAL	22.9M	24.1M	24.1M	24.1M	24.1M
MORE	23.7M	25.9M	27.7M	25.1M	25.9M
ROW	23.7M	26.0M	27.8M	25.2M	26.0M

We report the network sizes of the systems after learning the last task. We use an 'entry' to denote a parameter or a value required to learn and to do inference for a task.

All the systems except SLDA and L2P use the feature extractor DeiT-S/16 (Touvron et al., 2021) and adapter modules. The transformer consumes 21.6 millions (M) entries and the adapters take 1.2M and 2.4M entries for CIFAR10 and the other

datasets. SLDA fine-tunes only the classifier on top of the fixed pre-trained feature extractor as it does not have a protection mechanism. L2P uses a prompt pool with 23k entries. Since each method requires method-specific elements (e.g., task embedding for HAT), the number of entries required for each method is different. The number of entries for each model is reported in Tab. 5.

Our system saves the covariance matrices for computing the distance-based coefficient in Sec. 4.1. The covariances are saved for each task. Since each covariance is in size 384x384, the total entries for this step are 737.3k, 1.5M, 2.9M, 737.3k, and 1.5M for C10-5T, C100-10T, C100-20T, T-5T, and T-10T, respectively. The numbers are relatively small considering that some of the replay-based methods (e.g., iCaRL, HAL) require a teacher model the same size as the training model for knowledge distillation. More importantly, replay buffer requires the largest memory (e.g., for Tiny-ImageNet, saving 2,000 images of size 64x64x3 requires 24.6M entries). It is highly important that the system is robust to replay buffer size. ROW is shown to remain strong with small replay buffer sizes (see Tab. 3).

Our system ROW saves an additional classifier. WP head is of the same shape as the classifier of the standard baselines (e.g., iCaRL or DER++). OOD (or TP) head requires the same memory as WP with additional parameters for OOD class per task. The required memory is small. For instance, for C10-5T, using OOD head only introduces 5,775 additional entries.

E. Societal Impact and Limitation

We do not see any negative societal impact as we use public domain datasets for the experiments and our algorithm is just like any normal supervised learning in nature. In practical use, if the training data of the application is biased, it could affect the model just like in any other supervised learning. We believe that this can be alleviated by checking any potential bias in the training data.

The current theoretical study is only applicable to offline CIL. In future work, we will extend our study to online CIL, where the task boundary may be blurry.

F. Effect on Underrepresented Minorities

The existence of underrepresented samples in the current task does not affect our theory, but it will affect an actual implementation and give weaker results. The OOD detection method in our paper simply trains the system by considering the samples of the current task as in-distribution and the samples in the replay buffer as OOD of the current task. In case there is a set of underrepresented classes in the current task's dataset, one can use existing techniques proposed for the sample imbalance problem to alleviate the issue. However, this problem is not just relevant to our proposed method, but relevant to all existing OOD and CL methods, or even supervised learning methods.