A soft nearest-neighbor framework for continual semi-supervised learning

Zhiqi Kang, Enrico Fini, Moin Nabi, Elisa Ricci, Karteek Alahari

A soft nearest-neighbor framework for continual semi-supervised learning

Zhiqi Kang*[1] Enrico Fini*[2] Moin Nabi[3] Elisa Ricci[2,4] Karteek Alahari[1]

[1] Inria† [2] University of Trento [3] SAP AI Research [4] Fondazione Bruno Kessler

Abstract

Despite significant advances, the performance of state-of-the-art continual learning approaches hinges on the unrealistic scenario of fully labeled data. In this paper, we tackle this challenge and propose an approach for continual semi-supervised learning—a setting where not all the data samples are labeled. An underlying issue in this scenario is the model forgetting representations of unlabeled data and overfitting the labeled ones. We leverage the power of nearest-neighbor classifiers to non-linearly partition the feature space and learn a strong representation for the current task, as well as distill relevant information from previous tasks. We perform a thorough experimental evaluation and show that our method outperforms all the existing approaches by large margins, setting a strong state of the art on the continual semi-supervised learning paradigm. For example, on CIFAR100 we surpass several others even when using at least 30 times less supervision (0.8% vs. 25% of annotations). The code is publicly available on https://github.com/kangzhiq/NNCSL.

1. Introduction

Continual learning (CL) refers to the learning scenario where training data arrives sequentially, which results in a continuously evolving data distribution. Several efforts have been devoted to this topic in recent years, enabling impressive progress [19]. However, most of the state-of-the-art CL methods are based on a strong assumption: the data is fully labeled. This is an unrealistic scenario, as labeling data is oftentimes expensive (*e.g.*, for the expertise required or the amount of annotations), hazardous (*e.g.*, for privacy or safety concerns), or impractical (*e.g.*, in a real-time online scenario) to be acquired in real-world applications. Thus, enabling CL methods to learn with partially labeled data is of great importance. Unfortunately, this challenge has been overlooked for a long time, with a few recent attempts [8, 46, 53].

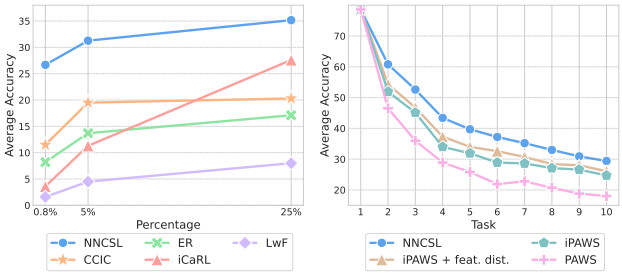


Figure 1. Left: The average accuracy with different percentages of labeled data on CIFAR-100. Our method (NNCSL) with 0.8% of the labels outperforms or matches the performance of all other methods at 25%. Right: Comparison of our approach with different versions of PAWS [4]. iPAWS is equivalent to NNCSL without our NND loss.

One way of addressing the above-mentioned problem is a setting where not all the data samples are labeled, such as *semi-supervised learning*. Many approaches exist to tackle this learning problem in the offline scenario, based on pseudo-labeling [29, 48], de-biasing [16], self-supervised learning [58] and recently non-parametric classifiers [4]. However, these methods are either ineffective or not easily extendable for the continual scenario, as shown in [8] and [53], even when paired with well-known CL methods. This stimulated the community to investigate new approaches for *continual semi-supervised learning* [8, 53], which was first formalized by [53]. This new learning scenario brings novel challenges. One issue is that the model catastrophically forgets the representations of unlabeled data while also overfitting the labeled set. This is exacerbated by another well-studied phenomenon in CL: overfitting the experience replay buffer [10]. The approaches in [8, 53] partially mitigate these issues on small-scale datasets. However, in our experiments, we find these strategies to be ineffective when the complexity of the data increases, *e.g.*, in datasets with more classes, more samples or higher resolution images (see Fig. 1 (left) and results in Sec. 6). Therefore, we argue that there is a clear need for more powerful continual semi-supervised learning methods.

*Zhiqi Kang and Enrico Fini contributed equally to this work

†Univ. Grenoble Alpes, CNRS, Grenoble INP, LJK, 38000 Grenoble, France.

Another finding of our investigation on continual semi-supervised learning is that methods that directly minimize risk on the memory buffer (i.e., using the cross-entropy loss on labeled samples) are more prone to the overfitting issue described above (see Sec. 6, Tab. 4). Instead, we found that methods like PAWS [4] that use the representations of labeled samples as a proxy to compare views of unlabeled samples are more robust on that matter, also enabling better information extraction from the unlabeled set. However, a naïve application of PAWS results in poor performance, as it is not designed for scenarios with data distribution shifts, as shown in Fig. 1 (right).

In this paper, inspired by [4], we propose to unleash the power of soft nearest-neighbor classifiers in the context of continual semi-supervised learning. In particular, we leverage their ability to non-linearly partition the feature space in two ways: i) to learn powerful representations of the current task using a self-supervised multi-view strategy, and ii) to distill previous knowledge accounting for both sample-wise and class-wise relationships. The resulting method, named NNCSL (**N**earest-**N**eighbor for **C**ontinual **S**emi-supervised **L**earning), outperforms all related methods by very large margins. For instance, as shown in Fig. 1, on CIFAR100 our method matches or surpasses all others with more than 30 times less supervision (0.8% vs. 25% of annotations), and our NND (**N**earest-**N**eighbor **D**istillation) loss mitigates forgetting better than other competitive distillation approaches.

The main contributions of this work are as follows:

- We propose NNCSL, a novel nearest-neighbor-based continual semi-supervised learning method that is, by design, impacted less by the overfitting phenomenon related to a small labeled buffer.
- We propose NND, a new distillation strategy that transfers both class-level and feature-level knowledge from the previously trained model using the outputs of a soft nearest-neighbor classifier, which effectively helps alleviate forgetting.
- We show that NNCSL outperforms the existing methods on several benchmarks by large margins, setting a new state of the art on continual semi-supervised learning. In contrast to previous approaches, our method works well on both low and high-resolution images and scales seamlessly to more complex datasets.

2. Related work

Semi-supervised learning. Semi-supervised methods focus on learning models from large-scale datasets where only a few samples have associated annotations. Since deep networks have become the mainstream in semi-supervised learning, several different approaches [18] have been introduced. Earlier strategies for this learning paradigm applied to deep architectures leveraged pseudo-labels and performed self-training based on them [29]. This scheme was later improved with confidence thresholding [2] and adaptive confidence thresholding [56, 59]. More sophisticated methods for incorporating the confidence of the predictions and filtering out spurious samples were also developed, such as FixMatch [48], which employs a student-teacher architecture. Other approaches demonstrated the benefit of co-training [37] and distillation [55].

Another class of approaches was derived with the idea of imposing similar predictions from the network for two samples obtained with different input perturbations. Notable examples of this consistency based category are [4, 5, 27, 33, 35, 43, 49, 52, 60]. For example, [4] considered a consistency loss and soft pseudo labels generated by comparing the representations of the image views to those of a set of randomly-sampled labeled images. Different strategies can be also combined and there exist methods which integrate both pseudo-labeling and consistency regularization. Recently, sample mixing techniques, such as MixUp, were also investigated in the context of semi-supervised learning for improving the model performance on low-density sample regions [6, 7, 31]. Ideas from self-supervised methods were also introduced into semi-supervised learning. For instance, self-supervised pre-training was found beneficial in [11], exponential moving average normalization was adopted in [11] and contrastive learning was considered in [3]. However, none of the aforementioned works addressed the problem of learning in an incremental setting.

Continual learning. Since learning from data in an incremental fashion is of utmost importance in many applications, several CL approaches have been proposed in the last few years. According to a recent survey [19], existing CL methods can be roughly categorized into three groups. The first category comprises regularization-based methods [13, 14, 20, 22, 24, 25, 30, 45, 54], which address the problem of catastrophic forgetting by introducing appropriate regularization terms in the objective function or identifying a set of parameters that are most relevant for certain tasks. Replay-based methods [9, 15, 32, 34, 39] correspond to the second group, and they store a few samples from previous tasks or generate them in order to rehearse knowledge during the training phases for subsequent tasks. Finally, the third category is parameter isolation methods [42, 44], which operate by allocating task-specific parameters.

While the vast majority of these methods operate in a supervised setting, recent works addressed the problem of overcoming catastrophic forgetting in the challenging case of limited or no supervision [1, 8, 21, 28, 38, 46, 47, 53]. However, most of them have default settings that are significantly different, *e.g.*, the use of external datasets, the accessibility of labeled/unlabeled data during continual learning stages, leaving only a few [8, 53] to be comparable in

our desired realistic setting. In particular, Wang *et al.* [53] addressed the continual semi-supervised learning problem and proposed ORDisCo, a method that continually learns a conditional generative adversarial network with a classifier from partially labeled data. Contrastive Continual Interpolation Consistency (CCIC) [8] is another approach, which leverages metric learning and consistency regularization for extracting knowledge from unlabeled samples. Our work radically departs from these previous methods, as we design NNCSL, a novel approach for continual semi-supervised learning based on a soft nearest-neighbor classifier. Our empirical evaluation demonstrates that NNCSL surpasses existing methods by a large margin.

3. Continual semi-supervised learning

We formally define the problem of continual semi-supervised learning in this section. Let the training data arrive sequentially, i.e., as a sequence of T tasks. The dataset associated to task t is denoted as D_t, with $t \in \{1, ..., T\}$. Learning is therefore performed task-wise, where only the current training data D_t is available during task t. When switching from one task to the next one, previous data is systematically discarded. Since the available dataset is not fully labeled, we further divide it into two subsets such that $D_t = U_t \cup L_t$. Typically in a semi-supervised learning scenario, we have $|L_t| \ll |U_t|$, the ratio $|L_t|/|U_t|$ is kept constant for all the tasks. In addition, it is common practice in the CL literature [8, 39] to allow the retention of a memory buffer M that stores and replays previously seen samples, as shown in Fig. 2.

Let f_θ be the model, parameterized by θ, and consisting of three components: a backbone g, a projector h and a classifier p. The backbone, here modeled as a convolutional neural network, is used to extract representations $\mathbf{z} = g(\mathbf{x})$ from an input image $\mathbf{x}$. The classifier takes the representation to predict a set of logits $\mathbf{p} = p(\mathbf{z})$, while the projector (implemented as a multi-layer perceptron) maps the backbone features to a lower-dimensional space $\mathbf{h} = h(\mathbf{z})$. In addition, we use superscript to refer to the state at a certain point in time, for instance for task t as f_θ^t, and for the previous task $t-1$ as f_θ^{t-1}. Similarly, we use $\mathbf{x}_u^t$ and $\mathbf{x}_l^t$ to refer to samples drawn from U_t and L_t respectively. Apart from images, the labeled dataset also contains one-hot ground truth annotations $\mathbf{y}$.

In the following sections, we introduce the proposed NNCSL method for continual semi-supervised learning. We first present PAWS [4], which inspired our NNCSL, (Sec. 4) and show why this method is not immediately applicable to the continual setting. Subsequently, we present its continual extension, iPAWS (Sec. 5.1), that solves many of its issues. However, iPAWS is still lacking a mechanism to counteract forgetting. Hence, we introduce NND, our novel distillation approach based on the soft nearest-neighbor classifier in Sec. 5.2. All these elements are harmoniously glued together in our full method: NNCSL, whose overall objective is summarised in Sec. 5.3. Fig. 3 shows an overview of NNCSL.

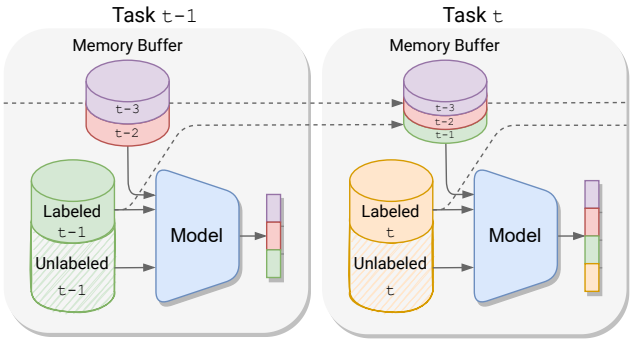


Figure 2. Illustration of the learning process in continual semi-supervised learning.

4. PAWS for CL: Strengths and weaknesses

We now introduce PAWS [4] and describe its strengths and weaknesses in scenarios with data distribution shifts. During training, the mini-batches that the model receives are composed of labeled and unlabeled data, with K and N as batch sizes for these two sets respectively. Unlabeled images in the batch are augmented twice using common data augmentation techniques to obtain two correlated views of the same sample $(\mathbf{x}, \hat{\mathbf{x}})$. The model processes the batch, producing the projected representations $\mathbf{h}_l$ and $(\mathbf{h}_u, \hat{\mathbf{h}}_u)$ for labeled and unlabeled samples respectively.

The main idea behind PAWS is to assign pseudo-labels for unlabeled samples in a non-parametric manner by considering their relationship with labeled samples. Samples are compared in the feature space using the cosine similarity of projected features, and then the pseudo-label is obtained by aggregating labels according to the similarities. More formally, let the superscript k represent the index of the k^{th} sample in the labeled mini-batch, and δ be the cosine similarity. One can apply a soft nearest-neighbor classifier to classify the augmented unlabeled sample $\hat{\mathbf{x}}_u$ as follows:

$$\hat{\mathbf{v}} = \mathrm{SNN}(\hat{\mathbf{h}}_u, \boldsymbol{S}, \epsilon) = \sum_k^K \frac{e^{\delta(\hat{\mathbf{h}}_u, \mathbf{h}_l^k)/\epsilon}}{\sum_i^K e^{\delta(\hat{\mathbf{h}}_u, \mathbf{h}_l^i)/\epsilon}} \mathbf{y}^k, \quad (1)$$

where $\boldsymbol{S} = [\mathbf{h}_l^1, ..., \mathbf{h}_l^K]$ are the features of the support samples and ϵ is a sharpening parameter that controls the entropy of the pseudo-label. Similarly, we can classify the other view of the same sample $\mathbf{v} = \mathrm{SNN}(\mathbf{h}_u, \tau)$, with the only difference that we use a more gentle sharpening parameter $\tau > \epsilon$, referred to as the temperature. Now, we can use $\hat{\mathbf{v}}$ as a target pseudo-label and train the network through

the cross-entropy loss:

$$\mathcal{L}_{\text{SNN}} = H(\mathbf{v}, \hat{\mathbf{v}}). \tag{2}$$

By nature, this loss is asymmetric, but it can be symmetrized by swapping the two views.

The mechanism described above encourages the network to output consistent representations for similar inputs, while also accounting for the distribution of the classes in the feature space. However, one issue with this formulation is that the network could output unbalanced or even degenerate predictions where some classes are predicted more frequently than others. To avoid this, PAWS imposes the distribution-wise likelihood of all the classes to be uniform using a regularization term called Mean Entropy Maximization (MEM) loss defined as:[†]

$$\mathcal{L}_{\text{MEM}} = H\left(\frac{1}{N}\sum_{n}^{N}\hat{\mathbf{v}}_n\right). \tag{3}$$

Given these two losses, the total loss for PAWS is a weighted average of the two:

$$\mathcal{L}_{\text{PAWS}} = \mathcal{L}_{SNN} + \lambda_{\text{MEM}} \cdot \mathcal{L}_{\text{MEM}}. \tag{4}$$

The advantage of this soft nearest-neighbor formulation is that it utilizes labeled samples as support vectors, not as training samples, which reduces overfitting. This property is interesting from the point of view of continual learning, since we would like to extract as much training signal as possible from the memory buffer. However, adapting PAWS for CL is not trivial, as the method is not designed to work under data distribution shifts. The key issue of PAWS in the CL setting is the assumption that the labeled and unlabeled sets exhibit the same distribution. This is untenable in CL, as the memory buffer contains classes that are not in the unlabeled set of the current task. MEM loss aggravates this problem, as it tries to scatter the pseudo label over all the classes, even for the ones whose unlabeled samples are unavailable. A simple solution would be to use the labeled data of the current task and discard the buffer, but this is sub-optimal as the buffer is critical for CL.

Another important drawback of PAWS in the context of CL is that it performs best when fine-tuned with supervision only (using a linear classifier), similar to many methods based on self-supervised objectives. This also alleviates the computational issues of nearest-neighbor at inference time. However, this two-stage pipeline (pre-training and then fine-tuning) poses a problem for CL. Should we retain the fine-tuned model or the one obtained after pre-training? How do we compare with other methods that do not need fine-tuning? In the following section, we describe our solution to overcome these limitations.

[†]With a slight abuse of notation we refer to $H(\cdot)$ with one argument as the entropy function, while when two arguments are passed we consider it as the cross-entropy function $H(\cdot,\cdot)$.

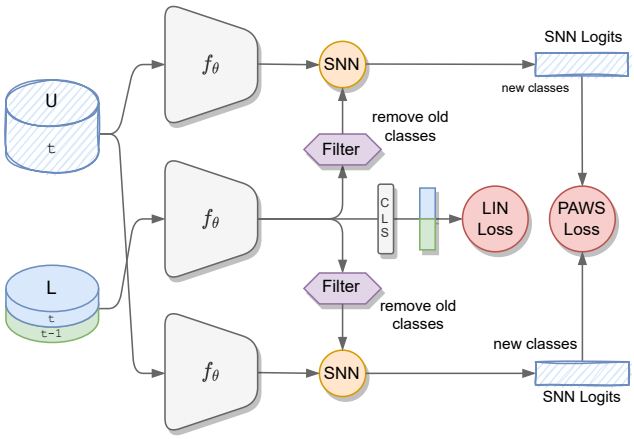


Figure 3. Overview of the base learner component of our method, which does not have a distillation loss. We refer to this as iPAWS.

5. NNCSL: Our nearest-neighbor approach for continual semi-supervised learning

Our NNCSL leverages the observations made in Sec. 4 about PAWS. It is composed of a base learner (iPAWS, Sec. 5.1) and a distillation loss (NND, Sec. 5.2).

5.1. Continually predicting view assignments

We first describe our proposed approach for leveraging PAWS' strengths while overcoming its weaknesses. As mentioned above, the easiest way to make the labeled and unlabeled distributions match is to disregard the memory buffer. This is obviously undesirable for CL. However, one could multi-task PAWS with another objective that also takes into account the information of the memory buffer. In particular, we suggest processing labeled samples of all the classes seen so far, but filtering out samples from the previous tasks so they do not interfere in the computation of Eq. 1. However, we can use the output of the linear classifier p and optimize a standard cross-entropy loss:

$$\mathcal{L}_{\text{LIN}} = \sum_{j}^{J} \text{H}(\mathbf{p}^k, \mathbf{y}^j), \tag{5}$$

on all the J labeled samples in the current batch (which also contains K labeled samples of the current task). The complete loss for our base continual semi-supervised learner (named iPAWS) is as follows:

$$\mathcal{L}_{\text{iPAWS}} = \mathcal{L}_{\text{PAWS}} + \lambda_{\text{LIN}} \cdot \mathcal{L}_{\text{LIN}}. \tag{6}$$

This loss has several favorable effects, it: i) stimulates the network to focus on the old classes while learning representations of the new ones through PAWS, ii) creates an ensemble effect between the two classifiers, iii) completely removes the need for fine-tuning, as the linear classifier is

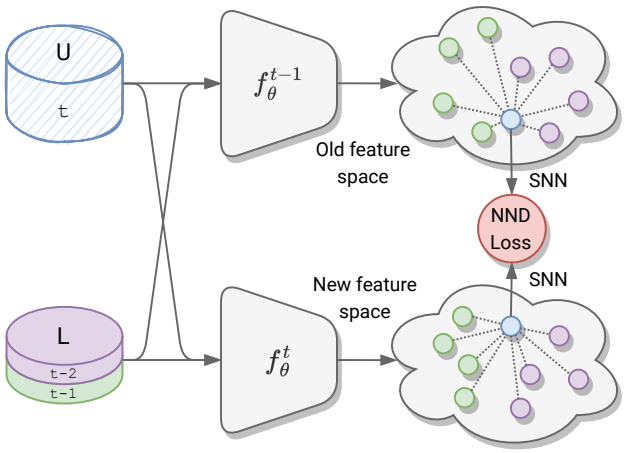


Figure 4. Illustration of our soft nearest-neighbor distillation loss.

trained online, and iv) enables us to control the trade-off between fitting labeled or unlabeled data through the parameter λ_{LIN}. Interestingly, we found that very small values of λ_{LIN} work well in practice, while larger values increase overfitting. We believe that, due to its partially self-supervised nature, $\mathcal{L}_{\text{SNN}}$ learns improved representations, that can be easily discriminated by the linear classifier. An illustration of the architecture of iPAWS is shown in Fig. 3.

5.2. Soft nearest-neighbor distillation

Distilling information [23] is a common practice in CL, which utilizes frozen models trained on previous tasks as a teacher to regularize the currently active model, which is a student. Let t be the index for the current task. The student model f_θ^t aims to mimic the outputs of the teacher f_θ^{t-1}, while learning the new task. Previous works [22, 30] typically distill either the logits of a linear classifier or the features of the hidden layers of the network. However, in iPAWS, the main driver for the network to learn representations is the loss applied to the soft nearest-neighbor classifier. As seen earlier, this loss does not give any signal on previous data, as old samples get filtered out and fed to the linear classifier only. This is made worse by the fact that the nearest-neighbor classifier is applied on a separated projection head h, that has no incentive to remember previous knowledge.

To mitigate these issues, we devise a novel Nearest-Neighbor Distillation (NND) loss that blends nicely with our framework. The loss is based on the intuition that we can assess the nearest-neighbor classifier on the old feature space using the same support samples. This equates to computing the following two vectors:

$$\mathbf{w} = \text{SNN}(\mathbf{h}_u, \boldsymbol{R}, \tau), \tag{7}$$

$$\mathbf{w}^{t-1} = \text{SNN}(\mathbf{h}_u^{t-1}, \boldsymbol{R}^{t-1}, \tau), \tag{8}$$

where $\mathbf{h}_u^{t-1}$ is a feature vector output by the teacher for an unlabeled sample $\mathbf{x}_u$, while $\boldsymbol{R}$ and $\boldsymbol{R}^{t-1}$ represent the support set of previous classes embedded in the old and new feature spaces respectively. To mitigate forgetting, we use the probabilities predicted by the teacher as a distillation target:

$$\mathcal{L}_{\text{NND}} = H(\mathbf{w}, \mathbf{w}^{t-1}). \tag{9}$$

Note that the output of the teacher is not sharpened as it is done in Eq. 1. We apply the same temperature for both new and old features. We emphasize that here we use an inverted filter as that of Sec. 5.1, to distill knowledge about the previous classes only.

Our NND loss may appear similar to standard distillation, but the rationale behind it is not the same. Knowledge distillation focuses on class-level distributions, so it loses information about representations of single features in the latent space. On the contrary, NND distills the aggregated relationships of each unlabeled sample with respect to all the labeled ones in the mini-batch. This encourages the model to maintain more stable representations (Fig. 4). NND is able to capture and transfer non-linear relationships between samples and classes, while knowledge distillation is a limited comparison of linear boundaries. Also, the SNN classifier is computed on a different support set sampled from the buffer at every iteration, which introduces a type of variation that further regularizes the model. With respect to feature distillation, NND carries more information about class distributions, which results in improved performance.

5.3. Overall loss

The overall NNCLS model, composed of two novel components, i.e., iPAWS and NND, is trained with the loss:

$$\mathcal{L}_{\text{NNCSL}} = \mathcal{L}_{\text{iPAWS}} + \lambda_{\text{NND}} \cdot \mathcal{L}_{\text{NND}}. \tag{10}$$

6. Evaluation and analysis

6.1. Experimental settings

Datasets. We evaluate our method on three datasets. CIFAR-10 [26] is a dataset of 10 classes with 50k training and 10k testing images. Each image is of size 32×32. CIFAR-100 [26] is similar to CIFAR-10, except it has 100 classes containing 500 training images and 100 testing images per class. ImageNet-100 [50] is a 100-class subset of the ImageNet-1k dataset from ImageNet Large Scale Visual Recognition Challenge 2012, containing 1300 training images and 50 test images per class.

Continual semi-supervised setting. For both CIFAR-10 and CIFAR-100, we mainly train the models with three different levels of supervision, i.e., $\lambda \in \{0.8\%, 5\%, 25\%\}$. For instance, this corresponds to 6, 25, 125 labeled samples per class in CIFAR-100. As for ImageNet-100, we opt for 1% labeled data. To build the continual datasets,

Method	CIFAR-10			CIFAR-100		
	0.8%	5%	25%	0.8%	5%	25%
Fine-tuning	13.6±2.9	18.2±0.4	19.2±2.2	1.8±0.2	5.0±0.3	7.8±0.1
LwF [30]	13.1±2.9	17.7±3.2	19.4±1.7	1.6±0.1	4.5±0.1	8.0±0.1
oEWC [25]	13.7±1.2	17.6±1.2	19.1±0.8	1.4±0.1	4.7±0.1	7.8±0.4
ER [41]	36.3±1.1	51.9±4.5	60.9±5.7	8.2±0.1	13.7±0.6	17.1±0.7
iCaRL [39]	24.7±2.3	35.8±3.2	51.4±8.4	3.6±0.1	11.3±0.3	27.6±0.4
GDumb [36]	39.6±9.6	40.9±11.8	44.8±5.4	8.6±0.1	9.9±0.4	10.1±0.4
CCIC [8] (500)	54.0±0.2	63.3±1.9	63.9±2.6	11.5±0.7	19.5±0.2	20.3±0.3
PAWS [4] (500)	51.8±1.6	64.6±0.6	65.9±0.3	16.1±0.4	21.2±0.4	19.2± 0.4
NNCSL (500)	**73.2**±0.1	**77.2**±0.2	**77.3**±0.1	**27.4**±0.5	**31.4**±0.4	**35.3**±0.3
CCIC [8] (5120)	55.2±1.4	74.3±1.7	**84.7**±0.9	12.0±0.3	29.5±0.4	44.3±0.1
ORDisCo [53] (12500)	41.7±1.2	59.9 ±1.4	67.6±1.8	-	-	-
NNCSL (5120)	**73.7**±0.4	**79.3**±0.3	81.0±0.2	**27.5**±0.7	**46.0**±0.2	**56.4**±0.5

Table 1. Average accuracy with standard deviation of different methods tested with 5-task CIFAR-10 and 10-task CIFAR-100 settings. The number between brackets indicates the size of the memory buffer for the labeled data.

we mainly use the standard setting in the literature [8, 53], and divide the datasets into equally disjoint tasks: 5/10/20 tasks for CIFAR-10/CIFAR-100/ImageNet-100, i.e., 2/10/5 classes per task, respectively. We follow the standard class-incremental learning setting in all our experiments. During the CL stages, we assume that all the data of previous tasks are discarded. A memory buffer can be eventually built, but only for labeled data. Following [8], we set the buffer size for labeled data as 500 or 5120, to ensure a fair comparison.

Metrics. We evaluate the performance of different methods considering the average accuracy over all the seen classes after each task, as is common in CL methods [19]. We also consider average incremental accuracy [12] for our analysis, which is the average of the average accuracy across all the tasks. The latter measure gives a global view of the entire continual learning scheme.

Implementation details. As in [8], we use ResNet18 as our backbone for all the datasets. We adopt most of the hyper-parameters and learning components from [4], unless explicitly stated. Specifically, we use the LARS optimizer [57] with a momentum value of 0.9. As the scale of the dataset is relatively small in continual semi-supervised learning, we increase the weight decay to 10^{-5}, reduce the batch-size to 256. The learning rate is reduced to 0.4 for CIFAR-10 and 1.2 for CIFAR-100 and ImageNet-100, respectively. We apply 10 epochs warm-up and reduce it with a cosine scheduler as well. For the two correlated views of the same sample, we generate two large crops and two small crops of each sample. The large crops serve as targets for each other, whereas they are both targets for the small crops. We apply data augmentation as in [17]. As for the labeled data, label smoothing is applied with a smoothing factor of 0.1. The additional linear evaluation head is a simple linear layer, which we use to predict labels during test time. We choose $\lambda_{\text{NND}} = 0.2$ and $\lambda_{\text{LIN}} = 0.005$. As for the memory buffer, we utilize a simple random sampling strategy. We run our experiments 3 times each with different random seeds. The standard deviation is also reported, if applicable. Further details and analyses are presented in the supplementary material.

Baselines. As baselines, we first consider traditional fully-supervised CL methods. They are not designed to deal with a semi-supervised setting. A straightforward way to convert them into a continual semi-supervised setting is to use only the labeled data available during training and to discard the unlabeled data. We consider two categories of methods: regularization-based methods, i.e., Learning without Forgetting (LwF) [30], online Elastic Weight Consolidation (oEWC) [25], and replay-based methods, i.e., Experience Replay (ER) [41], iCaRL [39] and GDumb [36]. We also consider continual semi-supervised learning baseline methods, such as CCIC [8] and ORDisCo [53]. While CCIC has an explicit definition of the memory buffer for labeled data, which is either 500 or 5120, ORDisCo directly stores all the labeled data that the model receives. We thus consider the upper bound as its memory buffer size, which is $M = 12500$, equivalent to 25% of CIFAR-10. We also consider an upper bound, which is commonly shown in CL [19], obtained by jointly training the model with all available data, and the lower bound, where we fine-tune the model on each new task.

6.2. Results

CIFAR-10 and CIFAR-100. We first report the performance of different methods on CIFAR-10 and CIFAR-100 in Tab. 1. The upper bound performance of joint training

Metric	CCIC	NNCSL
Average Accuracy	2.9	**29.3**
Average Incremental Accuracy	10.0	**37.7**

Table 2. Final Average Accuracy and Average Incremental Accuracy of CCIC and NNCSL in 20-task ImageNet-100.

on CIFAR-10 is of 92.1±0.1%, and that of CIFAR-100 is of 67.7±0.9%. NNCSL outperforms all the competitors in all settings but one, with a significant margin. For instance, when using a buffer of 5120 and 0.8% of labeled data, NNCSL performs better than or substantially matches almost all the others, even when they use 25% labeled data, i.e., about 30 times more supervision. It is also interesting to note that NNCSL has a very low variance (≤ 0.7) across all the settings, indicating a better convergence and representation learning during training.

From the results in Tab. 1, it is also clear that CL baselines originally developed for the supervised setting cannot handle well a situation of limited supervision, as in continual semi-supervised learning. On the other hand, methods such as CCIC and ORDisCo[†] benefit from their design specific to the continual semi-supervised learning scenario. While ORDisCo has a larger memory buffer, its performance is inferior to that of CCIC. We believe that this is due to the difficulty in jointly training a continual classifier with a GAN model. CCIC performs reasonably well on CIFAR-10, especially with a large memory buffer. When the buffer size is 5120, CCIC slightly outperforms NNCSL in the 25% setting. We suspect that our method underfits in this setting, due to the very small weight of the linear evaluation loss. In contrast, CCIC relies more on labeled data, with equal or even higher importance for the supervised loss. To validate our hypothesis, we increased the weight λ_{LIN} for our linear classifier loss. We obtained an accuracy of 84.5±0.4%, which is comparable with CCIC in the same setting (84.7±0.9%). It is also worth noting that the 5-task CL setting in CIFAR-10 is a relatively easy problem, wherein we can almost safely discard the unlabeled data if 25% of data is labeled and the memory buffer is more than 10% of the size of the dataset. However, in the case of a large-scale dataset such as CIFAR-100, the superiority of NNCSL is clearly evident.

ImageNet-100. We also evaluate NNCSL on the more challenging dataset of ImageNet-100 and compare it with the best-performing method, CCIC. We consider a 20-task continual semi-supervised setting with only 1% of labeled data. As shown in Fig. 5, our method outperforms CCIC with a large margin of over 25% (29.3% vs. 2.9%). It is also interesting to note that the average accuracy of our proposed method stabilizes at around 30% after the 10^{th} task, showing that NNCSL can effectively retain knowledge acquired during the continual learning steps. We also report the average incremental accuracy for both the approaches in Tab. 2, confirming that NNCSL is superior to CCIC. It is worth seeing in Fig. 5 that the curve of CCIC is monotonously decreasing, whereas NNCSL has a clear rebound between tasks 11 and 16, showing that the model is learning quicker than forgetting.

[†]Note that the results of ORDisCo are directly provided by the authors of [53] as there is no open-source implementation of their approach.

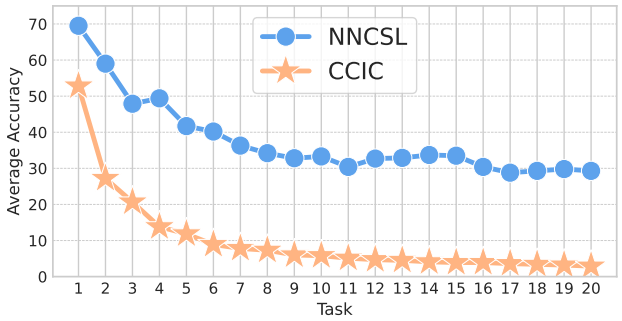


Figure 5. Comparison with CCIC [8] on 20 tasks setting, ImagetNet-100 dataset and 1% of labeled data. The average accuracy after each learning step is shown.

Method	Component			Dataset	
	Distill	Filter	Linear	C10	C100
PAWS				51.8	16.1
PAWS			✓	53.0	17.2
iPAWS		✓	✓	63.8	23.4
NNCSL	✓	✓	✓	**73.2**	**27.4**

Table 3. Ablation study of the effectiveness of the proposed components on 5-task CIFAR-10 and 10-task CIFAR-100, with memory buffer size $M = 500$. We use average accuracy as metrics.

6.3. Additional analysis

Ablation. We ablate the components of our framework in Tab. 3 on both CIFAR-10 (C10 in the table) and CIFAR-100 (C100). The largest improvement comes from the filtering (10.8% and 6.2% improvements on CIFAR-10 and CIFAR-100 respectively) and the distillation (9.4% and 4% improvements on CIFAR-10 and CIFAR-100 respectively). This confirms the contributions of our proposed components. Although the linear evaluation loss does not bring a significant improvement to the overall performance, we find it useful for stabilizing the learning process, especially for small datasets with very limited supervision. We justify the necessity of this term in the following analysis.

Evidence of effectiveness. We visualize in Fig. 6 the learning curve of average accuracy for unlabeled training data. As we illustrate in Sec. 5.1, the vanilla MEM loss

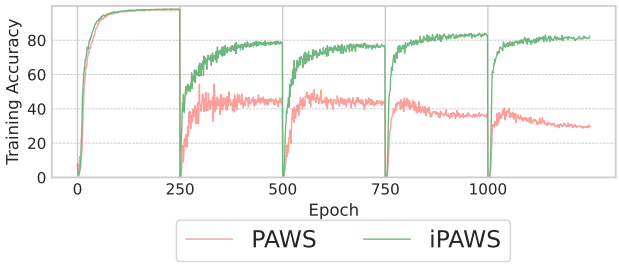


Figure 6. Average training accuracy of unlabeled data on 5-task CIFAR-10. Comparison between vanilla PAWS and our proposed iPAWS in the continual semi-supervised setting.

λ_{LIN}	1	0.05	0.005	0.001
Train (labeled)	99.9	99.9	97.9	96.5
Train (unlabeled)	74.7	76.0	**76.3**	75.2
Validation	77.1	78.4	**78.9**	77.3

Table 4. Training and validation accuracies on the current task (i.e., evaluate on task t while training on task t) with different values of λ_{LIN}. Larger weights increase overfitting on the labeled set and reduce generalization. These experiments are conducted on CIFAR-10 with 5 tasks.

Method & Distillation	**T1**	**T2**	**T3**	**T4**	**T5**
iPAWS	25.2	22.9	24.6	37.3	37.0
iPAWS + Logit dist.	27.8	24.3	22.9	35.4	31.7
iPAWS + Feature dits.	26.5	25.8	26.3	43.9	37.1
NNCSL	**32.2**	**26.3**	**28.1**	**46.8**	**38.5**

Table 5. Final accuracy on each task after training on CIFAR-100 with 5 tasks. We use iPAWS as a baseline to which we add different distillations. NNCSL is equivalent to iPAWS + NND.

is detrimental to the learning as it forces the model to assign incorrect pseudo-labels to the unlabeled data. Our proposed iPAWS effectively resolves this issue and allows the model to learn a better representation with the unlabeled data. Note that we only use the labels of unlabeled data to monitor the training, and no label information is leaked at train time.

Impact of distillation. Complementary to Fig. 1 (right), which qualitatively shows the superior performance of our proposed NND, we report in Tab. 5 the final accuracy of each task after training on all tasks. An effective distillation method should be able to maintain the performance of old tasks (*e.g.*, task 1 for NND vs. Feature distillation) and efficiently learn new tasks (*e.g.*, task 5 for NNS vs. Logit distillation).

Method	**Replay strategy**	**Average Accuracy**
NNCSL	Labeled	**76.7**
NNCSL	Labeled & Unlabeled	**82.1**
ORDisCo	Labeled & GR	65.9

Table 6. Comparison of different strategies for the replay buffer with 5-task CIFAR-10, using 3% of labeled data to match the setting of [53].

Linear evaluation head. We report the train and validation accuracy in Tab. 4. Note that for the accuracy at train time, we use the average of accuracy on the current task, i.e., evaluation on task t while training on task t. In general, having a high training accuracy of unlabeled data means the model learns a good representation, which leads to a good validation accuracy. One can observe overfitting when λ_{LIN} grows. This justifies our choice of i) having a small weight for the linear evaluation head, and ii) choosing PAWS as the basis, which does not directly use labeled data as training samples to avoid overfitting. Moreover, we observe underfitting when λ_{LIN} is smaller than 0.005. It confirms the need for this linear classifier in the continual semi-supervised scenario. We thus set λ_{LIN} as 0.005 in our experiments.

Replay strategies. ORDisCo utilizes a generative replay (GR) strategy to replay unlabeled data. Given that the generative model brings a memory overhead that is not negligible, it is reasonable to equip our method with a memory buffer for unlabeled data for a fair comparison. Specifically, we use 5000 samples, which is equivalent to the size of the generative model of ORDisCo. Tab. 6 shows that having access to the previously seen unlabeled data can indeed improve the performance of our method, and our NNCSL performs better with a simple memory buffer than ORDisCo with a sophisticated generative-replay strategy. This experiment confirms the ability of our method to exploit unlabeled data.

7. Conclusion

In this work, we studied continual semi-supervised learning and proposed NNCSL, a novel approach based on soft nearest-neighbors and distillation. Our extensive experiments show the superior performance of NNCSL with respect to existing methods, setting a new state of the art. Previous work [4] showed that using a more powerful network such as wider or deeper ResNet can further improve performance. While this has not been addressed in this work, we consider it an interesting direction for future work. In this paper, we considered a fixed ratio between labeled and unlabeled samples across all tasks. A varying ratio would be an even more challenging setting for future investigation.

Acknowledgements. This work was funded in part by the ANR grant AVENUE (ANR-18-CE23-0011). It was also granted access to the HPC resources of IDRIS under the allocation 2021-[AD011013084] made by GENCI.

Appendix

A. Implementation details

Although our model shares most of the hyper-parameters across different datasets, there are few differences, in values chosen empirically, to adapt to different scenarios. NNCSL, as well as PAWS and iPAWS for ablation study, are trained with 250 epochs per task for CIFAR-10 and CIFAR-100, and 100 epochs for ImageNet-100. For CIFAR-10, the learning rate is initialized as 0.08, warmed-up to 0.4, and reduced to 0.032 with the cosine scheduler. For CIFAR-100, a similar variation of learning rate is set from 0.08 to 1.2 to 0.032, and for ImageNet-100, it is 0.3 to 1.2 to 0.064. The color distortion ratio is set to 1 for ImageNet-100 and 0.5 for CIFAR-10 and CIFAR-100. The size of the mini-batch for labeled data is set to 5 for CIFAR-10 and 3 for CIFAR-100 and ImageNet-100. The size of the mini-batch for unlabeled data is set to 256 for CIFAR-10 and CIFAR-100 and 64 for ImageNet-100. These hyper-parameters are mostly based on the suggested default values of PAWS [4], and we empirically update them after testing with a moderate set of values variant around the default ones, based on the validation performance. However, We do not perform hyper-parameter tuning on ImageNet-100: we first adopt the hyper-parameters for ImageNet from PAWS, and update them with the same changes we apply on CIFAR-100.

For the continual learning setting, we initialize a unified linear evaluation head where the number of outputs is the total number of classes in the dataset. When a class is not yet seen by the model, the corresponding output is masked. To retain a copy of the previously trained model, we use the *deepcopy* method from the *copy* package[†] The copied model is in evaluation mode when training the current model on the new classes.

We have included our source code as part of the supplementary material. All the implementation details can be found in the options files, for instance, random seeds, labeled samples on each dataset. We plan to open-source our code upon acceptance of this submission.

B. Comparison of data augmentation

We note that the data augmentation of our proposed framework is not the same as the one used in CCIC [8]. CCIC utilizes random cropping and horizontal flipping (which we refer to as *weak DA*), whereas our proposed iPAWS and NNCSL include color distortion as an additional operation for data augmentation (referred to as *strong DA*). To verify the impact of this additional augmentation strategy, we include color distortion in the data augmentation process of CCIC and re-train it from scratch on CIFAR-10 with 5 tasks, 5% labeled data, and buffer size 5120, and also on ImageNet-100 with 20 tasks, 1% labeled data and buffer size 5120. The results are reported in Tab. 7. CCIC does not benefit from color distortion either on small- or large-scale datasets. We believe this is because CCIC does not have the multiple-view consistency to be robust with respect to strong data augmentation. Consequently, we chose to report results using CCIC's original (and more effective) data augmentation.

[†]https://docs.python.org/3/library/copy.html

Method	Dataset	Data Augmentation	
		Weak	Strong
CCIC	C10	**72.8**	69.4
CCIC	IN100	**3.1**	2.3

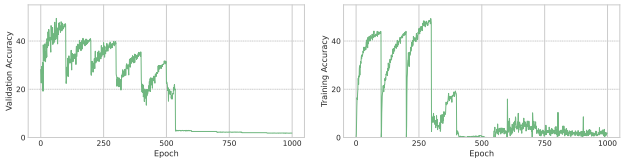
Table 7. Comparison of different data augmentation strategies for CCIC on CIFAR-10 (denoted as C10 in the table) and ImageNet-100 (denoted as IN100 in the table).

Method	Component			Dataset
	Distill	Filter	Linear	IN100
PAWS				Collapse
PAWS			✓	Collapse
iPAWS		✓	✓	27.1
NNCSL	✓	✓	✓	**29.3**

Table 8. Ablation study of the effectiveness of the proposed components on 20-task ImageNet-100, with memory buffer size $M = 5120$ and labeled ratio 1%. We use average accuracy as metrics.

C. Ablation study of NNCSL on ImageNet-100

We performed an ablation study of our proposed NNCSL on ImageNet-100 (IN100) to verify the effectiveness of our proposed components on a large dataset with a challenging 20-task scenario. We report the results in Tab. 8. The difference between PAWS and NNCSL again confirms that our proposed components are very effective. Note that we did not do any hyper-parameter tuning on ImageNet-100, instead, we re-use the set of hyper-parameters from PAWS's default setting and CIFAR-100 experiments. This explains why the gap between NNCSL and iPAWS is reduced. We believe a higher weight for the NND of NNCSL may improve the performance in this setting, as the learning is more difficult (e.g., high-resolution images) and the forgetting is more severe (e.g., more tasks than in CIFAR-100).



(a) Validation accuracy (b) Training accuracy (unlabeled)

Figure 7. Learning curve of PAWS on ImageNet-100.

It is interesting to see that PAWS diverges in this setting. Our analysis reveals that the vanilla MEM loss strongly impacts representation learning on unlabeled data and makes the learning procedure highly unstable, as shown in Fig. 7. Although we do not observe the same collapse of PAWS on CIFAR-10 or CIFAR-100, recall that in Fig. 6 of the main paper, the training accuracy of unlabeled data for PAWS is strongly constrained on CIFAR-10. This means the representation learning of PAWS is already vulnerable. As images of ImageNet-100 have a much larger resolution than that of CIFAR-10, learning a robust feature from the input of ImageNet-100 is significantly more difficult. In such a complex case, the model easily diverges but can hardly recover, we suspect that it is because the gradient is very noisy (due to the negative impact of MEM loss) and small (due to the partial supervision and indirect use of labeled data). To verify this assumption, we observe that adding the linear head slightly alleviates the divergence. However, it cannot prevent the collapse from happening. This means that the MEM loss is the main cause of the collapse and is indeed detrimental to representation learning.

Nevertheless, we believe it may be possible to resolve this collapse without changing the framework, i.e., PAWS. For example, one can conduct careful, extensive hyperparameter tuning to find an optimal set of parameters which can stabilize the learning. However, it is not realistic given the scale of the dataset. Hence, we did not conduct such experiments.

D. Impact of λ_{NND}

In Tab. 9, we report the performance of our NNCSL with respect to different values of λ_{LIN} on CIFAR-100, with 5 tasks, 1% of labeled data and buffer size 5120. λ_{NND} controls the importance of the distillation branch with respect to the PAWS loss. The higher the value, the stronger constraint the model receives to retain the previous knowledge. $\lambda_{\text{NND}} = 0$ means no distillation, which reduces the model back to iPAWS. We can clearly see that distillation helps the model perform better (e.g., $\lambda_{\text{NND}} = 0$ vs. $\lambda_{\text{NND}} = 0.2$) and too much regularization from distillation can constraint the model from learning new knowledge (e.g., $\lambda_{\text{NND}} = 0.2$ vs. $\lambda_{\text{NND}} = 1$).

λ_{LIN}	0	0.01	0.1	0.2	1
NNCSL	29.0	30.2	31.8	**33.6**	30.5

Table 9. Average Accuracy with different values of λ_{NND}. These experiments are conducted on CIFAR-100 with 5 tasks, 1% of labeled data and buffer size 5120.

Method	**FWT** ↑	**BWT** ↑
PAWS	1.1	-13.7
iPAWS	26.8	-18.25
NNCSL	**31.7**	**-17.15**

Table 10. Forward transfer (FWT) and backward transfer (BWT) of PAWS, iPAWS and NNCSL in 20-task ImageNet-100.

E. Forward and backward transfer analysis

Forward transfer (FWT) and backward transfer (BWT) are commonly used in continual learning literature [32, 40]. The former measures the capacity of the model to generalize to future tasks, whereas the latter shows the capacity of the model to retain the previously acquired knowledge. Specifically, they are defined as follows. Let T again be the total number of tasks for the continual learning stages, we therefore can divide the test set into T segments, each one representing one task. After each task t, the model is evaluated with respect to all T test sets. Consequently, we obtain a matrix $R \in \mathbb{R}^{T\times T}$, where the element $R_{i,j}$ is the test performance on task j with the model on task i. We use *classification accuracy* as our evaluation metrics. In addition, we define the random estimation as r_j, which represents the test performance on task j using a model with only random initialization. We can define the FWT and BWT as:

$$FWT = \frac{1}{T-1}\left(\sum_{i=2}^{T} R_{i-1,i} - r_i\right). \tag{11}$$

$$BWT = \frac{1}{T-1}\left(\sum_{i=1}^{T-1} R_{T,i} - R_{i,i}\right). \tag{12}$$

Similarly, we can define the average accuracy (ACC) as:

$$ACC = \frac{1}{T}\left(\sum_{i=1}^{T} R_{T,i}\right). \tag{13}$$

It should be noticed that computing the backward transfer for the first task or the forward transfer for the last task have little utility and are excluded from Eq. 11 and Eq. 12.

We report the results in Tab. 10 a comparison of our proposed components. Note that PAWS diverges in this setting, leading to a low FWT. Instead, PAWS is better than iPAWS and NNCSL if we look at BWT alone. It is simply

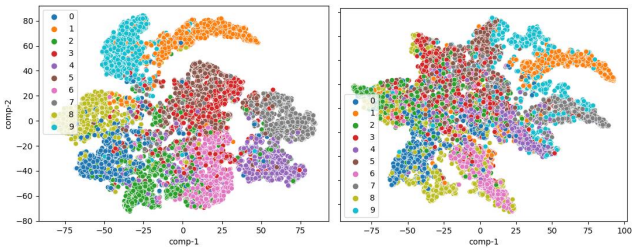


Figure 8. T-SNE visualization of deep features of 10 classes of CIFAR-10, these experiments are conducted with 5 tasks. Left: features from NNCSL after training on task 5, Right: features from PAWS after training on task 5. Data points are colored by their corresponding classes. A clear class boundary after several tasks shows a robust representation along the continual learning stages.

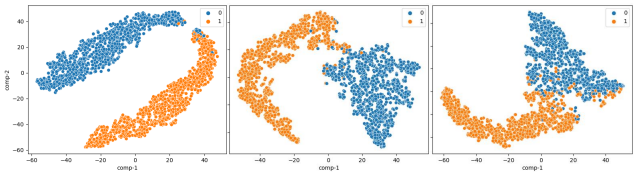


Figure 9. T-SNE visualization of deep features of the first 2 classes of CIFAR-10, these experiments are conducted with 5 tasks. Left: features from NNCSL after training on task 1, Middle: features from NNCSL after training on task 5, Right: features from PAWS after training on task 5. Data points are colored by their corresponding classes. It is clear that PAWS suffers from a blurry class boundary after several continual learning stages.

because $R_{T,i}$ and $R_{i,i}$ are all low after the divergence, having not much room for the model to forget. That is, a model cannot forget if it does not learn anything first. This observation confirms the limitation of BWT, as it only shows a relative difference with respect to its own performance, i.e., Eq. 12. Thus, BWT is more suitable to be an additional indicator when the average accuracy of the two methods are close to each other, e.g., NNCSL vs. iPAWS. Comparing NNCSL and iPAWS, we notice that the NND helps slightly improve the BWT. What is more interesting is that NND significantly improves FWT. We believe it is because NND stabilizes the representation learning, allowing the model to generalize better to future tasks.

We also notice that the absolute value of BWT is high for both NNCSL and iPAWS. We suggest that it is because the first task suffers the most from forgetting, as it is trained with a simple task and without any regularization of distillation, but it goes through all continual stages. To verify this assumption, we compute the BWT without the first task: -11.3 for NNCSL and -9.23 for iPAWS, which are significantly improved from the BWT scores in Tab. 10.

F. Visualization of the deep features

We use t-SNE [51] to project the deep features into a lower-dimensional space and visualize them to qualitatively verify the effectiveness of our proposed method. We apply t-SNE on the deep features $\mathbf{h}_u = h(\mathbf{z}_u)$ of **unlabeled data** and color them in the visualization with their ground-truth label. Ideally, if the features are well learned, one can see different clusters representing different classes in the visualization. Specifically, we choose 5-task CIFAR-10 to ensure a distinguishable class boundary.

The result is shown in Fig. 8. The figure on the left shows the features of all 10 classes after task 5, using NNCSL. Recall that CIFAR-10 is divided into 5 tasks. We can see a clear separation of different classes in the visualization. Fig. 8 Right shows the features of the same 10 classes after task 5 using PAWS. We can clearly see that the vanilla MEM loss of PAWS causes a blurry class boundary as it tried to scatter over all classes with partially available unlabeled data.

To have a more detailed view of the feature space, we select the first two classes as examples and visualize them at different training stages using different methods. Fig. 9 confirms that PAWS leads to a blurry boundary and is prone to severe forgetting due to this effect.

References

[1] Alessandro Achille, Tom Eccles, Loic Matthey, Christopher P Burgess, Nick Watters, Alexander Lerchner, and Irina Higgins. Life-long disentangled representation learning with cross-domain latent homologies. *NeurIPS*, 2018. 2

[2] Eric Arazo, Diego Ortego, Paul Albert, Noel E O'Connor, and Kevin McGuinness. Pseudo-labeling and confirmation bias in deep semi-supervised learning. In *International Joint Conference on Neural Networks (IJCNN)*, pages 1–8, 2020. 2

[3] Mahmoud Assran, Nicolas Ballas, Lluis Castrejon, and Michael Rabbat. Supervision accelerates pre-training in contrastive semi-supervised learning of visual representations. *arXiv preprint arXiv:2006.10803*, 2020. 2

[4] Mahmoud Assran, Mathilde Caron, Ishan Misra, Piotr Bojanowski, Armand Joulin, Nicolas Ballas, and Michael Rabbat. Semi-supervised learning of visual features by non-parametrically predicting view assignments with support samples. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 8443–8452, 2021. 1, 2, 3, 6, 8, 9

[5] Ben Athiwaratkun, Marc Finzi, Pavel Izmailov, and Andrew Gordon Wilson. There are many consistent explanations of unlabeled data: Why you should average. In *International Conference on Learning Representations*, 2019. 2

[6] David Berthelot, Nicholas Carlini, Ekin D Cubuk, Alex Kurakin, Kihyuk Sohn, Han Zhang, and Colin Raffel. Remixmatch: Semi-supervised learning with distribution alignment and augmentation anchoring. In *International Conference on Learning Representations*, 2020. 2

[7] David Berthelot, Nicholas Carlini, Ian Goodfellow, Nicolas Papernot, Avital Oliver, and Colin A Raffel. Mixmatch: A holistic approach to semi-supervised learning. *Advances in neural information processing systems*, 32, 2019. 2

[8] Matteo Boschini, Pietro Buzzega, Lorenzo Bonicelli, Angelo Porrello, and Simone Calderara. Continual semi-supervised learning through contrastive interpolation consistency. *Pattern Recognition Letters*, 162:9–14, 2022. 1, 2, 3, 6, 7, 9

[9] Pietro Buzzega, Matteo Boschini, Angelo Porrello, Davide Abati, and Simone Calderara. Dark experience for general continual learning: a strong, simple baseline. In *NeurIPS*, 2020. 2

[10] Pietro Buzzega, Matteo Boschini, Angelo Porrello, and Simone Calderara. Rethinking experience replay: a bag of tricks for continual learning. In *2020 25th International Conference on Pattern Recognition (ICPR)*, pages 2180–2187, 2021. 1

[11] Zhaowei Cai, Avinash Ravichandran, Subhransu Maji, Charless Fowlkes, Zhuowen Tu, and Stefano Soatto. Exponential moving average normalization for self-supervised and semi-supervised learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 194–203, 2021. 2

[12] Francisco M Castro, Manuel J Marin-Jimenez, Nicolas Guil, Cordelia Schmid, and Karteek Alahari. End-to-end incremental learning. In *ECCV*, 2018. 6

[13] Hyuntak Cha, Jaeho Lee, and Jinwoo Shin. Co2l: Contrastive continual learning. In *CVPR*, pages 9516–9525, 2021. 2

[14] Arslan Chaudhry, Puneet K Dokania, Thalaiyasingam Ajanthan, and Philip HS Torr. Riemannian walk for incremental learning: Understanding forgetting and intransigence. In *ECCV*, 2018. 2

[15] Arslan Chaudhry, Marc'Aurelio Ranzato, Marcus Rohrbach, and Mohamed Elhoseiny. Efficient lifelong learning with a-gem. In *ICLR*, 2019. 2

[16] Baixu Chen, Junguang Jiang, Ximei Wang, Jianmin Wang, and Mingsheng Long. Debiased pseudo labeling in self-training. *arXiv preprint arXiv:2202.07136*, 2022. 1

[17] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607, 2020. 6

[18] Yanbei Chen, Massimiliano Mancini, Xiatian Zhu, and Zeynep Akata. Semi-supervised and unsupervised deep visual learning: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022. 2

[19] Matthias De Lange, Rahaf Aljundi, Marc Masana, Sarah Parisot, Xu Jia, Aleš Leonardis, Gregory Slabaugh, and Tinne Tuytelaars. A continual learning survey: Defying forgetting in classification tasks. *IEEE transactions on pattern analysis and machine intelligence*, 44(7):3366–3385, 2021. 1, 2, 6

[20] Arthur Douillard, Matthieu Cord, Charles Ollion, Thomas Robert, and Eduardo Valle. Podnet: Pooled outputs distillation for small-tasks incremental learning. In *ECCV*, 2020. 2

[21] Enrico Fini, Victor G Turrisi da Costa, Xavier Alameda-Pineda, Elisa Ricci, Karteek Alahari, and Julien Mairal. Self-supervised models are continual learners. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9621–9630, 2022. 2

[22] Enrico Fini, Stèphane Lathuilière, Enver Sangineto, Moin Nabi, and Elisa Ricci. Online continual learning under extreme memory constraints. In *ECCV*, 2020. 2, 5

[23] Geoffrey Hinton, Oriol Vinyals, Jeff Dean, et al. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2(7), 2015. 5

[24] 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. 2

[25] 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. *Proc. of the national academy of sciences*, 2017. 2, 6

[26] Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, Univ. Toronto, 2009. 5

[27] Samuli Laine and Timo Aila. Temporal ensembling for semi-supervised learning. In *International Conference on Learning Representations*, 2017. 2

[28] Alexis Lechat, Stéphane Herbin, and Frédéric Jurie. Semi-supervised class incremental learning. In *2020 25th International Conference on Pattern Recognition (ICPR)*, pages 10383–10389. IEEE, 2021. 2

[29] Dong-Hyun Lee et al. Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks. In *Workshop on challenges in representation learning, ICML*, volume 3, page 896, 2013. 1, 2

[30] Zhizhong Li and Derek Hoiem. Learning without forgetting. *Trans. PAMI*, 2018. 2, 5, 6

[31] Zicheng Liu, Siyuan Li, Ge Wang, Cheng Tan, Lirong Wu, and Stan Z Li. Decoupled mixup for data-efficient learning. *arXiv preprint arXiv:2203.10761*, 2022. 2

[32] David Lopez-Paz and Marc-Aurelio Ranzato. Gradient episodic memory for continual learning. In *NeurIPS*, 2017. 2, 10

[33] Takeru Miyato, Shin-ichi Maeda, Masanori Koyama, Ken Nakae, and Shin Ishii. Distributional smoothing with virtual adversarial training. In *International Conference on Learning Representations*, 2016. 2

[34] Oleksiy Ostapenko, Mihai Puscas, Tassilo Klein, Patrick Jahnichen, and Moin Nabi. Learning to remember: A synaptic plasticity driven framework for continual learning. In *CVPR*, 2019. 2

[35] Sungrae Park, JunKeon Park, Su-Jin Shin, and Il-Chul Moon. Adversarial dropout for supervised and semi-supervised learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018. 2

[36] Ameya Prabhu, Philip H. S. Torr, and Puneet K. Dokania. Gdumb: A simple approach that questions our progress in continual learning. In *ECCV*, 2020. 6

[37] Siyuan Qiao, Wei Shen, Zhishuai Zhang, Bo Wang, and Alan Yuille. Deep co-training for semi-supervised image recognition. In *Proceedings of the European Conference on Computer Vision*, pages 135–152, 2018. 2

[38] Dushyant Rao, Francesco Visin, Andrei A Rusu, Yee Whye Teh, Razvan Pascanu, and Raia Hadsell. Continual unsupervised representation learning. *NeurIPS*, 2019. 2

[39] Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H. Lampert. iCaRL: Incremental classifier and representation learning. In *CVPR*, 2017. 2, 3, 6

[40] Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H Lampert. icarl: Incremental classifier and representation learning. In *CVPR*, 2017. 10

[41] David Rolnick, Arun Ahuja, Jonathan Schwarz, Timothy Lillicrap, and Gregory Wayne. Experience replay for continual learning. *Advances in Neural Information Processing Systems*, 32, 2019. 6

[42] A. A. Rusu, N. C. Rabinowitz, G. Desjardins, H. Soyer, J. Kirkpatrick, K. Kavukcuoglu, R. Pascanu, and R. Hadsell. Progressive neural networks. *arXiv 1606.04671*. 2

[43] Mehdi Sajjadi, Mehran Javanmardi, and Tolga Tasdizen. Regularization with stochastic transformations and perturbations for deep semi-supervised learning. *Advances in neural information processing systems*, 29, 2016. 2

[44] Joan Serra, Didac Suris, Marius Miron, and Alexandros Karatzoglou. Overcoming catastrophic forgetting with hard attention to the task. *ICML*, 2018. 2

[45] Hanul Shin, Jung Kwon Lee, Jaehong Kim, and Jiwon Kim. Continual learning with deep generative replay. In *NeurIPS*, 2017. 2

[46] James Smith, Jonathan Balloch, Yen-Chang Hsu, and Zsolt Kira. Memory-efficient semi-supervised continual learning: The world is its own replay buffer. In *2021 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2021. 1, 2

[47] James Smith, Cameron Taylor, Seth Baer, and Constantine Dovrolis. Unsupervised progressive learning and the stam architecture. *arXiv preprint arXiv:1904.02021*, 2019. 2

[48] Kihyuk Sohn, David Berthelot, Nicholas Carlini, Zizhao Zhang, Han Zhang, Colin A Raffel, Ekin Dogus Cubuk, Alexey Kurakin, and Chun-Liang Li. Fixmatch: Simplifying semi-supervised learning with consistency and confidence. *Advances in neural information processing systems*, 33:596–608, 2020. 1, 2

[49] Antti Tarvainen and Harri Valpola. Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. *Advances in neural information processing systems*, 30, 2017. 2

[50] Yonglong Tian, Dilip Krishnan, and Phillip Isola. Contrastive multiview coding. In *ECCV*, pages 776–794, 2020. 5

[51] Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(11), 2008. 11

[52] Vikas Verma, Kenji Kawaguchi, Alex Lamb, Juho Kannala, Yoshua Bengio, and David Lopez-Paz. Interpolation consistency training for semi-supervised learning. In *Proceedings of the International Joint Conference on Artificial Intelligence, IJCAI*, pages 3635–3641, 2019. 2

[53] Liyuan Wang, Kuo Yang, Chongxuan Li, Lanqing Hong, Zhenguo Li, and Jun Zhu. Ordisco: Effective and efficient usage of incremental unlabeled data for semi-supervised continual learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5383–5392, 2021. 1, 2, 3, 6, 7, 8

[54] Yue Wu, Yinpeng Chen, Lijuan Wang, Yuancheng Ye, Zicheng Liu, Yandong Guo, and Yun Fu. Large scale incremental learning. In *CVPR*, 2019. 2

[55] Qizhe Xie, Minh-Thang Luong, Eduard Hovy, and Quoc V Le. Self-training with noisy student improves imagenet classification. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10687–10698, 2020. 2

[56] Yi Xu, Lei Shang, Jinxing Ye, Qi Qian, Yu-Feng Li, Baigui Sun, Hao Li, and Rong Jin. Dash: Semi-supervised learning with dynamic thresholding. In *International Conference on Machine Learning*, pages 11525–11536, 2021. 2

[57] Yang You, Igor Gitman, and Boris Ginsburg. Large batch training of convolutional networks. *arXiv preprint arXiv:1708.03888*, 2017. 6

[58] Xiaohua Zhai, Avital Oliver, Alexander Kolesnikov, and Lucas Beyer. S4l: Self-supervised semi-supervised learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1476–1485, 2019. 1

[59] Bowen Zhang, Yidong Wang, Wenxin Hou, Hao Wu, Jindong Wang, Manabu Okumura, and Takahiro Shinozaki. Flexmatch: Boosting semi-supervised learning with curriculum pseudo labeling. *Advances in Neural Information Processing Systems*, 34:18408–18419, 2021. 2

[60] Liheng Zhang and Guo-Jun Qi. Wcp: Worst-case perturbations for semi-supervised deep learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3912–3921, 2020. 2