A soft nearest-neighbor framework for continual semi-supervised learning: Supplementary Materials

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 [1] 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[1]

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.

[1]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 1: 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 2: Comparison of different strategies for the replay buffer with 5-task CIFAR-10, using 3% of labeled data to match the setting of [6].

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 [2]. 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. 1. 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.

Buffer size	0	8	16	50	500	5120
NNCSL	19.7	37.7	53.2	69.2	73.2	73.7

Table 3: CIFAR-10 average accuracy wrt memory buffer sizes with 5 tasks, 0.8% of labeled data.

C. Replay strategies

ORDisCo [6] 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. 2 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. Ablation study of the memory buffer

We present the ablation study of the memory buffer in a table, as shown in Tab. 3, for better readability. The use of a memory buffer is a common practice in CL and most replay-based methods would fail if the buffer is removed. For instance, we observe a drastic decrease in performance (19.7% vs. 73.2%) when no replay buffer is allowed. In addition, increasing the memory buffer leads to improvements in performance with a certain upper bound. The performance plateaus when the memory buffer is larger than 500 (500 vs. 5000) because the buffer size is larger than the total number of labeled data in this case (i.e., 400 labeled samples).

E. Ambiguity of two-stage methods in CL

We show in Fig. 1 how two-stage methods (pre-training and then fine-tuning) can be adapted to a CL setting. For instance, after training on Task 1, one has to choose the model to be utilized in the subsequent tasks. The left path reuses the pre-trained model. It ensures the generalizability for learning each new task but results in additional memory overhead, as the fine-tuned model has to be saved for testing. In contrast, the right path reuses the fine-tuned model, which leads to a unified model for all tasks. However, the overfitting caused by the small labeled set is detrimental to the generalizability of the model. We explicitly show such loss in Fig.1 by associating the size of the fine-tuned model with its generalizability. On the right path, the fine-tuned model is shrinking due to the overfitting issue.

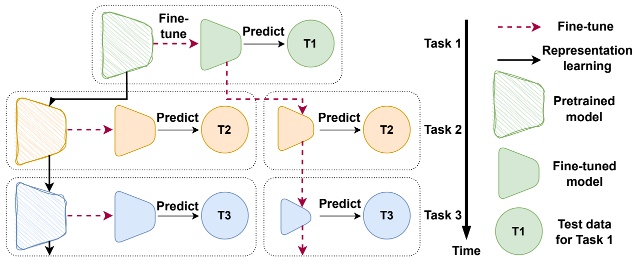


Figure 1: The ambiguity of two-stage methods in CL. On the right path, the shrinking size of the fine-tuned model after each task shows the loss of generalizability.

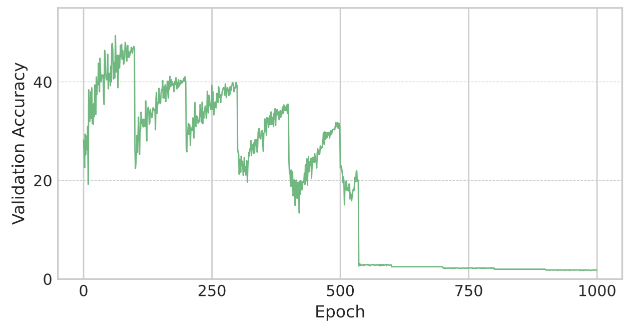
F. Training analysis on ImageNet-100

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. 2. 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.

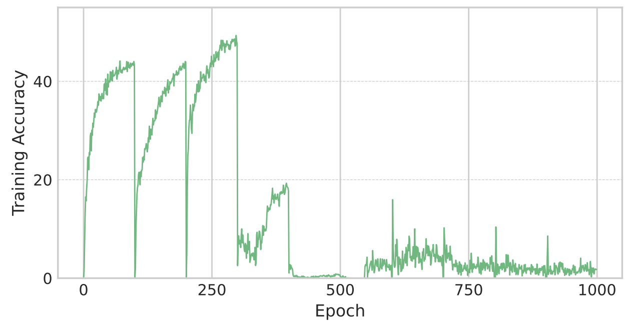
Nevertheless, we believe it may be possible to resolve this collapse without changing the framework, i.e., PAWS. For example, one can conduct careful, extensive hyper-parameter tuning to find an optimal set of parameters that can stabilize the learning. However, it is not realistic given the scale of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

G. Impact of λ_{NND}

In Tab. 4, we report the performance of our NNCSL with respect to different values of λ_{NND} 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 CSL. We can clearly see that distillation helps the



(a) Validation accuracy of PAWS



(b) Training accuracy for unlabeled data of PAWS

Figure 2: Learning curve of PAWS on ImageNet-100.

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

Table 4: 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.

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. $\lambda_{\text{NND}} = 1$).

H. Forward and backward transfer analysis

Forward transfer (FWT) and backward transfer (BWT) are commonly used in continual learning literature [3, 4]. 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:

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

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

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

$$ACC = \frac{1}{T}\left(\sum_{i=1}^{T} R_{T,i}\right). \tag{3}$$

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. 1 and Eq. 2.

We report the results in Tab. 5 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. 2. 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. 5.

I. Visualization of the features

We use t-SNE [5] 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

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

Table 5: Forward transfer (FWT) and backward transfer (BWT) of PAWS, CSL and NNCSL in 20-task ImageNet-100.

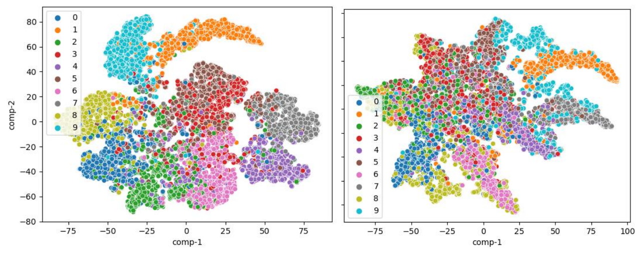


Figure 3: T-SNE visualization of deep features of 10 classes of CIFAR-10, these experiments are conducted with 5 tasks. Left: features from NNCSL after training on task 5, Right: features from PAWS after training on task 5. Data points are colored by their corresponding classes. A clear class boundary after several tasks shows a robust representation along the continual learning stages.

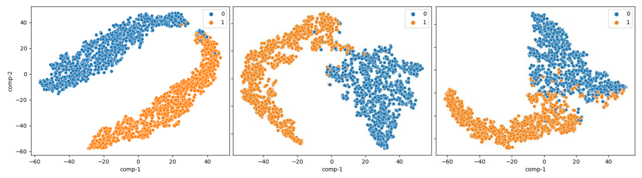


Figure 4: T-SNE visualization of deep features of the first 2 classes of CIFAR-10, these experiments are conducted with 5 tasks. Left: features from NNCSL after training on task 1, Middle: features from NNCSL after training on task 5, Right: features from PAWS after training on task 5. Data points are colored by their corresponding classes. It is clear that PAWS suffers from a blurry class boundary after several continual learning stages.

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. 3. 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. 3 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. 4 confirms that PAWS leads to a blurry boundary and is prone to severe forgetting due to this effect.

References

[1] 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*, 2021. 1

[2] 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

[3] David Lopez-Paz and Marc-Aurelio Ranzato. Gradient episodic memory for continual learning. In *NeurIPS*, 2017. 3

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

[5] Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(11), 2008. 3

[6] 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*, 2021. 1, 2