# 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. A primary issue in this scenario is the model forgetting representations of unlabeled data and overfitting the labeled samples. We leverage the power of nearest-neighbor classifiers to nonlinearly partition the feature space and flexibly model the underlying data distribution thanks to its non-parametric nature. This enables the model to learn a strong representation for the current task, and 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 solid state of the art on the continual semi-supervised learning paradigm. For example, on CIFAR-100 we surpass several others even when using at least 30 times less supervision (0.8% vs. 25% of annotations). Finally, our method works well on both low and high resolution images and scales seamlessly to more complex datasets such as ImageNet-100. The code is publicly available on https://github.com/kangzhiq/NNCSL*

## 1. Introduction

Several efforts have been devoted to the continual learning (CL) [17] paradigm wherein training data samples arrive sequentially. However, most of the state-of-the-art CL methods [10, 11, 19] are based on a strong assumption: the data is fully labeled. This is an unrealistic requirement as labeling data is oftentimes expensive for the expertise required or the amount of annotations, hazardous due to privacy or safety concerns, or impractical in a real-time online scenario. A natural way of tackling this issue is by leveraging the *semi-supervised learning* framework, where not all the data samples are labeled.

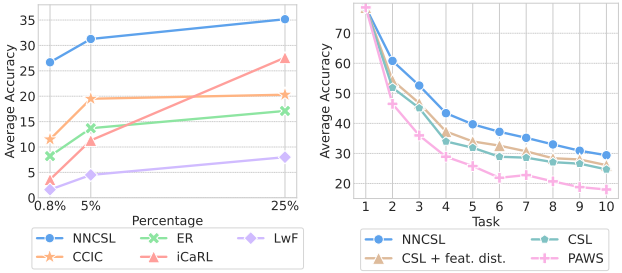



Figure 1: Left: The average accuracy with different percentages of labeled data on CIFAR-100. Our method (NNCSL) with 0.8% of the labels outperforms or matches the performance of all other methods at 25%. Right: Comparison of different versions of our approach and PAWS [3]. CSL is equivalent to NNCSL without our NND loss.

In recent years, this stimulated the community to investigate a new line of research named *continual semi-supervised learning* [7, 44, 51]. It refers to the setting where each task in the sequence is semi-supervised. This learning scenario brings novel challenges, as the models catastrophically forget the representations of unlabeled data while also overfitting the labeled set. This is further exacerbated by another well-studied phenomenon in CL: overfitting the experience replay buffer [9]. These challenges result in vanilla CL methods underperforming, as they lack the ability to extract information from the unlabeled set, thus largely overfitting to the labeled set [52]. On the other hand, semi-supervised learning approaches [14, 28, 46, 57] balance well the labeled and unlabeled sets but cannot handle the continual scenario, and suffer from forgetting even when paired with well-known CL methods (see Fig. 1 (right) and Tab. 1).

A few recent approaches [7, 51] 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.*, on datasets with more classes, more samples, or higher resolution images (see Fig. 1 (left) and results in Sec. 6). For instance, the best-performing related approach obtains only 19.3% accuracy on ImageNet-100 (for reference, our method reaches 56.2%

*Zhiqi Kang and Enrico Fini contributed equally to this work

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

on the same setting). These approaches cannot learn effective representations from unlabeled data and their architecture is not scalable. Therefore, we argue that there is a clear need for more powerful continual semi-supervised learning methods, with a more suitable use of the labeled set, and efficient and stable representation learning from the unlabeled set.

In this paper, we unleash the power of nearest-neighbors in the context of continual semi-supervised learning. In particular, we propose a new method, NNCSL (**N**earest-**N**eighbor for **C**ontinual **S**emi-supervised **L**earning), that leverages the ability of the nearest-neighbor classifier to non-linearly partition the feature space in two ways: i) to learn powerful and stable representations of the current task using a self-supervised multi-view strategy, and ii) to distill previous knowledge and transfer the local structure of the feature space. The latter is achieved through our proposed NND (**N**earest-**N**eighbor **D**istillation), a novel semi-supervised distillation loss that mitigates forgetting in continual semi-supervised learning better than other competitive distillation approaches. In contrast with knowledge distillation [10, 21, 29, 38] and feature distillation [18, 20, 23], which focus exclusively on class-level and sample-level distributions respectively, NND simultaneously distills relationships between classes and samples by leveraging the nearest-neighbor classifier. Overall, NNCSL outperforms all related methods by very large margins on both small and large scale datasets and both low and high resolution images. For instance, as shown in Fig. 1, NNCSL matches or surpasses all others with more than 30 times less supervision (0.8% vs. 25% of annotations) on CIFAR100.

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 representation-level and class-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 [16]. Early strategies for this learning paradigm applied to deep architectures leveraged pseudo-labels and performed self-training based on them [28]. This scheme was later improved with confidence thresholding [2] and adaptive confidence thresholding [55, 58]. More sophisticated methods for incorporating the confidence of the predictions and filtering out spurious samples were also developed, such as FixMatch [46], which employs a student-teacher architecture. Other approaches demonstrated the benefit of co-training [36] and distillation [54].

Another class of approaches was derived with the idea of imposing similar predictions from the network for two samples obtained with different input perturbations [3, 4, 26, 32, 34, 41, 47, 50, 59]. For example, [3] 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. Recently, sample mixing techniques, such as MixUp, were also investigated in the context of semi-supervised learning for improving the model performance on low sample density regions [5, 6, 30]. However, none of the aforementioned works addressed the problem of learning in an incremental setting.

**Continual learning.** Several CL approaches have been proposed in the last few years to learn from data in an incremental fashion. According to a recent survey [17], existing CL methods can be roughly categorized into three groups. The first category comprises regularization-based methods, which address the problem of catastrophic forgetting by introducing appropriate regularization terms in the objective function [11, 18, 21, 23, 29] or identifying a set of parameters that are most relevant for certain tasks [12, 24, 53]. Replay-based methods correspond to the second group, and they store a few samples from previous tasks [8, 13, 31, 38] or generate them [33, 43] in order to rehearse knowledge during the training phases for subsequent tasks. Finally, the third category is parameter isolation methods [40, 42], 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 [7, 27, 44, 51] or no supervision [1, 20, 37, 45]. However, most of them have default settings that are significantly different, e.g., the use of external datasets, and the accessibility of labeled/unlabeled data during continual learning stages, leaving only a few [7, 51] to be comparable in our desired realistic setting. Wang *et al.* [51] 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) [7] is another approach, which leverages metric learning and consistency regularization for

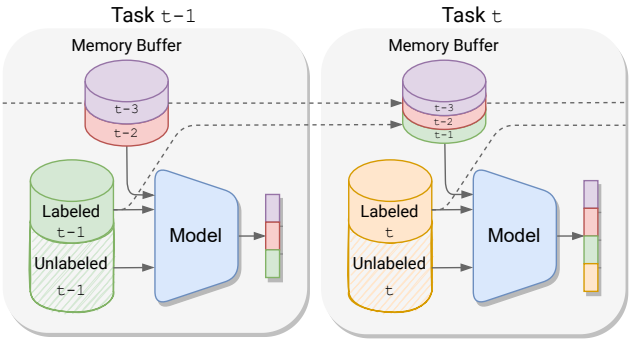



Figure 2: Illustration of the learning process in continual semi-supervised learning.

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 these methods by a large margin.

## 3. Continual semi-supervised learning

We now formally define the problem of continual semi-supervised learning. 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 [7, 38] to allow the retention of a memory buffer $M$ that stores and replays previously seen samples, as shown in Fig. 2.

Let $f_\theta$ be the model, parameterised by $\theta$, 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 this 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 $f_\theta^t$, and for the previous task $t-1$ as $f_\theta^{t-1}$. 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 [3], which inspired our NNCSL, (Sec. 4) and show why this method is not immediately applicable to the continual setting. Subsequently, we present CSL, our base continual semi-supervised learner (Sec. 5.1), which solves many of its issues. However, this base method lacks 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 integrated into in our full method: NNCSL, whose overall objective is summarized in Sec. 5.3.

## 4. Nearest-neighbor meets continual learning: strengths and weaknesses

We now discuss the use of nearest-neighbor techniques [3] in the context of continual learning, 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 from [3] is to assign pseudo-labels for unlabeled samples in a non-parametric manner by considering their relationship with labeled samples, i.e., nearest-neighbor label assignments. 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 $\delta$ be the cosine similarity. One can apply a soft nearest-neighbor classifier to classify the augmented unlabeled sample $\hat{\mathbf{x}}_u$ as follows:

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

where $\boldsymbol{S} = [\mathbf{h}_l^1, ..., \mathbf{h}_l^K]$ are the features of the support samples and $\epsilon$ is a sharpening parameter that controls the entropy of the pseudo-label. Similarly, we can classify the other view of the same sample $\mathbf{v} = \text{SNN}(\mathbf{h}_u, \boldsymbol{S}, \tau)$, 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:

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

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.

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 without overfitting to it. However, PAWS 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 not in the current task's unlabeled set. 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 the learned backbone is fine-tuned using the linear classifier together with the labeled set. This two-stage pipeline (pre-training and then fine-tuning) is prone to a loss of generalizability as the labeled set is very small. *Offline* methods such as PAWS can afford this to gain specialized knowledge on the targeted tasks and computational efficiency (linear classifier vs. nearest-neighbor) at inference time. However, in CL it is unclear which checkpoint of the two-stage methods should be used to further train the model on subsequent tasks. Using the checkpoint after pre-training preserves a more general model, but brings additional memory overhead by storing models from both stages, which is undesirable in a continual setting. In contrast, using the checkpoint after the fine-tuning results in a noticeable loss of performance on the subsequent tasks. 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 $H(\cdot,\cdot)$.

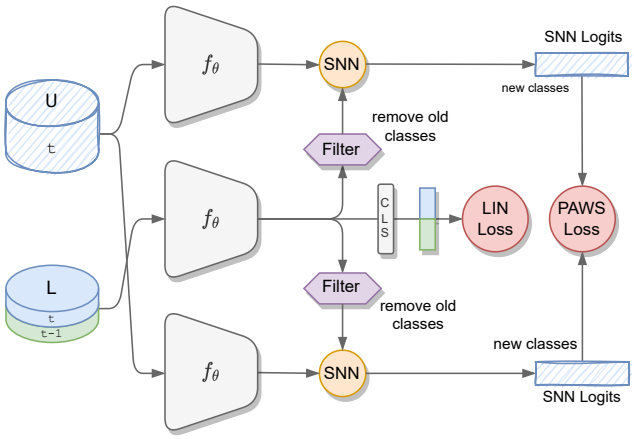



Figure 3: Overview of the base learner component of our method, which does not have a distillation loss. We refer to this as CSL.

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

### 5.1. Our continual semi-supervised learner

We first describe our proposed approach that leverages the strengths of the nearest-neighbor approach described in Sec. 4, while overcoming its weaknesses. 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:

$$\mathcal{L}_{\text{LIN}} = \sum_{j}^{J} \text{H}(\mathbf{p}^k, \mathbf{y}^j), \tag{4}$$

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 CSL) is as follows:

$$\mathcal{L}_{\text{CSL}} = \mathcal{L}_{SNN} + \lambda_{\text{MEM}} \cdot \mathcal{L}_{\text{MEM}} + \lambda_{\text{LIN}} \cdot \mathcal{L}_{\text{LIN}}. \tag{5}$$

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 trained online, and iv) enables us to control the trade-off between fitting labeled or unlabeled data through the parameter $\lambda_{\text{LIN}}$. Interestingly, we found that very small values of $\lambda_{\text{LIN}}$ work well in practice, while larger values in-

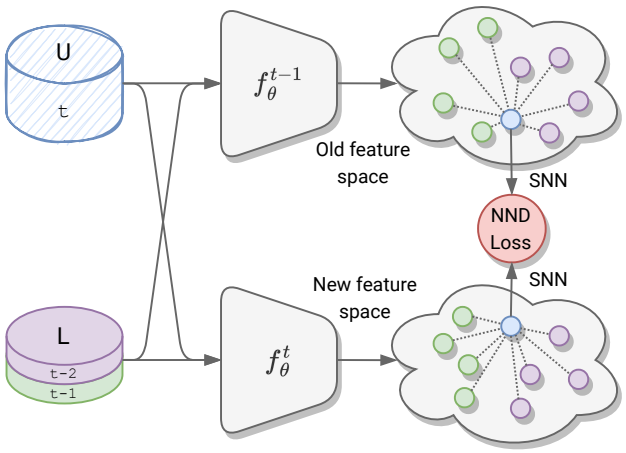



Figure 4: Illustration of our proposed NND loss, which compares the predictions of $f_\theta^t$ and $f_\theta^{t-1}$ with the same unlabeled samples and support samples.

crease 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 CSL is shown in Fig. 3.

### 5.2. Soft nearest-neighbor distillation

Distilling information [22] is a common practice in CL, which utilizes frozen models (or a bank of features and/or probabilities) 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 $f_\theta^t$ aims to mimic the outputs of the teacher $f_\theta^{t-1}$, while learning the new task. Previous works [21, 29] typically distill either the logits of a linear classifier or the features of the hidden layers of the network. However, in CSL, the main driver for the network to learn representations is the loss applied to the soft nearest-neighbor classifier. As explained in Sec. 5.1, 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 well with our framework. The loss is based on the intuition that we can evaluate the nearest-neighbor classifier on the old feature space using the same support samples. This equates to computing the following two vectors: $\mathbf{w} = \text{SNN}(\mathbf{h}_u, \boldsymbol{R}, \tau)$ and $\mathbf{w}^{t-1} = \text{SNN}(\mathbf{h}_u^{t-1}, \boldsymbol{R}^{t-1}, \tau)$,

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:

$$\mathcal{L}_{\text{NND}} = H(\mathbf{w}, \mathbf{w}^{t-1}). \tag{6}$$

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. See Fig. 4 for a visual intuition and supplementary material for the visualization of its impact on deep features. It is worth mentioning that the memory overhead introduced by storing $f_\theta^{t-1}$ can be easily reduced by storing instead the features, which are lightweight compared to images. For instance, storing a 128-dimension feature takes roughly $1/1000^{th}$ memory w.r.t. an RGB image of size 224 $\times$ 224.

Our NND loss is different from the standard distillation loss. Knowledge distillation focuses on class-level distributions calculated by comparing samples of the new tasks with the prototype of each class. This causes a loss of information about representations of individual samples 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, by anchoring unlabeled samples to labeled samples. In addition, NND can capture and transfer non-linear sample-class relationships by definition, unlike knowledge distillation which is limited to linear boundaries. Also, the SNN classifier is computed on a different support set sampled from the buffer at every iteration, which introduces randomness that further regularizes the model. Furthermore, when distilling at very low temperatures (e.g., 0.1), the *softmax* function from Eq. 1 behaves like an *argmax* (i.e., selecting the closest samples in the feature space with high cosine similarity), leading to the transfer of information mainly on the local neighborhood from the teacher to the student. Finally, with respect to feature distillation which lacks alignment with class labels, 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.*, CSL and NND, is trained with the loss:

$$\mathcal{L}_{\text{NNCSL}} = \mathcal{L}_{\text{CSL}} + \lambda_{\text{NND}} \cdot \mathcal{L}_{\text{NND}}. \tag{7}$$

## 6. Evaluation and analysis

### 6.1. Experimental settings

**Datasets**. We evaluate our method on three datasets. CIFAR-10 [25] is a dataset of 10 classes with 50k training and 10k testing images. Each image is of size 32 $\times$ 32. CIFAR-100 [25] is similar to CIFAR-10, except it has 100 classes containing 500 training images and 100 testing images per class. ImageNet-100 [48] is a 100-class subset of

| 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 [29] | 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 [24] | 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 (500) [39] | 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 (500) [38] | 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 (500) [35] | 39.6±9.6 | 40.9±11.8 | 44.8±5.4 | 8.6±0.1 | 9.9±0.4 | 10.1±0.4 |
| PseudoER (500) | 50.5±0.1 | 56.5±0.6 | 57.0±0.6 | 8.7±0.4 | 11.4±0.5 | 12.3±0.2 |
| CCIC [7] (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 [3] (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 |
| PseudoER (5120) | 55.4±0.5 | 70.0±0.3 | 71.5±0.2 | 15.1±0.2 | 24.9±0.5 | 30.1±0.7 |
| CCIC [7] (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 [51] (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.

the ImageNet-1k dataset from the 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 train the models with three different levels of supervision, *i.e.*, $\lambda \in \{0.8\%, 5\%, 25\%\}$. For instance, this corresponds to 4, 25, 125 labeled samples per class in CIFAR-100. As for ImageNet-100, we opt for 1% labeled data. To build the continual datasets, we use the standard setting in the literature [7, 51], 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 [7], we set the buffer size for labeled data as 500 or 5120, to ensure a fair comparison.

**Metrics.** We mainly evaluate the performance of different methods considering the average accuracy over all the seen classes after each task, as is common in CL methods [17]. Analysis with other metrics, such as forward and backward transfer, can be found in the supplementary material.

**Implementation details**. As in [7], we use ResNet18 as our backbone for all the datasets. We adopt the implementation of [3], unless explicitly stated. Specifically, we use the LARS optimizer [56] with a momentum value of 0.9. We set the weight decay as $10^{-5}$, the batch-size as 256. The learning rate is set 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. 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 [15]. 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 at 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 with different random seeds. The standard deviation is also reported, if applicable. Further implementation details and analysis of data augmentation can be found in the supplementary material.

**Baselines.** As baselines, we first consider traditional fully-supervised CL methods. A straightforward way to convert them into a continual semi-supervised setting is to use only the labeled data during training and to discard the unlabeled data. We consider two categories of methods: regularization-based methods, namely, learning without forgetting (LwF) [29], online elastic weight consolidation (oEWC) [24], and replay-based methods, *i.e.*, experience replay (ER) [39], iCaRL [38] and GDumb [35]. We denote PseudoER as an additional two-stage baseline method, which is a combination of semi-supervised learning (PAWS) and CL (ER) methods. More precisely, we continually train a PAWS and use it to self-label the unlabeled data. An ER method is applied afterward on the labeled and pseudo-labeled data. We also consider continual semi-supervised learning baseline methods, such as CCIC [7] and ORDisCo [51]. 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 denote its memory buffer size with the largest value, which is $M = 12500$, equivalent to 25% of CIFAR-10. As for the upper bound, which is commonly shown in CL [17], it is obtained by jointly training the model with all the available data. The lower bound refers to fine-tuning 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 on CIFAR-10 is 92.1±0.1%, and that on CIFAR-100 is 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 ($\leq 0.7$) across all the settings, indicating a better convergence and representation learning during training.

From the results in Tab. 1, the memory buffer is shown to be effective to alleviate forgetting, as replay-based methods significantly outperform regularization-based ones. PseudoER performs better than ER when labeled data is limited (0.8%), but underperforms when the ratio is higher (5%, 25%). We conjecture this as a result of noisy pseudo-labeled data replacing the ground truth in the memory buffer by the sampling strategy of ER, which causes stronger forgetting. In addition, PseudoER is upper-bounded by PAWS, since ER is dependent on the accuracy of pseudo-labeling.

Methods such as CCIC and ORDisCo[†] benefit from their design specific to the continual semi-supervised learning scenario. Even though 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 performs better than 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 $\lambda_{\text{LIN}}$ 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%). In the case of a large-scale dataset such as CIFAR-100, the superiority of NNCSL is clearly evident. Further comparison with the replay strategy of ORDisCo can be found in the supplementary material.

[†]Note that the results of ORDisCo are directly provided by the authors of [51] as there is no open-source implementation of their approach.

| Method | ImageNet-100 | | |
|---|---|---|---|
| | 1% | 5% | 10% |
| Fine-tuning | 1.4±0.2 | 2.0±0.2 | 2.3±0.3 |
| ER [39] (5120) | 9.2±0.2 | 15.5±0.7 | 17.4±0.2 |
| CCIC [7] (5120) | 14.2±0.3 | 17.8±0.2 | 19.3±0.3 |
| CSL (5120) | 26.8±0.4 | 47.9±0.2 | 51.5±0.3 |
| NNCSL (5120) | **29.7**±0.4 | **51.3**±0.1 | **56.2**±0.2 |

Table 2: Average accuracy with standard deviation of different methods tested with 20-task ImageNet-100 settings. The number between brackets indicates the size of the memory buffer for the labeled data.

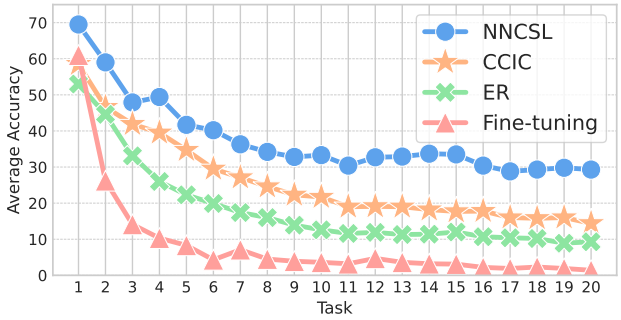



Figure 5: Comparison of NNCSL with existing methods on 20-task ImagetNet-100 and 1% of labeled data. The average accuracy after each learning step is shown.

**ImageNet-100.** We also evaluate on the more challenging dataset of ImageNet-100 with a 20-task continual semi-supervised setting. As shown in Tab. 2, our NNCSL is the best-performing one. CCIC cannot get significant improvement with more labeled data (5% vs. 10%). We suspect this is a limitation of its representation learning for extracting information from images with higher resolution and larger variance, as we observe its training accuracy in this setting is significantly lower. In addition, Fig. 5 visualizes the evolution of average accuracy. We note that the average accuracy of our NNCSL stabilizes at around 30% after the $10^{th}$ task and has a clear rebound between tasks 11 and 16, showing that NNCSL can effectively retain knowledge acquired during the continual learning steps. In contrast, the curves of competitive methods are monotonically decreasing.

### 6.3. Additional analysis

**Ablation**. We ablate the components of our framework in Tab. 3 on both CIFAR-10 (C10 in the table), CIFAR-100 (C100) and ImageNet100 (IN100). The largest improvement comes from filtering (10.8% and 6.2% improvements on CIFAR-10 and CIFAR-100 respectively) and 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

| Method | Component | | | Dataset | | |
|---|---|---|---|---|---|---|
| | Distill | Filter | Linear | C10 | C100 | IN100 |
| PAWS | | | | 51.8 | 16.1 | Collapse |
| CSL (w/o filter) | | | ✓ | 53.0 | 17.2 | Collapse |
| CSL | | ✓ | ✓ | 63.8 | 23.4 | 27.1 |
| NNCSL | ✓ | ✓ | ✓ | **73.2** | **26.8** | **29.3** |

Table 3: Ablation study of the effectiveness of the proposed components on 5-task CIFAR-10 and 10-task CIFAR-100, with $M = 500$ and 0.8% of labeled data, and 20-task ImageNet-100 with $M = 5120$ and 1% of labeled data. We use average accuracy as metrics.

| Method & Distillation | T1 | T2 | T3 | T4 | T5 |
|---|---|---|---|---|---|
| CSL | 25.2 | 22.9 | 24.6 | 37.3 | 37.0 |
| CSL + Knowledge distill | 27.8 | 24.3 | 22.9 | 35.4 | 31.7 |
| CSL + Feature distill | 26.5 | 25.8 | 26.3 | 43.9 | 37.1 |
| CSL + NND | **32.2** | **26.3** | **28.1** | **46.8** | **38.5** |

Table 4: Final accuracy on each task after training on 5-task CIFAR-100. We use CSL as baseline to which we add different distillations. NNCSL is equivalent to CSL + NND.

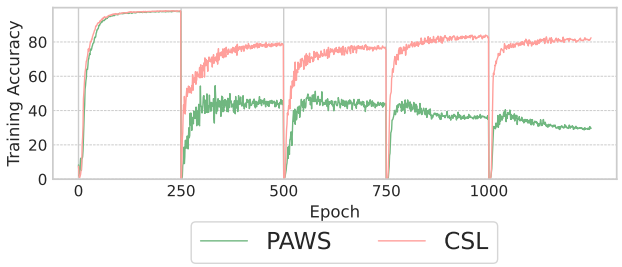



Figure 6: Average training accuracy of unlabeled data on 5-task CIFAR-10. Comparison between vanilla PAWS and our proposed CSL in the continual semi-supervised setting.

does not bring a significant improvement on the overall performance, we find it useful for stabilizing the learning process, especially for a small dataset with very limited supervision. Interestingly, PAWS and CSL (w/o filter) diverged on ImageNet-100, as they cannot learn an effective representation with increased scale, due to the instability introduced by distribution drifts and forgetting of unlabeled data (Sec. 4). We have included additional analysis on this collapse behavior in the supplementary material.

**Impact of distillation.** Complementary to Fig. 1 (right), which qualitatively shows the superior performance of our proposed NND, we report in Tab. 4 the final accuracy of each task after training on all tasks. An effective distillation method should 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 NND vs. knowledge distillation).

**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 is detrimental to the learning as it forces the model to assign incorrect pseudo-labels to the unlabeled data. Our proposed CSL 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.

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

**Linear evaluation head.** We report the train and validation accuracy in Tab. 5. 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 $\lambda_{\text{LIN}}$ 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 $\lambda_{\text{LIN}}$ is smaller than 0.005. It confirms the need for this linear classifier in the continual semi-supervised scenario. We thus set $\lambda_{\text{LIN}}$ as 0.005 in our experiments.

**Ablation of memory buffer.** To verify the effectiveness of the replay buffer, we run our NNCSL without this and observe a drastic decrease in performance (19.7% vs 73.2%) on 5-task CIFAR-10 with 0.8% of labeled data. However, our method recovers 95% of the performance (69.2% vs 73.2%) while using only 10% of memory (50 vs 500). We conjecture such data efficiency results from the linear classifier that provides strong gradients in our method.

## 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 [3] showed that using a more powerful network such as wider or deeper ResNet can further improve performance. While this is not 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 [3] and CSL 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, 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, and 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 [7]. CCIC utilizes random cropping and horizontal flipping (which we refer to as *weak DA*), whereas our proposed CSL 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 (C10 in the table) with 5 tasks, 5% labeled data and buffer size 5120, and also on CIFAR-100 (C100 in the table) with 10 tasks, 0.8% labeled data and buffer size 5120. The results are reported in Tab. 6. CCIC does not benefit from color distortion on both 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 | C100 | **12.0** | 9.9 |

Table 6: Comparison of different data augmentation strategies for CCIC on CIFAR-10 (denoted as C10 in the table) and CIFAR-100 (denoted as C100 in the table).

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

Table 7: Comparison of different strategies for the replay buffer with 5-task CIFAR-10, using 3% of labeled data to match the setting of [51].

## C. Replay strategies

ORDisCo [51] 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. 7 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.

## D. Training analysis on ImageNet-100

It is interesting to see that PAWS diverges in this setting, as is shown in the Tab. 3 of the main paper. 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.

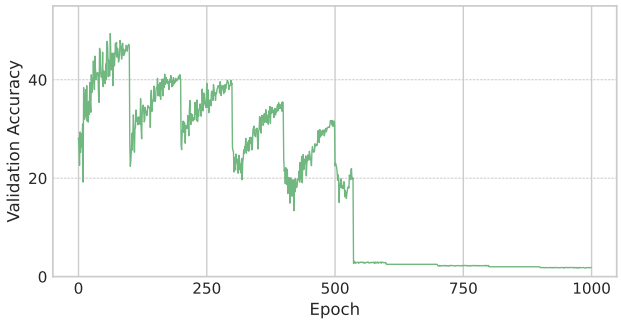
Nevertheless, we believe it may be possible to resolve this collapse without changing the framework, i.e., PAWS. For example, one can conduct careful, extensive hyper-parameter tuning to find an optimal set of parameters that can stabilize the learning. However, it is not realistic given the scale of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

## E. Impact of $\lambda_{\text{NND}}$

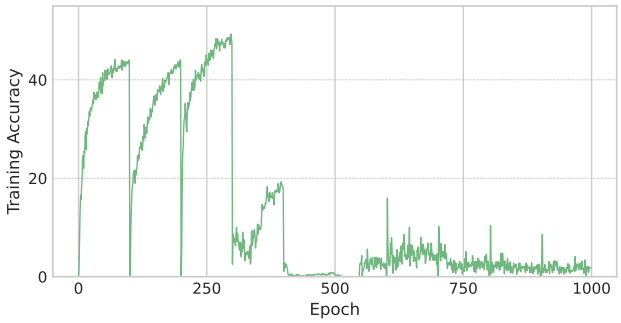
In Tab. 8, we report the performance of our NNCSL with respect to different values of $\lambda_{\text{NND}}$ on CIFAR-100, with 5 tasks, 1% of labeled data and buffer size 5120. $\lambda_{\text{NND}}$ controls the importance of the distillation branch with respect to the PAWS loss. The higher the value, the stronger constraint the model receives to retain the previous knowledge. $\lambda_{\text{NND}} = 0$ means no distillation, which reduces the model back to CSL. 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. $\lambda_{\text{NND}} = 1$).

## F. Forward and backward transfer analysis

Forward transfer (FWT) and backward transfer (BWT) are commonly used in continual learning literature [31, 38]. 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




(a) Validation accuracy of PAWS




(b) Training accuracy for unlabeled data of PAWS

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

| $\lambda_{\text{NND}}$ | 0 | 0.01 | 0.1 | 0.2 | 1 |
|---|---|---|---|---|---|
| NNCSL | 29.0 | 30.2 | 31.8 | **33.6** | 30.5 |

Table 8: Average Accuracy with different values of $\lambda_{\text{NND}}$. These experiments are conducted on CIFAR-100 with 5 tasks, 1% of labeled data and buffer size 5120.

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:

$$FWT = \frac{1}{T-1}\left(\sum_{i=2}^{T} R_{i-1,i} - r_i\right). \tag{8}$$

$$BWT = \frac{1}{T-1}\left(\sum_{i=1}^{T-1} R_{T,i} - R_{i,i}\right). \tag{9}$$

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

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

Table 9: Forward transfer (FWT) and backward transfer (BWT) of PAWS, CSL and NNCSL in 20-task ImageNet-100.

$$ACC = \frac{1}{T}\left(\sum_{i=1}^{T} R_{T,i}\right). \tag{10}$$

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. 8 and Eq. 9.

We report the results in Tab. 9 a comparison of our proposed components. Note that PAWS diverges in this setting, leading to a low FWT. Instead, PAWS is better than CSL and NNCSL if we look at BWT alone. It is simply 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. 9. Thus, BWT is more suitable to be an additional indicator when the average accuracy of the two methods is close to each other, e.g., NNCSL vs. CSL. Comparing NNCSL and CSL, 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 CSL. 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 CSL, which are significantly improved from the BWT scores in Tab. 9.

## G. Visualization of the features

We use t-SNE [49] to project the learned 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.

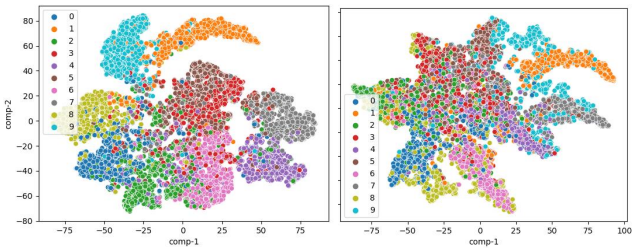



Figure 8: T-SNE visualization of deep features of 10 classes of CIFAR-10, these experiments are conducted with 5 tasks. Left: features from NNCSL after training on task 5, Right: features from PAWS after training on task 5. Data points are colored by their corresponding classes. A clear class boundary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

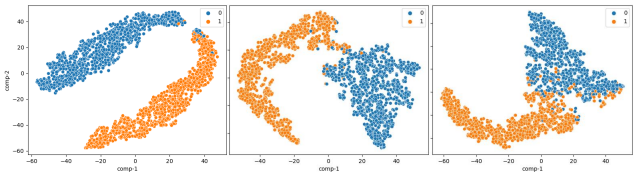



Figure 9: T-SNE visualization of deep features of the first 2 classes of CIFAR-10, these experiments are conducted with 5 tasks. Left: features from NNCSL after training on task 1, Middle: features from NNCSL after training on task 5, Right: features from PAWS after training on task 5. Data points are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

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 on 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)*, 2020. 2

[3] 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 *ICCV*, pages 8443–8452, 2021. 1, 2, 3, 6, 8, 9

[4] 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

[5] 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

[6] David Berthelot, Nicholas Carlini, Ian Goodfellow, Nicolas Papernot, Avital Oliver, and Colin A Raffel. Mixmatch: A holistic approach to semi-supervised learning. *NeurIPS*, 32, 2019. 2

[7] 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

[8] 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

[9] Pietro Buzzega, Matteo Boschini, Angelo Porrello, and Simone Calderara. Rethinking experience replay: a bag of tricks for continual learning. In *ICPR*, 2021. 1

[10] Francisco M Castro, Manuel J Marin-Jimenez, Nicolas Guil, Cordelia Schmid, and Karteek Alahari. End-to-end incremental learning. In *ECCV*, 2018. 1, 2

[11] Hyuntak Cha, Jaeho Lee, and Jinwoo Shin. Co2l: Contrastive continual learning. In *CVPR*, pages 9516–9525, 2021. 1, 2

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

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

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

[15] 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

[16] Yanbei Chen, Massimiliano Mancini, Xiatian Zhu, and Zeynep Akata. Semi-supervised and unsupervised deep visual learning: A survey. *Trans. PAMI*, 2022. 2

[17] 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. *Trans. PAMI*, 44(7):3366–3385, 2021. 1, 2, 6, 7

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

[19] Arthur Douillard, Alexandre Ramé, Guillaume Couairon, and Matthieu Cord. Dytox: Transformers for continual learning with dynamic token expansion. In *CVPR*, pages 9285–9295, 2022. 1

[20] Enrico Fini, Victor G Turrisi da Costa, Xavier Alameda-Pineda, Elisa Ricci, Karteek Alahari, and Julien Mairal. Self-supervised models are continual learners. In *CVPR*, pages 9621–9630, 2022. 2

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

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

[23] 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

[24] 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

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

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

[27] Alexis Lechat, Stéphane Herbin, and Frédéric Jurie. Semi-supervised class incremental learning. In *ICPR*, 2021. 2

[28] 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*, 2013. 1, 2

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

[30] 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

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

[32] 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

[33] 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.

[34] Sungrae Park, JunKeon Park, Su-Jin Shin, and Il-Chul Moon. Adversarial dropout for supervised and semi-supervised learning. In *AAAI*, 2018. 2

[35] 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

[36] Siyuan Qiao, Wei Shen, Zhishuai Zhang, Bo Wang, and Alan Yuille. Deep co-training for semi-supervised image recognition. In *ECCV*, pages 135–152, 2018. 2

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

[38] Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H. Lampert. iCaRL: Incremental classifier and representation learning. In *CVPR*, 2017. 2, 3, 6, 10

[39] David Rolnick, Arun Ahuja, Jonathan Schwarz, Timothy Lillicrap, and Gregory Wayne. Experience replay for continual learning. *NeurIPS*, 32, 2019. 6, 7

[40] 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

[41] Mehdi Sajjadi, Mehran Javanmardi, and Tolga Tasdizen. Regularization with stochastic transformations and perturbations for deep semi-supervised learning. *NeurIPS*, 29, 2016. 2

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

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

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

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

[46] 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. *NeurIPS*, 33:596–608, 2020. 1, 2

[47] Antti Tarvainen and Harri Valpola. Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. *NeurIPS*, 30, 2017. 2

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

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

[50] 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

[51] 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 *CVPR*, pages 5383–5392, 2021. 1, 2, 6, 7, 9

[52] Liyuan Wang, Xingxing Zhang, Hang Su, and Jun Zhu. A comprehensive survey of continual learning: Theory, method and application. *arXiv preprint arXiv:2302.00487*, 2023. 1

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

[54] Qizhe Xie, Minh-Thang Luong, Eduard Hovy, and Quoc V Le. Self-training with noisy student improves imagenet classification. In *CVPR*, pages 10687–10698, 2020. 2

[55] 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

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

[57] Xiaohua Zhai, Avital Oliver, Alexander Kolesnikov, and Lucas Beyer. S4l: Self-supervised semi-supervised learning. In *ICCV*, pages 1476–1485, 2019. 1

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

[59] Liheng Zhang and Guo-Jun Qi. Wcp: Worst-case perturbations for semi-supervised deep learning. In *CVPR*, pages 3912–3921, 2020. 2