A Theoretical Study on Solving Continual Learning

Gyuhak Kim[*1], **Changnan Xiao**[*2], **Tatsuya Konishi**[† 3], **Zixuan Ke**[1], **Bing Liu**[‡ 1]
[1] University of Illinois at Chicago
[2] ByteDance
[3] KDDI Research

Abstract

Continual learning (CL) learns a sequence of tasks incrementally. There are two popular CL settings, *class incremental learning* (CIL) and *task incremental learning* (TIL). A major challenge of CL is *catastrophic forgetting* (CF). While several techniques are available to effectively overcome CF for TIL, CIL remains to be challenging due to the additional difficulty of *inter-task class separation*. So far little theoretical work has been done to provide a *principled guidance* and *necessary and sufficient* conditions for solving the CIL problem. This paper performs such a study. It first probabilistically decomposes the CIL problem into two sub-problems: *within-task prediction* (WP) and *task-id prediction* (TP). It further proves that TP is correlated with *out-of-distribution* (OOD) detection. The key *result* is that regardless of whether WP and TP or OOD detection are defined explicitly or implicitly by a CIL algorithm, good WP and good TP or OOD detection are *necessary* and *sufficient* for good CIL performances. Additionally, TIL is simply WP. Based on the theoretical result, new CIL methods are also designed, which outperform strong baselines in both CIL and TIL settings by a large margin.[4]

1 Introduction

Continual learning aims to incrementally learn a sequence of tasks [1]. Each task consists of a set of classes to be learned together. A major challenge of CL is *catastrophic forgetting* (CF). Although a large number of CL techniques have been proposed, they are mainly empirical. Limited theoretical research has done on how to solve CL. This paper performs such a theoretical study about the necessary and sufficient conditions for effective CL. There are two main CL settings that have been extensively studied: *class incremental learning* (CIL) and *task incremental learning* (TIL) [2]. 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, $1, 2, ..., T$. Each task k 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.

In the TIL setup, each task is a separate 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

[*]Equal contribution
[†]The work was done when this author was visiting Bing Liu's group at University of Illinois at Chicago
[‡]Correspondance author. Bing Liu <liub@uic.edu>
[4]The code is available at https://github.com/k-gyuhak/WPTP

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

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, $1, 2, ..., T$. Each task k 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).

Several techniques are available to effectively overcome CF for TIL (with almost no CF) [3, 4]. However, CIL remains to be highly challenging due to the additional problem of *Inter-task Class Separation* (ICS) (establishing class boundaries between the classes of the new task and the previous tasks) as the previous task data is not accessible. Before discussing the proposed work, let us recall the *closed-world* assumption made by traditional machine learning, which means that the classes seen in testing must have been seen in training [1, 5]. However, in many applications, there are unknowns in the testing or applications, which is called the *open world* setting [1, 5]. In open world learning, the training (or known) classes are called *in-distribution* (IND) classes. A classifier built for the open world can (1) classify test instances of training/IND classes to their respective classes, which is called *IND prediction*, and (2) detect test instances that do not belong to any of the IND/known classes but some unknown or *out-of-distribution* (OOD) classes, which is called *OOD detection*. Many OOD detection algorithms can perform both IND prediction and OOD detection [6, 7, 8, 9]. The commonality of OOD detection and CL is that they both need to consider future unknowns.

This paper conducts a theoretical study of CIL, which is applicable to any CIL classification model. Instead of focusing on the traditional PAC generalization bound [10, 11], 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 done 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 in testing. This paper then proves based on the popular cross-entropy loss that (1) the CIL performance is bounded by WP and TP performances, and (2) TP and task OOD detection performance bound each other (which connects CL and OOD detection). The key result is that regardless of whether WP and TP or OOD detection are defined explicitly or implicitly by a CIL algorithm, good WP and good TP or OOD detection are *necessary* and *sufficient* conditions for good CIL performances. This result is applicable to both batch/offline and online CIL and to CIL problems with blurry task boundaries. The intuition is also quite simple because if a CIL model is perfect at detecting OOD samples for each task (which solves the ICS problem), then CIL is reduced to WP.

This theoretical result provides a principled guidance for solving the CIL problem, i.e., to help design better CIL algorithms that can achieve strong WP and TP performances. Since WP is basically IND prediction for each task and most OOD techniques perform both IND prediction and OOD detection, to achieve good CIL accuracy, a strong OOD performance for each task is necessary.

Based on the theoretical guidance, several new CIL methods are designed, including techniques based on the integration of a TIL method and an OOD detection method for CIL, which outperform strong baselines in both the CIL and TIL settings by a large margin. This combination is particularly attractive because TIL has achieved no forgetting, and we only need a strong OOD technique that can perform both IND prediction and OOD detection to learn each task to achieve strong CIL results.

2 Related Work

Although numerous CL techniques have been proposed, little study has been done to provide a theoretical guidance on how to solve the problem. Existing approaches mainly belong to several categories. Using regularization [12, 13] to minimize changes to model parameters learned from previous tasks is a popular approach [14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26]. Memorizing some old examples and using them to jointly train the new task is another popular approach (called *replay*) [27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40]. Some systems learn to generate pseudo training data of old tasks and use them to jointly train the new task, called *pseudo-replay* [41, 42, 43, 44, 45, 46, 21, 47, 48]. Orthogonal projection learns each task in an orthogonal space to other tasks [49, 50, 51]. Our theoretical study is applicable to any continually trained classification models.

Parameter isolation is yet another popular approach, which makes different subsets (which may overlap) of the network parameters dedicated to different tasks using masks [3, 52, 53, 4, 54]. This approach is particularly suited for TIL. Several methods have almost completely overcome forgetting. HAT [3] and CAT [52] protect previous tasks by masking the important parameters to those tasks. PackNet [53], CPG [54] and SupSup [4] find an isolated sub-network for each task. HyperNet [55] initializes task-specific parameters conditioned on task-id. ADP [56] decomposes parameters into shared and adaptive parts to construct an order robust TIL system. CCLL [57] uses task-adaptive calibration in convolution layers. Our methods designed based on the proposed theory make use of two parameter isolation-based TIL methods and two OOD detection methods. A strong OOD detection method CSI in [6] helps produce very strong CIL results. CSI is based on data augmentation [58] and contrastive learning [59]. Excellent surveys of OOD detection include [60, 61].

Some methods have used a TIL method for CIL with an additional task-id prediction technique. iTAML [62] requires each test batch to be from a single task. This is not practical as test samples usually come one by one. CCG [63] builds a separate network to predict the task-id. Expert Gate [64] constructs a separate autoencoder for each task. HyperNet [55] and PR-Ent [65] use entropy to predict the task id. Since none of these papers is a theoretical study, they did not know that strong OOD detection is the key. Our methods based on OOD detection perform dramatically better.

Several theoretical studies have been made on lifelong/continual learning. However, they focus on traditional generalization bound. [10] proposes a PAC-Bayesian framework to provide a learning bound on expected error in future tasks by the average loss on the observed tasks. The work in [66] studies the generalization error by task similarity and [11] studies the dependence of generalization error on sample size or number of tasks including forward and backward transfer. [67] shows that orthogonal gradient descent gives a tighter generalization bound than SGD. Our work is very different as we focus on how to solve the CIL problem, which is orthogonal to the existing theoretical analysis.

3 CIL by Within-Task Prediction and Task-ID Prediction

This section presents our theory. 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 CL and OOD detection. Finally, we study the necessary conditions for a good CIL model, which includes a good WP, and a good TP (or OOD detection).

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. A CL system learns a sequence of tasks $\{(\mathbf{X}_k, \mathbf{Y}_k)\}_{k=1,\ldots,T}$, 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 jth class in task k. Let $\mathbf{X}_{k,j}$ to be the domain of jth 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) (Sec. 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 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 the training process, but we will propose some concrete learning algorithms based on the theoretical result in the experiment section.

Remark 3. Note that the CIL definition and the subsequent analysis are applicable 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 [68]) 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. Furthermore, 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 CL 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

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

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 ϵ and δ 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 $U(\mathbf{X})$ represents 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 Sec. 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 kth 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

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

In CIL, the 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 is a sigmoid of maximum logit value or a 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 $H_{TP}(x) \leq \delta$, let $\mathbf{P}'_k(x \in \mathbf{X}_k|D) = \mathbf{P}(x \in \mathbf{X}_k|D)$, then $H_{OOD,k}(x) \leq \delta, \forall\, k = 1, \ldots, T$. ii) If $H_{OOD,k}(x) \leq \delta_k, k = 1, \ldots, T$, let $\mathbf{P}(x \in \mathbf{X}_k|D) = \frac{\mathbf{P}'_k(x \in \mathbf{X}_k|D)}{\sum_k \mathbf{P}'_k(x \in \mathbf{X}_k|D)}$, then $H_{TP}(x) \leq (\sum_k \mathbf{1}_{x \in \mathbf{X}_k} e^{\delta_k})(\sum_k 1 - e^{-\delta_k})$, 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 δ). Similarly, the second statement (ii) says that the TP performance improves if the OOD detection performance on each task improves (i.e., lower δ_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 δ_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.

In Theorem 1, we studied how CIL is related to WP and TP. In Theorem 2, we showed that TP and OOD 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.4.

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

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

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 are sufficient to guarantee a good performance of CIL. For completeness, we study the necessary conditions of a well-performed CIL in this sub-section.

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

The detailed proof is given in Appendix A.5. 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, $H_{OOD,k}(x) > \eta$, $k = 1, \ldots, T$, 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 a good CIL performance.

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 4.3 and 4.4).

Recall in Section 2, we reviewed prior works that have tried to use a TIL method for CIL with a task-id prediction method [55, 64, 62, 63, 65]. However, since they did not know that the key to the success of this approach is a strong OOD detection algorithm, they are quite weak (see Section 4).

4 New CIL Techniques and Experiments

Based on Theorem 3, we have designed several new CIL methods, each of which integrates an existing CL algorithm and an OOD detection algorithm. The OOD detection algorithm that we use can perform both within-task IND prediction (WP) and OOD detection. Our experiments have two goals: (1) to show that a good OOD detection method can help improve the accuracy of an existing CIL algorithm, and (2) to fully compare two of these methods (see some others in Sec. 4.5) with strong baselines to show that they outperform the existing strong baselines considerably.

4.1 Datasets, CL Baselines and OOD Detection Methods

Datasets and CIL Tasks. Four popular benchmark image classification datasets are used, from which six CIL problems are created following recent papers [25, 34, 26]. **(1)** ***MNIST*** consists of handwritten images of 10 digits with 60,000/10,000 training/testing samples. We create a CIL problem (**M-5T**) of 5 tasks with 2 consecutive classes/digits as a task. **(2)** ***CIFAR-10*** consists of 32x32 color images of 10 classes with 50,000/10,000 training/testing samples. We create a CIL problem (**C10-5T**) of 5 tasks with 2 consecutive classes as a task. **(3)** ***CIFAR-100*** consists of 60,000 32x32 color images with 50,000/10,000 training/testing samples. We create two CIL problems by splitting 100 classes into 10 tasks (**C100-10T**) and 20 tasks (**C100-20T**), where each task has 10 and 5 classes, respectively. **(4)** ***Tiny-ImageNet*** has 120,000 64x64 color images of 200 classes with 500/50 images per class for training/testing. We create two CIL problems by splitting 200 classes into 5 tasks (**T-5T**) and 10 tasks (**T-10T**), where each task has 40 and 20 classes, respectively.

Baseline CL Methods. We include different families of CL methods: *regularization*, *replay*, *orthogonal projection*, and *parameter isolation*. MUC [25] and PASS [26] are regularization-based methods. For replay methods, we use LwF [13], iCaRL [29], Mnemonics [69], BiC [32], DER++ [34], and Co^{2}L [37]. For orthogonal projection, we use OWM [49]. Finally, for parameter isolation, we use CCG [63], HyperNet [55], HAT [3], SupSup [4] (Sup), and PR [65].[5] We use the official codes for the baselines except for Co^{2}L, CCG, and PR. For these three systems, we copy the results from their papers as the code for CCG is not released and we are unable to run Co^{2}L and PR on our machines.

OOD Detection Methods. Two OOD detection methods are used. We combine them with the above existing CL algorithms. Both these methods can also perform *within-task IND prediction* (WP).

(1). ODIN: Researchers have proposed several methods to improve the OOD detection performance of a trained network by post-processing [7, 70, 71]. ODIN [7] is a representative method. It adds perturbation to input and applies a temperature scaling to the softmax output of a trained network.

(2). CSI: It is a recently proposed OOD detection technique [6] that is highly effective. It is based on data and class label augmentation and supervised contrastive learning [72]. Its rotation data

[5] iTAML [62] is not included as it requires a batch of test data from the same task to predict the task-id. When each batch has only one test sample, which is our setting, it is very weak. For example, its CIL accuracy is only 33.5% on C100-10T. Expert Gate (EG) [64] is also very weak. Its CIL accuracy is only 43.2 on M-5T. They are much weaker than many baselines. DER [38] is not included as it expands the network after each task, which is somewhat unfair to other systems as all others do not expand the network. DER can generate a large number of parameters after the last task, e.g., 117.6 millions (M) for C100-20T while our proposed methods require 44.6M (HAT+CSI) and 11.6M (Sup+CSI) (refer to Appendix H). The average accuracy of DER over the 6 CL experiments is 61.4 while our methods achieve 67.9 (HAT+CSI+c) and 64.9 (Sup+CSI+c) (refer to Tab. 3).

augmentations create distributional shifted samples to act as negative data for the original samples for contrastive learning. The details of CSI is given in Appendix D.

4.2 Training Details and Evaluation Metrics

Training Details. For the backbone structure, we follow [4, 26, 34]. AlexNet-like architecture [73] is used for MNIST and ResNet-18 [74] is used for CIFAR-10. For CIFAR-100 and Tiny-ImageNet, ResNet-18 is also used as CIFAR-10, but the number of channels are doubled to fit more classes. All the methods use the same backbone architecture except for OWM and HyperNet, for which we use their original architectures. OWM uses AlexNet. It is not obvious how to apply the technique to the ResNet structure. HyperNet uses a fully-connected network and ResNet-32 for MNIST and other datasets, respectively. We are unable to change the structure due to model initialization arguments unexplained in the original paper. For the replay methods, we use memory buffer 200 for MNIST and CIFAR-10 and 2000 for CIFAR-100 and Tiny-ImageNet as in [29, 34]. We use the hyper-parameters suggested by the authors. If we could not 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 G. 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 *prediction methods* (see below). We also report *forgetting rate* in Appendix J.

(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, e.g., HAT and Sup. For HAT and Sup, we can simply switch their methods for learning each task with CSI (see Sec.4.4).

Prediction Methods. The theoretical result in Sec. 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.)

$$\hat{y} = \arg\max f(x) \tag{8}$$

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

$$\hat{y} = \arg\max \bigoplus_k f(x)_k \tag{9}$$

where $\bigoplus$ indicate concatenation and $f(x)_k$ is the output of task k.[6]

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

[6]The Sup paper proposed an 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 accuracy of 50.2 on C10-5T, which is much lower than 62.6 by using Eq. 9. The task-id prediction of HyperNet also works poorly. The accuracy by its id prediction is 49.34 on C10-5T while it is 53.4 using Eq. 9. 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).

4.3 Better OOD Detection Produces Better CIL Performance

The key theoretical result in Sec. 3 is that better OOD detection will produce better CIL performance. Recall our considered methods ODIN and CSI can perform both WP and OOD detection.

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 τ_k of task k as

$$s(x;\tau_k)_j = e^{f(x)_{kj}/\tau_k} / \sum_j e^{f(x)_{kj}/\tau_k} \tag{10}$$

and the input changes by the noise factor ϵ_k as

$$\tilde{x} = x - \epsilon_k \text{sign}(-\nabla_x \log s(x;\tau_k)_{\hat{y}}) \tag{11}$$

where $\hat{y}$ is the class with the maximum output value in task k. This is a positive adversarial example inspired by [75]. The values τ_k and ϵ_k are hyper-parameters and we use the same values for all tasks except for PASS, for which we had to use a validation set to tune τ_k (see Appendix B).

Tab. 1 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 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 experienced 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 [32]. 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 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.

Table 1: Performance comparison based on C100-10T between the original output and the output post-processed with OOD detection technique ODIN. Note that ODIN is not applicable to iCaRL and Mnemonics as they are not based on softmax but some distance functions. The results for other datasets are reported in Appendix B.

Method	OOD	AUC	CIL
OWM	Original	71.31	28.91
	ODIN	70.06	28.88
MUC	Original	72.69	30.42
	ODIN	72.53	29.79
PASS	Original	69.89	33.00
	ODIN	69.60	31.00
LwF	Original	88.30	45.26
	ODIN	87.11	51.82
BiC	Original	87.89	52.92
	ODIN	86.73	48.65
DER++	Original	85.99	53.71
	ODIN	88.21	55.29
HAT	Original	77.72	41.06
	ODIN	77.80	41.21
HyperNet	Original	71.82	30.23
	ODIN	72.32	30.83
Sup	Original	79.16	44.58
	ODIN	80.58	46.74

Applying CSI. We now apply the OOD detection method CSI. Due to its sophisticated data augmentation, supervised constrative learning and results ensemble, it is hard to apply CSI to other baselines without fundamentally change 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 (the full detail is given in Appendix D). As mentioned earlier, both HAT and SupSup as TIL methods have almost no forgetting.

Tab. 2 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.

4.4 Full Comparison of HAT+CSI and Sup+CSI with Baselines

We now make a full comparison of the two strong systems (HAT+CSI and Sup+CSI) designed based on the theoretical results. These combinations are particularly attractive because both HAT and Sup are TIL systems and have little or no CF. Then a strong OOD method (that can also perform WP (within-task/IND prediction) will result in a strong CIL method. Since HAT and Sup are exemplar-free CL methods, HAT+CSI and Sup+CSI also do not need to save any previous task data. Tab. 3

Table 2: Average CIL and AUC of HAT and Sup with 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

Table 3: Average accuracy after all tasks are learned. 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.

Method	M-5T	C10-5T	C100-10T	C100-20T	T-5T	T-10T
OWM	95.8±0.13	51.8±0.05	28.9±0.60	24.1±0.26	10.0±0.55	8.6±0.42
MUC	74.9±0.46	52.9±1.03	30.4±1.18	14.2±0.30	33.6±0.19	17.4±0.17
PASS†	76.6±1.67	47.3±0.98	33.0±0.58	25.0±0.69	28.4±0.51	19.1±0.46
LwF	85.5±3.11	54.7±1.18	45.3±0.75	44.3±0.46	32.2±0.50	24.3±0.26
iCaRL*	96.0±0.43	63.4±1.11	51.4±0.99	47.8±0.48	37.0±0.41	28.3±0.18
Mnemonics†*	96.3±0.36	64.1±1.47	51.0±0.34	47.6±0.74	37.1±0.46	28.5±0.72
BiC	94.1±0.65	61.4±1.74	52.9±0.64	48.9±0.54	41.7±0.74	33.8±0.40
DER++	95.3±0.69	66.0±1.20	53.7±1.30	46.6±1.44	35.8±0.77	30.5±0.47
Co^{2}L		65.6				
CCG	97.3	70.1				
HAT	81.9±3.74	62.7±1.45	41.1±0.93	25.6±0.51	38.5±1.85	29.8±0.65
HyperNet	56.6±4.85	53.4±2.19	30.2±1.54	18.7±1.10	7.9±0.69	5.3±0.50
Sup	70.1±1.51	62.4±1.45	44.6±0.44	34.7±0.30	41.8±1.50	36.5±0.36
PR-Ent	74.1	61.9	45.2			
HAT+CSI	94.4±0.26	87.8±0.71	63.3±1.00	54.6±0.92	45.7±0.26	47.1±0.18
Sup+CSI	80.7±2.71	86.0±0.41	65.1±0.39	60.2±0.51	48.9±0.25	45.7±0.76
HAT+CSI+c	96.9±0.30	88.0±0.48	65.2±0.71	58.0±0.45	51.7±0.37	47.6±0.32
Sup+CSI+c	81.0±2.30	87.3±0.37	65.2±0.37	60.5±0.64	49.2±0.28	46.2±0.53

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. We note that Sup works very poorly on M-5T, but Sup+CSI improved it drastically, although still very weak compared to HAT+CSI.

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 α_k^* and β_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 the calibrated methods as HAT+CSI+c and Sup+CSI+c. They are trained by using the memory buffer of the same size as the replay methods (see Sec. 4.2). Tab. 3 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 α_k and β_k in Appendix E.

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 Tab. 4. Our systems again outperform the baselines by large margins on more challenging datasets (e.g., CIFAR100 and Tiny-ImageNet).

Table 4: TIL (WP) results of 3 best performing baselines and our methods. The full results are given in Appendix F. The calibrated versions (+c) of our methods are omitted as calibration does not affect TIL performances.

Method	M-5T	C10-5T	C100-10T	C100-20T	T-5T	T-10T
DER++	99.7±0.08	92.0±0.54	84.0±9.43	86.6±9.44	57.4±1.31	60.0±0.74
HAT	99.9±0.02	96.7±0.18	84.0±0.23	85.0±0.98	61.2±0.72	63.8±0.41
Sup	99.6±0.01	96.6±0.21	87.9±0.27	91.6±0.15	64.3±0.24	68.4±0.22
HAT+CSI	99.9±0.00	98.7±0.06	92.0±0.37	94.3±0.06	68.4±0.16	72.4±0.21
Sup+CSI	99.0±0.08	98.7±0.07	93.0±0.13	95.3±0.20	65.9±0.25	74.1±0.28

4.5 Implications for Existing CL Methods, Open-World Learning and Future Research

Implication for regularization and replay methods. Regularization-based (exemplar-free) methods try to protect important information of previous tasks to prevent CF. However, since the training of each task is not based on an OOD detection model, TP will be weak, which causes difficulty for *inter-task class separation* (ICS) and thus low CL accuracy. Replay-based methods are better as the replay data from previous tasks can be naturally regarded as OOD data for the current task, then a better OOD model is built, which improves TP. However, since the replay data is small. the OOD model is sub-optimal, especially for earlier tasks because their training cannot see any future task data. But for both approaches, it will be beneficial to train an OOD model for each task, e.g., [76].

Implication for open-world learning. Since our theory says that CL needs OOD detection, and OOD detection is also the first step in open-world learning (OWL), CL and OWL naturally work together to achieve *self-motivated open-world continual learning* [5] for autonomous learning or AI autonomy. That is, the AI agent can continually discover new tasks (OOD detection) and incrementally learn the tasks (CL) all on its own with no involvement of human engineers. Further afield, this work is also related to curiosity-driven self-supervised learning [77] in reinforcement learning and 3D navigation.

Limitation and future work. The proposed theory provides a principled guidance on what needs to be done in order to achieve good CIL results, but it gives no guidance on how to do it. Although two example techniques are presented and evaluated, they are empirical. There are many options to define WP and TP (or OOD). An idea in [40] may be helpful in this regard. [40] argues that a continual learner should learn *holistic feature representations* of the input data, meaning to learn as many features as possible from the input data. The rationale is that if the system can learn all possible features from each task, then a future task does not have to learn those shared/intersecting features by modifying the parameters, which will result in less CF and also better ICS. A full representation of the IND data also improves OOD detection because the OOD score of a data point is basically the distance between the data point and the IND distribution. Only capturing a subset of features (e.g., by cross entropy) will result in poor OOD detection [78] because those missing features may be necessary to separate IND and some OOD data. In our future work, we will study how to optimize WP and TP/OOD and find the necessary conditions for them to do well.

5 Conclusion

This paper proposed a theoretical study on how to solve the highly challenging continual learning (CL) problem. *class incremental learning* (CIL) (the other popular CL setting is *task incremental learning* (TIL)). The theoretical result provides a principled guidance for designing better CIL algorithms. The paper first decomposed the CIL prediction into *within-task prediction* (WP) and *task-id prediction* (TP). WP is basically TIL. The paper further theoretically demonstrated that TP is correlated with *out-of-distribution* (OOD) detection. It then proved that a good performance of the two is both necessary and sufficient for good CIL performances. Based on the theoretical result, several new CIL methods were designed. They outperform strong baselines in CIL and also in TIL by a large margin. Finally, we also discussed the implications for existing CL techniques and open-world learning.

Acknowledgments

The work of Gyuhak Kim, Zixuan Ke and Bing Liu was supported in part by a research contract from KDDI, two NSF grants (IIS-1910424 and IIS-1838770), and a DARPA contract HR001120C0023.

References

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

[2] Gido M van de Ven and Andreas S Tolias. Three scenarios for continual learning. *arXiv preprint arXiv:1904.07734*, 2019.

[3] Joan Serrà, Dídac Surís, Marius Miron, and Alexandros Karatzoglou. Overcoming catastrophic forgetting with hard attention to the task. In *ICML*, 2018.

[4] Mitchell Wortsman, Vivek Ramanujan, Rosanne Liu, Aniruddha Kembhavi, Mohammad Rastegari, Jason Yosinski, and Ali Farhadi. Supermasks in superposition. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *NeurIPS*, 2020.

[5] Bing Liu, Eric Robertson, Scott Grigsby, and Sahisnu Mazumder. Self-initiated open world learning for autonomous ai agents. *Proceedings of AAAI Symposium on Designing Artificial Intelligence for Open Worlds*, 2021.

[6] Jihoon Tack, Sangwoo Mo, Jongheon Jeong, and Jinwoo Shin. Csi: Novelty detection via contrastive learning on distributionally shifted instances. In *NeurIPS*, 2020.

[7] Shiyu Liang, Yixuan Li, and R. Srikant. Enhancing the reliability of out-of-distribution image detection in neural networks. In *ICLR*, 2018.

[8] Sepideh Esmaeilpour, Bing Liu, Eric Robertson, and Lei Shu. Zero-shot out-of-distribution detection based on the pretrained model clip. In *Proceedings of the AAAI conference on artificial intelligence*, 2022.

[9] Mengyu Wang, Yijia Shao, Haowei Lin, Wenpeng Hu, and Bing Liu. Cmg: A class-mixed generation approach to out-of-distribution detection. *Proceedings of ECML/PKDD-2022*, 2022.

[10] Anastasia Pentina and Christoph Lampert. A pac-bayesian bound for lifelong learning. In *International Conference on Machine Learning*, pages 991–999. PMLR, 2014.

[11] Ryo Karakida and Shotaro Akaho. Learning curves for continual learning in neural networks: Self-knowledge transfer and forgetting. In *International Conference on Learning Representations*, 2022.

[12] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526, 2017.

[13] Zhizhong Li and Derek Hoiem. Learning Without Forgetting. In *ECCV*, pages 614–629. Springer, 2016.

[14] Heechul Jung, Jeongwoo Ju, Minju Jung, and Junmo Kim. Less-forgetting learning in deep neural networks. *arXiv preprint arXiv:1607.00122*, 2016.

[15] Raffaello Camoriano, Giulia Pasquale, Carlo Ciliberto, Lorenzo Natale, Lorenzo Rosasco, and Giorgio Metta. Incremental robot learning of new objects with fixed update time. In *ICRA*, 2017.

[16] Friedemann Zenke, Ben Poole, and Surya Ganguli. Continual learning through synaptic intelligence. In *ICML*, pages 3987–3995, 2017.

[17] Hippolyt Ritter, Aleksandar Botev, and David Barber. Online structured laplace approximations for overcoming catastrophic forgetting. In *NeurIPS*, 2018.

[18] Jonathan Schwarz, Jelena Luketina, Wojciech M Czarnecki, Agnieszka Grabska-Barwinska, Yee Whye Teh, Razvan Pascanu, and Raia Hadsell. Progress & compress: A scalable framework for continual learning. *arXiv preprint arXiv:1805.06370*, 2018.

[19] Ju Xu and Zhanxing Zhu. Reinforced continual learning. In *NeurIPS*, 2018.

[20] Francisco M Castro, Manuel J Marín-Jiménez, Nicolás Guil, Cordelia Schmid, and Karteek Alahari. End-to-end incremental learning. In *ECCV*, pages 233–248, 2018.

[21] Wenpeng Hu, Zhou Lin, Bing Liu, Chongyang Tao, Zhengwei Tao, Jinwen Ma, Dongyan Zhao, and Rui Yan. Overcoming catastrophic forgetting for continual learning via model adaptation. In *ICLR*, 2019.

[22] Prithviraj Dhar, Rajat Vikram Singh, Kuan-Chuan Peng, Ziyan Wu, and Rama Chellappa. Learning without memorizing. In *CVPR*, 2019.

[23] Kibok Lee, Kimin Lee, Jinwoo Shin, and Honglak Lee. Overcoming catastrophic forgetting with unlabeled data in the wild. In *CVPR*, 2019.

[24] Hongjoon Ahn, Sungmin Cha, Donggyu Lee, and Taesup Moon. Uncertainty-based continual learning with adaptive regularization. In *NeurIPS*, 2019.

[25] Yu Liu, Sarah Parisot, Gregory Slabaugh, Xu Jia, Ales Leonardis, and Tinne Tuytelaars. More classifiers, less forgetting: A generic multi-classifier paradigm for incremental learning. In *ECCV*, pages 699–716. Springer International Publishing, 2020.

[26] Fei Zhu, Xu-Yao Zhang, Chuang Wang, Fei Yin, and Cheng-Lin Liu. Prototype augmentation and self-supervision for incremental learning. In *CVPR*, 2021.

[27] Andrei A Rusu, Neil C Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. Progressive neural networks. *arXiv preprint arXiv:1606.04671*, 2016.

[28] David Lopez-Paz and Marc'Aurelio Ranzato. Gradient Episodic Memory for Continual Learning. In *NeurIPS*, pages 6470–6479, 2017.

[29] Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, and Christoph H Lampert. iCaRL: Incremental classifier and representation learning. In *CVPR*, pages 5533–5542, 2017.

[30] Arslan Chaudhry, Marc'Aurelio Ranzato, Marcus Rohrbach, and Mohamed Elhoseiny. Efficient lifelong learning with a-gem. In *ICLR*, 2019.

[31] Saihui Hou, Xinyu Pan, Chen Change Loy, Zilei Wang, and Dahua Lin. Learning a unified classifier incrementally via rebalancing. In *CVPR*, pages 831–839, 2019.

[32] Yue Wu, Yinpeng Chen, Lijuan Wang, Yuancheng Ye, Zicheng Liu, Yandong Guo, and Yun Fu. Large scale incremental learning. In *CVPR*, 2019.

[33] David Rolnick, Arun Ahuja, Jonathan Schwarz, Timothy P. Lillicrap, and Greg Wayne. Experience replay for continual learning. In *NeurIPS*, 2019.

[34] Pietro Buzzega, Matteo Boschini, Angelo Porrello, Davide Abati, and Simone Calderara. Dark experience for general continual learning: a strong, simple baseline. In *NeurIPS*, 2020.

[35] Jathushan Rajasegaran, Munawar Hayat, Salman Khan, Fahad Shahbaz Khan, Ling Shao, and Ming-Hsuan Yang. An adaptive random path selection approach for incremental learning, 2020.

[36] Yaoyao Liu, Bernt Schiele, and Qianru Sun. Adaptive aggregation networks for class-incremental learning. In *CVPR*, 2021.

[37] Hyuntak Cha, Jaeho Lee, and Jinwoo Shin. Co2l: Contrastive continual learning. In *ICCV*, 2021.

[38] Shipeng Yan, Jiangwei Xie, and Xuming He. Der: Dynamically expandable representation for class incremental learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3014–3023, 2021.

[39] Liyuan Wang, Xingxing Zhang, Kuo Yang, Longhui Yu, Chongxuan Li, Lanqing Hong, Shifeng Zhang, Zhenguo Li, Yi Zhong, and Jun Zhu. Memory replay with data compression for continual learning. *Proceedings of International Conference on Learning Representations (ICLR)*, 2022.

[40] Yiduo Guo, Bing Liu, and Dongyan Zhao. Online continual learning through mutual information maximization. In *International Conference on Machine Learning*, pages 8109–8126. PMLR, 2022.

[41] Alexander Gepperth and Cem Karaoguz. A bio-inspired incremental learning architecture for applied perceptual problems. *Cognitive Computation*, 8(5):924–934, 2016.

[42] Nitin Kamra, Umang Gupta, and Yan Liu. Deep Generative Dual Memory Network for Continual Learning. *arXiv preprint arXiv:1710.10368*, 2017.

[43] Hanul Shin, Jung Kwon Lee, Jaehong Kim, and Jiwon Kim. Continual learning with deep generative replay. In *NIPS*, pages 2994–3003, 2017.

[44] Chenshen Wu, Luis Herranz, Xialei Liu, Joost van de Weijer, Bogdan Raducanu, et al. Memory replay gans: Learning to generate new categories without forgetting. In *NeurIPS*, 2018.

[45] Ari Seff, Alex Beatson, Daniel Suo, and Han Liu. Continual learning in generative adversarial nets. *arXiv preprint arXiv:1705.08395*, 2017.

[46] Ronald Kemker and Christopher Kanan. FearNet: Brain-Inspired Model for Incremental Learning. In *ICLR*, 2018.

[47] Mohammad Rostami, Soheil Kolouri, and Praveen K. Pilly. Complementary learning for overcoming catastrophic forgetting using experience replay. In *IJCAI*, 2019.

[48] Oleksiy Ostapenko, Mihai Puscas, Tassilo Klein, Patrick Jahnichen, and Moin Nabi. Learning to remember: A synaptic plasticity driven framework for continual learning. In *CVPR*, pages 11321–11329, 2019.

[49] Guanxiong Zeng, Yang Chen, Bo Cui, and Shan Yu. Continuous learning of context-dependent processing in neural networks. *Nature Machine Intelligence*, 2019.

[50] Yiduo Guo, Wenpeng Hu, Dongyan Zhao, and Bing Liu. Adaptive orthogonal projection for batch and online continual learning. In *Proceedings of AAAI-2022*, 2022.

[51] Arslan Chaudhry, Naeemullah Khan, Puneet K. Dokania, and Philip H. S. Torr. Continual learning in low-rank orthogonal subspaces, 2020.

[52] Zixuan Ke, Bing Liu, and Xingchang Huang. Continual learning of a mixed sequence of similar and dissimilar tasks. In *NeurIPS*, 2020.

[53] Arun Mallya and Svetlana Lazebnik. PackNet: Adding Multiple Tasks to a Single Network by Iterative Pruning. *arXiv preprint arXiv:1711.05769*, 2017.

[54] Ching-Yi Hung, Cheng-Hao Tu, Cheng-En Wu, Chien-Hung Chen, Yi-Ming Chan, and Chu-Song Chen. Compacting, picking and growing for unforgetting continual learning. In *NeurIPS*, volume 32, 2019.

[55] Johannes von Oswald, Christian Henning, João Sacramento, and Benjamin F Grewe. Continual learning with hypernetworks. *ICLR*, 2020.

[56] Jaehong Yoon, Saehoon Kim, Eunho Yang, and Sung Ju Hwang. Scalable and order-robust continual learning with additive parameter decomposition. In *ICLR*, 2020.

[57] Pravendra Singh, Vinay Kumar Verma, Pratik Mazumder, Lawrence Carin, and Piyush Rai. Calibrating cnns for lifelong learning. *NeurIPS*, 2020.

[58] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *CVPR*, 2020.

[59] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *ICML*, 2020.

[60] Saikiran Bulusu, Bhavya Kailkhura, Bo Li, Pramod K Varshney, and Dawn Song. Anomalous instance detection in deep learning: A survey. *arXiv preprint arXiv:2003.06979*, 2020.

[61] Chuanxing Geng, Sheng-jun Huang, and Songcan Chen. Recent advances in open set recognition: A survey. *IEEE transactions on pattern analysis and machine intelligence*, 2020.

[62] Jathushan Rajasegaran, Salman Khan, Munawar Hayat, Fahad Shahbaz Khan, and Mubarak Shah. itaml: An incremental task-agnostic meta-learning approach. In *CVPR*, 2020.

[63] Davide Abati, Jakub Tomczak, Tijmen Blankevoort, Simone Calderara, Rita Cucchiara, and Ehteshami Bejnordi. Conditional channel gated networks for task-aware continual learning. In *CVPR*, pages 3931–3940, 2020.

[64] Rahaf Aljundi, Punarjay Chakravarty, and Tinne Tuytelaars. Expert gate: Lifelong learning with a network of experts. In *CVPR*, 2017.

[65] Christian Henning, Maria Cervera, Francesco D'Angelo, Johannes Von Oswald, Regina Traber, Benjamin Ehret, Seijin Kobayashi, Benjamin F Grewe, and João Sacramento. Posterior meta-replay for continual learning. *NeurIPS*, 34, 2021.

[66] Sebastian Lee, Sebastian Goldt, and Andrew Saxe. Continual learning in the teacher-student setup: Impact of task similarity. In *International Conference on Machine Learning*, pages 6109–6119. PMLR, 2021.

[67] Mehdi Abbana Bennani, Thang Doan, and Masashi Sugiyama. Generalisation guarantees for continual learning with orthogonal gradient descent. *Lifelong Learning Workshop at the ICML*, 2020.

[68] Jihwan Bang, Heesu Kim, YoungJoon Yoo, Jung-Woo Ha, and Jonghyun Choi. Rainbow memory: Continual learning with a memory of diverse samples. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8218–8227, 2021.

[69] Yaoyao Liu, Yuting Su, An-An Liu, Bernt Schiele, and Qianru Sun. Mnemonics training: Multi-class incremental learning without forgetting. In *CVPR*, 2020.

[70] Weitang Liu, Xiaoyun Wang, John Owens, and Yixuan Li. Energy-based out-of-distribution detection. *Advances in Neural Information Processing Systems*, 2020.

[71] Kimin Lee, Kibok Lee, Honglak Lee, and Jinwoo Shin. A simple unified framework for detecting out-of-distribution samples and adversarial attacks. *Advances in neural information processing systems*, 31, 2018.

[72] Prannay Khosla, Piotr Teterwak, Chen Wang, Aaron Sarna, Yonglong Tian, Phillip Isola, Aaron Maschinot, Ce Liu, and Dilip Krishnan. Supervised contrastive learning. *arXiv preprint arXiv:2004.11362*, 2020.

[73] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012.

[74] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

[75] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *ICLR*, 2015.

[76] Gyuhak Kim, Zixuan Ke, and Bing Liu. A multi-head model for continual learning via out-of-distribution replay. *arXiv preprint arXiv:2208.09734*, 2022.

[77] Deepak Pathak, Pulkit Agrawal, Alexei A Efros, and Trevor Darrell. Curiosity-driven exploration by self-supervised prediction. In *International conference on machine learning*, pages 2778–2787. PMLR, 2017.

[78] Wenpeng Hu, Mengyu Wang, Qi Qin, Jinwen Ma, and Bing Liu. Hrn: A holistic approach to one class learning. *Advances in Neural Information Processing Systems*, 33:19111–19124, 2020.

[79] Vivek Ramanujan, Mitchell Wortsman, Aniruddha Kembhavi, Ali Farhadi, and Mohammad Rastegari. What's hidden in a randomly weighted neural network? In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11893–11902, 2020.

[80] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *CVPR*, 2015.

[81] Dan Hendrycks, Mantas Mazeika, Saurav Kadavath, and Dawn Song. Using self-supervised learning can improve model robustness and uncertainty. In *NeurIPS*, pages 15663–15674, 2019.

[82] Yang You, Igor Gitman, and Boris Ginsburg. Large batch training of convolutional networks. *arXiv preprint arXiv:1708.03888*, 2017.

[83] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*, 2016.

[84] Gyuhak Kim, Sepideh Esmaeilpour, Changnan Xiao, and Bing Liu. Continual learning based on ood detection and task masking. In *CVPR 2022 Workshop on Continual Learning*, 2022.

Checklist

The checklist follows the references. Please read the checklist guidelines carefully for information on how to answer these questions. For each question, change the default **[TODO]** to [Yes] , [No] , or [N/A] . You are strongly encouraged to include a **justification to your answer**, either by referencing the appropriate section of your paper or providing a brief inline description. For example:

- Did you include the license to the code and datasets? [Yes] See Section ??.
- Did you include the license to the code and datasets? [No] The code and the data are proprietary.
- Did you include the license to the code and datasets? [N/A]

Please do not modify the questions and only use the provided macros for your answers. Note that the Checklist section does not count towards the page limit. In your paper, please delete this instructions block and only keep the Checklist section heading above along with the questions/answers below.

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] See Sec. 4.5
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] In Appendix
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes] See Sec. 3.1.
   (b) Did you include complete proofs of all theoretical results? [Yes] See Appendix.
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes]
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See Sec. 4.2 and Appendix.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See Appendix
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [No]
   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

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

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

□

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)

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

and ii)

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

□

A.3 Proof of Theorem 2.

Proof. i) Assume $x \in \mathbf{X}_{k_0}$.

For $k = k_0$, we have

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

For $k \neq k_0$, we have

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

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

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

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,

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

□

A.4 Proof of Theorem 3.

Proof. Using Theorem 1 and 2,

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

□

A.5 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,

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

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,

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

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,

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

□

B Additional Results and Explanation Regarding Table 1 in the Main Paper

In Sec. 4.3, we showed that a better OOD detection improves CIL performance. For the post-processing method ODIN, we only reported the results on C100-10T due to space limitations. Tab. 5 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 τ_k using a validation set. This is because the softmax in Eq. 10 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 is larger than the other tasks in PASS. Therefore, larger τ_1 and smaller τ_k, for $k > 1$, are chosen to compensate the relative values.

Table 5: Performance comparison between the original output and output post-processed with OOD detection technique ODIN. Note that ODIN is not applicable to iCaRL and Mnemonics as they are not based on softmax but some distance functions. The result for C100-10T are reported in the main paper.

		M-5T		C10-5T		C100-20T		T-5T		T-10T	
Method	OOD	AUC	CIL	AUC	CIL	AUC	CIL	AUC	CIL	AUC	CIL
OWM	Original	99.13	95.81	81.33	51.79	71.90	24.15	58.49	10.00	59.48	8.57
	ODIN	98.86	95.16	71.72	40.65	68.52	23.05	58.46	10.77	59.38	9.52
MUC	Original	92.27	74.90	79.49	52.85	66.20	14.19	68.42	33.57	62.63	17.39
	ODIN	92.67	75.71	79.54	53.22	65.72	14.11	68.32	33.45	62.17	17.27
PASS	Original	98.74	76.58	66.51	47.34	70.26	24.99	65.18	28.40	63.27	19.07
	ODIN	90.40	74.33	63.08	35.20	69.81	21.83	65.93	29.03	62.73	17.78
LwF	Original	99.19	85.46	89.39	54.67	89.84	44.33	78.20	32.17	79.43	24.28
	ODIN	98.52	90.39	88.94	63.04	88.68	47.56	76.83	36.20	77.02	28.29
BiC	Original	99.40	94.11	90.89	61.41	89.46	48.92	80.17	41.75	80.37	33.77
	ODIN	98.57	95.14	91.86	64.29	87.89	47.40	74.54	37.40	76.27	29.06
DER++	Original	99.78	95.29	90.16	66.04	85.44	46.59	71.80	35.80	72.41	30.49
	ODIN	99.09	94.96	87.08	63.07	87.72	49.26	73.92	37.87	72.91	32.52
HAT	Original	94.46	81.86	82.47	62.67	75.35	25.64	72.28	38.46	71.82	29.78
	ODIN	94.56	82.06	82.45	62.60	75.36	25.84	72.31	38.61	71.83	30.01
HyperNet	Original	85.83	56.55	78.54	53.40	72.04	18.67	54.58	7.91	55.37	5.32
	ODIN	86.89	64.31	79.39	56.72	73.89	23.8	54.60	8.64	55.53	6.91
Sup	Original	90.70	70.06	79.16	62.37	81.14	34.70	74.13	41.82	74.59	36.46
	ODIN	90.68	69.70	82.38	62.63	81.48	36.35	73.96	41.10	74.61	36.46

C Definitions of TP

As noted in the main paper, the class prediction in Eq. 2 varies by definition of WP and TP. The precise definition of WP and TP depends on implementation. Due to this subjectivity, we follow the prediction method as the existing methods 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.

Theorem 5 (Extension of Theorem 2)**.** *i) If $H_{TP}(x) \leq \delta$, 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 $H_{OOD,k}(x) \leq \max(\delta/\tau_k, -\log(1-(1-e^{-\delta})^{1/\tau_k}), \forall\, k = 1, \ldots, T$.*

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

In Theorem 5 (proof appears later), we can observe that δ/τ_k decreases with the increase of τ_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 τ_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 $\tau_1, \ldots, \tau_T$ 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 τ_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 σ 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.

Table 6: Average classification accuracy. The results are based on class prediction method defined with WP and TP in Eq. 12 and Eq. 13, respectively. The results can improve by finding optimal temperature scaling parameters.

Method	M-5T	C10-5T	C100-10T	C100-20T	T-5T	T-10T
OWM	95.1±0.11	40.6±0.47	28.6±0.82	22.9±0.32	10.4±0.54	9.2±0.35
MUC	75.7±0.51	53.2±1.32	30.6±1.21	14.0±0.12	33.1±0.18	17.2±0.13
PASS†	64.5±2.64	33.6±0.71	18.5±1.85	20.8±0.85	21.4±0.44	13.0±0.55
LwF	90.4±1.18	63.0±0.34	51.9±0.88	47.5±0.62	35.9±0.32	27.8±0.29
iCaRL*	87.4±4.89	65.3±0.83	52.9±0.39	48.2±0.70	34.8±0.34	27.3±0.17
Mnemonics†*	91.8±1.03	65.6±1.55	50.7±0.72	47.9±0.71	36.3±0.30	27.7±0.78
BiC	95.1±0.47	65.5±0.81	50.8±0.69	47.2±0.71	37.0±0.58	29.1±0.34
DER++	94.9±0.50	63.1±1.12	54.6±1.21	48.9±1.18	37.4±0.72	32.1±0.44
HAT	82.1±3.77	62.6±1.31	41.5±0.80	25.9±0.56	38.9±1.62	30.1±0.52
HyperNet	64.3±2.98	56.7±1.23	32.4±1.07	24.5±1.12	8.9±0.58	7.0±0.52
Sup	69.7±0.97	62.6±1.11	46.8±0.34	36.0±0.32	41.5±1.17	35.7±0.40
HAT+CSI	88.7±1.27	85.2±0.92	62.9±1.07	53.6±0.84	47.0±0.38	46.2±0.30
Sup+CSI	64.9±1.95	87.4±0.40	66.6±0.23	60.5±0.89	47.7±0.30	46.3±0.30
HAT+CSI+c	93.4±0.43	85.2±0.94	63.6±0.69	55.4±0.79	51.4±0.38	46.5±0.26
Sup+CSI+c	62.2±3.49	86.2±0.79	67.0±0.14	60.4±1.04	48.2±0.35	46.1±0.32

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

$$\mathbf{P}(x \in \mathbf{X}_{k,j} | x \in \mathbf{X}_k, D) = \frac{e^{f(x)_{kj}/\nu_k}}{\sum_j e^{f(x)_{kj}/\nu_k}}, \tag{12}$$

where ν_k is a temperature scaling parameter for task k, and the TP is

$$\mathbf{P}(x \in \mathbf{X}_k | D) = \frac{\mathbf{P}'_k(x \in \mathbf{X}_k | D)}{\sum_k \mathbf{P}'_k(x \in \mathbf{X}_k | D)}, \tag{13}$$

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 τ_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 τ and ν 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 memory buffer. The CIL results for the new prediction method is in Tab. 6.

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

For $k = k_0$, we have

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

For $k \neq k_0$, we have

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

ii) Assume $x \in \mathbf{X}_{k_0}$.

For $k = k_0$, by $H_{OOD,k_0}(x) \leq \delta_{k_0}$, 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

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

which means

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

Therefore, we have

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

Hence,

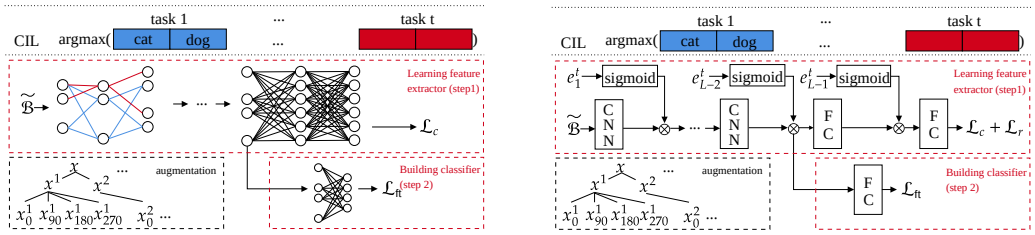
$$\begin{aligned}
H_{TP}(x) &= -\log \mathbf{P}(x \in \mathbf{X}_{k_0}|D) \\
&\leq -\log \frac{e^{-\delta_{k_0}/\tau_{k_0}}}{1 - (1 - e^{-\delta_{k_0}})^{1/\tau_{k_0}}} \cdot \frac{1}{1 + \frac{\sum_k (1 - e^{-\delta_k})^{1/\tau_k}}{1 - (1 - e^{-\delta_{k_0}})^{1/\tau_{k_0}}}} \\
&= \frac{\delta_{k_0}}{\tau_{k_0}}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

□

D Details of HAT, Sup, and CSI

We have proposed two highly effective new CIL methods, HAT+CSI and Sup+CSI, by integrating the existing parameter isolation based continual learning (CL) method HAT [3] or Sup [4] with the strong OOD detection method CSI [6]. We replaced the training loss of HAT and Sup by that of CSI while applying the continual learning techniques of the respective method. In this section, we overview Sup, HAT, and CSI, and explain how to train them continually. Figure 1 shows the overall training frameworks of Sup+CSI and HAT+CSI.

Denote feature extractor by h, classifier by f, and the parameters by $\mathbf{W}$. In the main paper, we denote the output of task k by $f(x)_k$ for both a single-head or multi-head method (e.g., Eq. 9) for consistency. In this section, we use $f(x, k)$ to indicate the output of task k to be more explicit as both HAT and Sup are multi-head methods (one head for each task) designed for task incremental learning (TIL).



(a) (b)

Figure 1: Overview of prediction and training framework of Sup+CSI and HAT+CSI. (a) Sup+CSI: The CIL prediction is made by taking argmax over the concatenated output values from each task. In training the network, the training batch is augmented to give different views of samples for contrastive training in the OOD detection algorithm CSI. The training consists of two steps following CSI. The first step is learning the feature extractor. In this step, the Edge Popup algorithm [79] is applied to find a sparse network for each task. The sparse networks, which are indicated by edges of different colors in the diagram. The second step fine-tunes the classifier only the fixed feature extractor. (b) HAT+CSI: The CIL prediction is also made by argmax over the concatenated output from each task as Sup+CSI method. Due to the OOD detection algorithm CSI, the overall training process is similar to Sup+CSI except that it applies the hard attention algorithm [3]. In training feature extractor, task embeddings are applied to find hard masks at each layer. Then given the learned feature representations, fine-tunes the classifier in step 2.

D.1 Sup

SupSup (Sup) [4] trains supermasks by Edge Popup algorithm [79]. More precisely, given initial $\mathbf{W}$, find binary masks $\mathbf{M}_k$ for task k to minimize the cross-entropy loss

$$\mathcal{L} = -\frac{1}{|\mathbf{X}_k|} \sum \log p(y|x, k), \tag{14}$$

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

$$p(y|x, k) = f(h(x; \mathbf{W} \otimes \mathbf{M}_k)), \tag{15}$$

where $\otimes$ indicates an 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 Figure 1(a).

Given the task-id k of a test instance at inference, the system (which is referred as Sup GG in the original Sup paper) uses the task-specific mask $\mathbf{M}_k$ to obtain the classification output. By integrating the OOD detection method, CSI, during training, Sup+CSI does not require to know the task-id of test instance, which makes Sup+CSI applicable to CIL (class incremental learning).

D.2 HAT

We now discuss the hard attention (mask) mechanism of HAT [3]. It finds binary masks a_l^k for each layer l and task k, and uses them to block/unblock information flow at forward and backward pass. More precisely, the hard attention is defined as

$$a_l^k = \sigma(s e_l^k), \tag{16}$$

where σ is the sigmoid, s is a positive constant, and e_l^k is a learnable embedding. To approximate the binary mask, the system uses a large s value. The attention is applied to the output at each layer as

$$h_l^{'} = a_l^k \otimes h_l, \tag{17}$$

where $\otimes$ is an element-wise product, and

$$h_l = \text{ReLU}(W_l h_{l-1} + b_l). \tag{18}$$

The neurons with attention value 1 is important for task k while those with zero attention value are not necessary for the task, and thus they can be freely changed without affecting the output value

$h_l^{'}$. The system needs to know which neurons are important to protect the previous knowledge from forgetting. Denote the accumulated attentions of all previous tasks by

$$a_l^{<k} = \max(a_l^{<k-1}, a_l^{k-1}), \tag{19}$$

where a_l^0 is the zero vector and $\max$ is an element-wise maximum. The gradients of parameters corresponding to important neurons is modified as

$$\nabla w'_{ij,l} = \left(1 - \min\left(a_{i,l}^{<k}, a_{j,l-1}^{<k}\right)\right) \nabla w_{ij,l}, \tag{20}$$

where $a_{i,l}^{<k}$ is the i'th unit of $a_l^{<k}$ and $l = 1, \cdots, L-1$. The hard attention is not applied to the last layer L since it is a task-specific classification layer.

To encourage sparsity in a_l^k, the system uses regularization as

$$\mathcal{L}_r = \lambda_k \frac{\sum_l \sum_i a_{i,l}^k \left(1 - a_{i,l}^{<k}\right)}{\sum_l \sum_i \left(1 - a_{i,l}^{<k}\right)}, \tag{21}$$

where λ_k is a hyper-parameter. The system minimizes the loss

$$\mathcal{L} = \mathcal{L}_{ce} + \mathcal{L}_r, \tag{22}$$

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

D.3 CSI

We now explain the OOD detection method CSI, and how to incorporate it in HAT and Sup. CSI is based on contrastive learning [59, 58] and data augmentation due to their excellent performance [6]. Since this section focuses on how to learn a single task based on OOD detection, we omit the task-id unless necessary. The OOD training process is similar to that of contrastive learning. It consists of two steps: 1) learning the feature representation by the composite $g \circ h$, where h is a feature extractor and g is a projection to contrastive representation, and 2) learning a linear classifier f mapping the feature representation of h to the label space. This two step training process is outlined in Figure 1(a) and (b). In the following, we describe the training process: contrastive learning for feature representation learning (1), and OOD classifier building (2). We then explain how to make a prediction based on an ensemble method to further improve prediction.

D.3.1 Contrastive Loss for Feature Learning.

This is step 1. 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 and each version then goes through *three initial augmentations* (horizontal flip, color changes, and Inception crop [80]) 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 [81, 6], 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 $z_x = g(h(x,t))/\|g(h(x,t))\|$, where t is the current task. In learning, we minimize the supervised contrastive loss [72] of task t.

$$\mathcal{L}_c = \frac{1}{8N} \sum_{x \in \tilde{\mathcal{B}}} \frac{-1}{|P(x)|} \sum_{p \in P(x)} \log \frac{\exp(z_x \cdot z_p/\tau)}{\sum_{x' \in \tilde{\mathcal{B}}(x)} \exp(z_x \cdot z_{x'}/\tau)}, \tag{23}$$

where τ 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, a protection is needed to prevent catastrophic forgetting. HAT and Sup use their respective technique to protect their feature extractor from forgetting. Therefore, the losses $\mathcal{L}$ of Eq. 14 and $\mathcal{L}_{ce}$ of Eq. 22 are replaced by Eq. 23 while the forgetting prevention mechanisms still hold.

D.3.2 Learning the Classifier.

This is step 2. Given the feature extractor h trained with the loss in Eq. 23, we *freeze* h and only *fine-tune* the linear classifier f, which is trained to predict the classes of task t *and* the augmented rotation classes. f maps the feature representation to the label space in $\mathcal{R}^{4|\mathcal{C}^t|}$, where 4 is the number of rotation classes including the original data with 0° rotation and $|\mathcal{C}^t|$ is the number of original classes in task t. We minimize the cross-entropy loss,

$$\mathcal{L}_{\text{ft}} = -\frac{1}{|\tilde{\mathcal{B}}|} \sum_{(x,y)\in\tilde{\mathcal{B}}} \log \tilde{p}(y|x,t), \tag{24}$$

where ft indicates fine-tune, and

$$\tilde{p}(y|x,t) = \text{softmax}\left(f(h(x,t))\right) \tag{25}$$

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

D.3.3 Ensemble Class Prediction.

We describe how to predict a label $y \in \mathcal{C}^t$ (TIL) and $y \in \mathcal{C}$ (CIL) ($\mathcal{C}$ is the set of original classes of all tasks). We assume all tasks have been learned and their models are protected by masks.

We discuss the prediction of class label y for a test sample x in the TIL setting first. Note that the network $f \circ h$ in Eq. 25 returns logits for rotation classes (including the original task classes). Note also for each original class label $j_k \in \mathcal{C}^k$ (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 $f_{j_k}(h(x,k))$ for each class $j_k \in \mathcal{C}^k$ of task k,

$$f(h(x,k))_{j_k} = \frac{1}{4} \sum_{\text{deg}} f(h(x_{\text{deg}},k))_{j_k,\text{deg}}. \tag{26}$$

We use Eq. 9 to make the CIL class prediction, where the final class prediction is made as

$$\hat{y} = \arg\max \bigoplus_i f(h(x,i)). \tag{27}$$

E Output Calibration

In this section, we discuss the output calibration technique used in Sec. 4.4 to improve the final prediction accuracy. Even if an OOD detection of each task was perfect (i.e. the model accept and reject IND and OOD samples perfectly), the system could make 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 α_k and shifting β_k for each task. The optimal parameters $(\alpha_k, \beta_k) \in R \times R$ can be found by solving optimization problem using samples in 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 Tth task, we find optimal calibration parameters by minimizing the cross-entropy loss,

$$\mathcal{L} = -\frac{1}{|\mathcal{M}|} \sum_{(x,y)\in\mathcal{M}} \log p(y|x) \tag{28}$$

Table 7: The TIL results of all the systems. The calibrated versions (+c) of our methods are omitted as calibration does not affect TIL performance. Exemplar-free methods are italicized.

Method	M-5T	C10-5T	C100-10T	C100-20T	T-5T	T-10T
OWM	99.7±0.03	85.0±0.07	59.6±0.83	65.4±0.48	22.4±0.87	28.1±0.55
MUC	99.9±0.02	95.1±0.10	77.3±0.83	73.4±9.16	55.9±0.26	47.2±0.22
PASS†	99.5±0.14	83.8±0.68	72.1±0.70	76.8±0.32	49.9±0.56	46.5±0.39
LwF	99.9±0.09	95.2±0.30	86.2±1.00	89.0±0.45	56.4±0.48	55.3±0.35
iCaRL	99.9±0.08	94.9±0.34	84.2±1.04	85.7±0.68	54.5±0.29	52.7±0.37
Mnemonics†*	99.9±0.03	94.5±0.46	82.3±0.30	86.2±0.46	54.8±0.16	52.9±0.66
BiC	99.9±0.03	95.4±0.35	84.6±0.48	88.7±0.19	61.5±0.60	62.2±0.45
DER++	99.7±0.08	92.0±0.54	84.0±9.43	86.6±9.44	57.4±1.31	60.0±0.74
HAT	99.9±0.02	96.7±0.18	84.0±0.23	85.0±0.98	61.2±0.72	63.8±0.41
HyperNet	99.7±0.04	94.6±0.37	76.8±1.22	83.5±0.98	23.9±0.60	28.0±0.69
Sup	99.6±0.01	96.6±0.21	87.9±0.27	91.6±0.15	64.3±0.24	68.4±0.22
HAT+CSI	99.9±0.00	98.7±0.06	92.0±0.37	94.3±0.06	68.4±0.16	72.4±0.21
Sup+CSI	99.0±0.08	98.7±0.07	93.0±0.13	95.3±0.20	65.9±0.25	74.1±0.28

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

$$\text{softmax} \bigoplus [\alpha_k f(x)_k + \beta_k] \tag{29}$$

where $\bigoplus$ indicates the concatenation and $f(x)_k$ is the output of task k as Eq. 9. Given the optimal parameters (α^*, β^*), we make final prediction as

$$\hat{y} = \arg\max \bigoplus [\alpha_k^* f(x)_k + \beta_k^*] \tag{30}$$

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

F TIL (WP) Results

The TIL (WP) results of all the systems are reported in Tab. 7. 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 the strong CIL performance.

G 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 [6]. We use LARS [82] optimization with 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 [83] without restart as in [6, 59]. After training the feature extractor, we train the linear classifier for 100 epochs with SGD with 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.

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 batch size of 32 while HAT+CSI trains for 700 epochs with batch size of 256. The hard attention regularization penalty λ_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

Table 8: The number of parameters used at inference after learning the final task. The M after each value indicates millions.

Method	M-5T	C10-5T	C100-10T	C100-20T	T-5T	T-10T
OWM	5.27M	5.27M	5.36M	5.36M	5.46M	5.46M
MUC	1.06M	11.19M	45.06M	45.06M	45.47M	45.47M
PASS	1.03M	11.17M	44.76M	44.76M	44.86M	44.86M
LwF	1.03M	11.17M	44.76M	44.76M	44.86M	44.86M
iCaRL	1.03M	11.17M	44.76M	44.76M	44.86M	44.86M
Mnemonics	1.03M	11.17M	44.76M	44.76M	44.86M	44.86M
BiC	1.03M	11.17M	44.76M	44.76M	44.86M	44.86M
DER++	1.03M	11.17M	44.76M	44.76M	44.86M	44.86M
HAT	1.04M	11.23M	45.01M	45.28M	44.97M	45.11M
HyperNet	0.48M	0.47M	0.47M	0.47M	0.48M	0.48M
Sup	0.05M	1.43M	5.75M	11.45M	2.95M	5.80M
HAT+CSI	1.07M	11.25M	45.31M	45.58M	45.59M	45.72M
HAT+CSI+c	1.07M	11.25M	45.31M	45.58M	45.59M	45.72M
Sup+CSI	0.28M	1.38M	5.90M	11.60M	3.04M	6.05M
Sup+CSI+c	0.28M	1.38M	5.90M	11.60M	3.04M	6.05M

$\lambda_2 = \cdots = \lambda_{20} = 2.5$ 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 λ_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 lower penalty than later tasks. We manually give larger penalty to the first task. We did not search hyper-parameter λ_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 learning rate 0.01, 160 training iterations, and 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 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 in 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.

H Computes and Resources Used in Experiments

This paper provides a guidance on how to solve the CIL problem, backed by theoretical justifications. Based on the guidance, we have proposed some new CIL methods. Two outstanding ones are HAT+CSI and Sup+CSI. These methods achieve state-of-the-art CIL performances, but by no mean, they are the only approaches. Many CIL algorithms can be designed following the analysis as it is general to any CL model.

Despite the generality of our work, we report the execution time and required memory for HAT+CSI and Sup+CSI. The report is based on a machine with NVIDIA RTX 3090 on C10-5T experiments. HAT+CSI takes 28.68 hours while Sup+CSI runs for 18.41 hours, which are slower than baselines. Contrastive learning and extensive data augmentation in CSI are the major reason for the slow execution time. However, if other more efficient OOD detection algorithms can replace CSI, the running time can be improved with the new OOD detection methods.

As noted in Sec. 4.2, all the methods use the same backbone architecture with the same width and depth except for OWM and HyperNet for the reasons explained in the main paper. We report the number of parameters of each method required for inference after learning the last task. Sup and Sup+CSI uses a very small number of parameters because Sup finds a sparse subnetwork for each task.

Our methods HAT+CSI introduces 7.7K, 17.6K, 68.0K, 47.5K, 191.0K, and 109.0K parameters on M-5T, C10-5T, C100-10T, C100-20T, T-5T, and T-10T, respectively, at each task. Sup+CSI introduces 56.3K, 284.9K, 590.3K, 580.0K, 607.7K, and 605.1K parameters on the same experiments. The calibrated methods HAT+CSI+c and Sup+CSI+c introduce 2 parameters (α_k, β_k) per task.

For HAT and HAT+CSI, the reported number of parameters is based on the network at full capacity. The hard attention masks consume 71.10, 86.31, 98.89, 99.71, 92.94, and 98.67% of the total network capacity on average over 5 runs for HAT on M-5T, C10-5T, C100-10T, C100-20T, T-5, and T-10T, respectively. Similarly, 99.39, 99.56, 99.56, 50.68, 94.94, and 99.18% of the total network capacity are used for HAT+CSI on the same datasets on average.

I Negative Societal Impacts

The goal of continual learning is to learn a sequence of tasks incrementally. Like many machine learning algorithms, our proposed methods could be affected by bias in the input data as this work does not deal with fairness or bias in the data. A possible solution to mitigate the problem is to check bias in data before training.

J Forgetting Rate

We discuss forgetting rate (i.e., backward transfer) [28], which is defined for task t as

$$\mathcal{F}^t = \frac{1}{t-1} \sum_{k=1}^{t-1} \mathcal{A}_k^{\text{init}} - \mathcal{A}_k^t, \tag{31}$$

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.

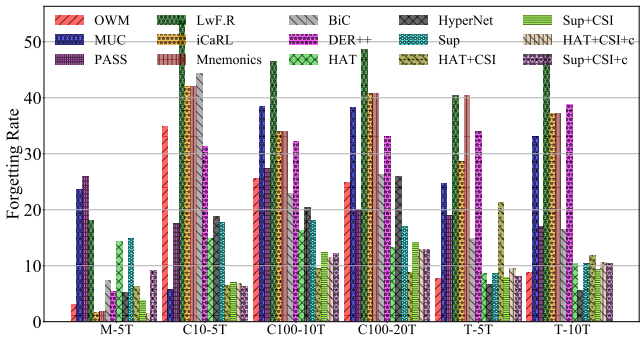


Figure 2: Average forgetting rate (%). The lower the value, the better the method is on forgetting.

Figure 2 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 independent subset of parameters for each task, and the learned parameters are not interfered during training. For the forgetting rate results in the TIL setting, refer to our earlier workshop paper [84].