Efficient Self-Supervised Continual Learning with Progressive Task-correlated Layer Freezing

***Abstract*—Inspired by the success of Self-Supervised Learning (SSL) in learning visual representations from unlabeled data, a few recent works have studied SSL in the context of Continual Learning (CL), where multiple tasks are learned sequentially, giving rise to a new paradigm, namely Self-Supervised Continual Learning (SSCL). It has been shown that the SSCL outperforms Supervised Continual Learning (SCL) as the learned representations are more informative and robust to catastrophic forgetting. However, building upon the training process of SSL, prior SSCL studies involve training all the parameters for each task, resulting to prohibitively high training cost. In this work, we first analyze the training time and memory consumption and reveals that the backward gradient calculation is the bottleneck. Moreover, by investigating the task correlations in SSCL, we further discover an interesting phenomenon that, with the SSL-learned background model, the intermediate features are highly correlated between tasks. Based on these new finding, we propose a new SSCL method with layer-wise freezing which progressively freezes partial layers with the highest correlation ratios for each task to improve training computation efficiency and memory efficiency. Extensive experiments across multiple datasets are performed, where our proposed method shows superior performance against the SoTA SSCL methods under various SSL frameworks. For example, compared to LUMP, our method achieves 1.18x, 1.15x, and 1.2x GPU training time reduction, 1.65x, 1.61x, and 1.6x memory reduction, 1.46x, 1.44x, and 1.46x backward FLOPs reduction, and 1.31%/1.98%/1.21% forgetting reduction without accuracy degradation on three datasets, respectively.**

***Index Terms*—Continual Learning, Self-supervised Learning, Layer Freezing**

I. Introduction

Self-Supervised Learning (SSL) has achieved great success for unsupervised visual representation learning, which aims to learn representation without the need for human annotations. Recent studies (e.g., [1]–[4]) have shown that SSL can achieve comparable or even better performance than the supervised learning counterpart, by utilizing different augmentation views from the same images to generate and optimize contrastiveness. However, SSL still suffers from two major challenges: *(C1)* SSL typically assumes that all training data is available during the training process and learns offline with large amounts of data and resources. In order to integrate new knowledge into the model, the current SSL methods need to train repeatedly on the entire dataset. This may hinder their applications in some real-world scenarios, where new unlabeled data is made available progressively over time and old data becomes unavailable. Also, the learners must be able to cope with non-stationary data in a continuous manner when they are exposed to tasks with varying distributions of data. *(C2)* Compared to supervised learning, obtaining the trained model with the same performance requires much larger training cost in various settings (e.g., heavy model size, larger batch-size, longer training epochs, etc,.).

Fortunately, Continual Learning (CL) [5] provides a promising solution to address the challenge *C1*. Notably, CL aims to incrementally update a model over a sequence of tasks, performing knowledge transfer from the old tasks to the new ones without catastrophic forgetting. A large body of works has been proposed (e.g., [6]–[15]) for the Supervised Continual Learning (SCL) paradigm. Most recently, a few works [16]–[20] have emerged to study CL for the self-supervised learning paradigm, named as *Self-Supervised Continual Learning* (SSCL), which demonstrates that self-supervised visual representations are more robust to catastrophic forgetting compared to supervised learning. Specifically, CaSSLe [16] and PFR [17] propose a similar method that designs a temporal projection module to ensure that the newly learned feature space preserves the information of the previous one. UCL [18] adopts the Mixup [21] technique to interpolate data between the current task and previous tasks' instances to alleviate catastrophic forgetting of the learned representations. However, these SSCL works directly combine the existing SSL frameworks with supervised continual learning techniques (e.g, knowledge distillation, mixup, memory replay, etc,.), while overlooking the substantial training costs inherited from SSL. Compared to Supervised Learning, the training cost of SSL models is notoriously high, and this issue is further exacerbated in SSCL where new tasks arrive continuously. How to improve the training efficiency of SSCL becomes a critical question that needs to be solved, in order to facilitate the development and application of SSCL methods in practice.

In this work, we seek to tackle these problems to reduce training costs while maintaining accuracy for SSCL. Towards this end, we first analyze the training time and memory consumption, revealing two key insights: 1) gradient calculations during the backward pass, rather than the forward pass, account for 70% of the total training time, and 2) intermediate activation storage, not model parameter, consumes 69% of the total memory. This motivates us to explore layer freezing, which has proven to be an effective solution for simultaneously reducing gradient calculation and activation storage. Moreover, by first analyzing the task correlations based on gradient projection in SSCL, we find that the intermediate representations (i.e. activation) learned by self-supervised learning are *highly correlated and varied among tasks* in comparison to SCL. Inspired by this new finding, we propose a new SSCL method with **progressive task-correlated layer freezing (PTLF)** during training for each task to reduce training time and memory cost. Meanwhile, due to the fact that freezing part of layers naturally preserves the knowledge of prior tasks when training on current task, PTLP is able to remain the accuracy by persisting the issue of catastrophic forgetting. Specifically, we first define the *task correlation ratio* according to the gradient projection norm to formally characterize the correlation between the current task and prior tasks. Then, the top-ranked layers with higher task correlation ratios among tasks are progressively frozen during the self-supervised continual learning process for each task. It's also important to note that PTLF is a general method that can be applied on various SSCL methods.

In the experiments, we validate the proposed method against the state-of-the-art SSCL methods on mutiple benchmarks, including Split CIFAR-10, Split CIFAR-100, Split TinyImageNet and ImageNet-100. The experimental results on various SSL frameworks have shown that our method demonstrates superior performance of training efficiency and forgetting over the baseline methods, with comparable or even better accuracy on both settings of class- and task-incremental learning. For example, in comparison to LUMP [18], by using SimSiam [2] SSL framework, our method achieves $1.18\times$, $1.15\times$, and $1.2\times$ GPU training time reduction, $1.65\times$, $1.61\times$, and $1.6\times$ memory reduction, $1.46\times$, $1.44\times$, and $1.46\times$ backward FLOPs reduction, and 1.31%/1.98%/1.21% forgetting reduction without accuracy degradation on three datasets, respectively.

II. Related Work

A. Self-Supervised learning

Self-supervised learning aims to learn visual representation without data labeling cost. Recent advances [1]–[4], [22] show that self-supervised learning can achieve similar or even better performance than supervised representation learning. A common strategy of these methods is to learn representations that are invariant under different data augmentations by maximizing their similarity with contrastive loss optimization. However, these approaches require large-sized batches and negative samples. SimSiam [2] addresses this issue by utilizing the stop-gradient technique to prevent the collapsing of Siamese networks. In addition, given the distorted versions of an instance, BarlowTwin [4] minimizes the redundancy between their embedding vector components while conserving the maximum information. Since SimSiam and BarlowTwim have no requirements for large batch size and negative samples, in this work, we adopt these two works as base learning methods for self-supervised continual learning.

B. Continual Learning

Plentiful continual learning methods have been developed in supervised learning and can be generally divided into three categories: 1) *Regularization-based methods* (e.g., [5], [23], [24]) preserve the knowledge of old tasks by adding an additional regularization term in the loss function, in order to constrain the weight update when learning the new task. 2) *Structure-based methods* (e.g., [8], [25]–[27]) adapt different model parameters or architectures with a sequence of tasks. 3) *Memory-based methods* can be further divided into memory-replay methods and orthogonal-projection based methods. Memory-replay methods (e.g., [28]–[30]) store and replay the old tasks data when learning the new task, while orthogonal-projection based methods (e.g., [11]–[15]) update the model for each new task in the orthogonal direction to the subspace spanned by inputs of old tasks.

More recently, a few works [16]–[19], [31] have emerged to tackle the problem of self-supervised continual learning. They show that self-supervised continual learning can mitigate catastrophic forgetting and learn more general representations compared to supervised continual learning. Specifically, [31] learned task-specific representations on shared parameters. However, it is restricted to simple low-resolution tasks and not scalable to standard CL benchmark datasets. In addition, CaSSLe [16] and PFR [17] propose a similar method that designs a temporal projection module to ensure that the newly learned feature space preserves the information of the previous one. LUMP [18] adapts the Mixup [21] technique to interpolate data between the current task and previous tasks' instances to alleviate catastrophic forgetting for unsupervised representations. Kaizen [20] proposes to leverage knowledge distillation to mitigate catastrophic forgetting, while a classifier is continually trained with supervision. However, these works directly combine the existing self-supervised with continual learning techniques (e.g, knowledge distillation, mixup, memory replay, etc,.) that still suffer from large training costs.

C. Layer Freezing

There existed several works on accelerating the training of deep neural networks for one single task by using layer freezing techniques [32]–[36]. These works are mainly motivated by the fact that front layers mainly extract general features of the raw data (e.g., the shape of objects) and are easier to well-trained, while deeper layers are more task-specific and capture complicated features output from front layers. Specifically, Liu et al. [32] propose to automatically freeze layers during training according to the parameter gradients. Wang et al. [34] adopt knowledge distillation [35] to guide the layer freezing schedule. [36] apply layer freezing on sparse training. However, these works primarily focus on training one single task. Different from all these related works, we study the layer freezing in SSCL with the consideration of task correlations.

III. Method

A. Problem Setup

In Supervised Continual Learning (SCL), a model continuously learns from a sequential data stream in which new tasks (namely, classification tasks with new classes) are added over time. More formally, we consider a sequence of tasks $\{1, 2, ..., T\}$ where the task at time t comes with training data $D_t = \{\boldsymbol{x}_{t,i}, \boldsymbol{y}_{t,i}\}_{i=1}^{N_t}$. Note that each task t can contain a sequence of classes. We denote $f(\cdot)$ as the operation of a feature extractor and $g(\cdot)$ as a classifier model. The main objective is to optimize the parameter $\boldsymbol{w}$ of both the feature extractor and the classifier:

$$\min_{\boldsymbol{w}_f,\boldsymbol{w}_h} \sum_{t=1}^{T}\sum_{i=1}^{N_t}\mathcal{L}_t(g(f(\boldsymbol{x}_{t,i})), \boldsymbol{y}_{t,i}) \qquad (1)$$

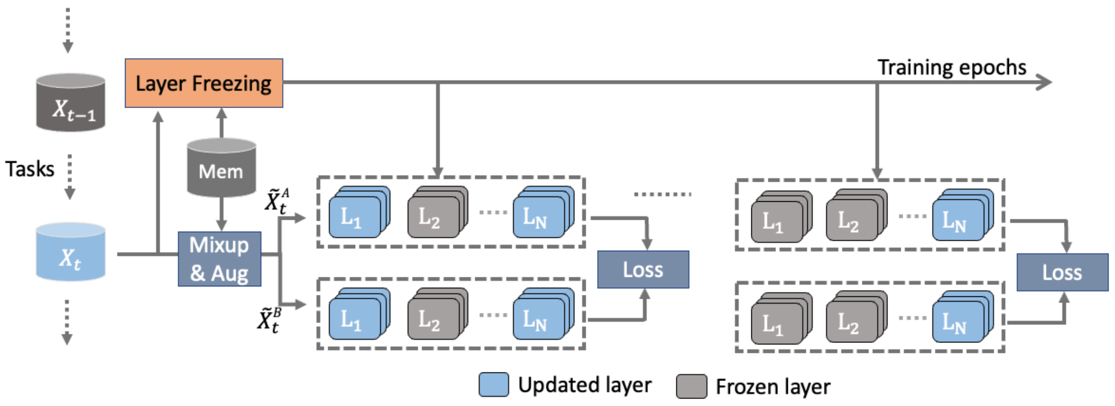
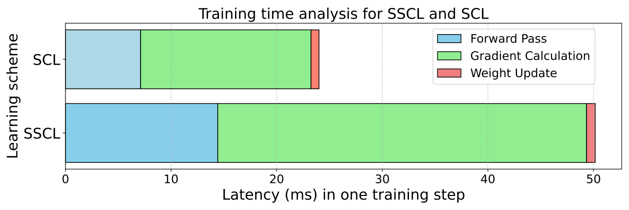
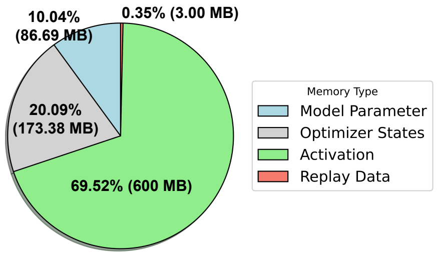


Fig. 1: The overview of our proposed method which progressively freezes partial layers during the whole training process for each task.



(a) Training time analysis.



(b) Training memory anaysis on SSCL.

Fig. 2: Training time and memory analysis by using ResNet18 as backbone model and SimSam as self-supervised learning method for SSCL under 256 batch size.

where $\mathcal{L}_t(\cdot)$ is cross entropy loss function in general.

In contrast, self-supervised continual learning does not require labels during training. We denote $h(\cdot)$ as objection layer, the objective is to learn a general representation that is invariant to augmentations on all tasks, which can be formulated as:

$$\min_{\boldsymbol{w}_f} \sum_{t=1}^{T} \sum_{i=1}^{N_t} \mathcal{L}_t(h(f(\boldsymbol{x}_{t,i}^1)), f(\boldsymbol{x}_{t,i}^2)) \tag{2}$$

where $\boldsymbol{x}_{t,i}^1$ and $\boldsymbol{x}_{t,i}^2$ are augmented images generated from $\boldsymbol{x}_{t,i}$. The choices of the feature extractor and loss function depend on the self-supervised learning method, such as SimSiam and BarlowTwin. After the feature extractor is learned on all tasks, following [18], we use K-nearest neighbor (KNN) classifier [37] or linear classification to evaluate the performance of the learned representation. In this work, we investigate self-supervised continual learning on the settings of both task- and class- incremental learning [38].

B. SSCL Suffers From High Training Cost

Compared to Supervised Continual learning, Self-Supervised Continual Learning suffers from much higher training cost. The main reason is that as shown in Eq. 2, SSCL has a Siamese architecture where two augments of the same images will pass through the same feature extractor $f(\cdot)$ (i.e., ResNet-18), respectively. Moreover, projection layer $h(\cdot)$ in SSCL has much larger parameter size than classifier $h(\cdot)$ in SCL. To understand the training cost of SSCL, we quantitatively evaluate the training cost by measuring the actual training time and calculate memory consumption[1] in one training step for both SCL and SSCL.

Analysis on training time. Fig. 2(a) presents the training time analysis for SCL and SSCL respectively. In general, the training process can be divided into two consecutive steps: the forward pass and the backward pass. The backward pass can further be broken down into two phases: gradient calculation and weight update. As shown in Fig. 2(a), it's clear to see that 1) SSCL suffers from substantial training cost that is $\sim 2\times$ compared to SCL. 2) For SSCL, the gradient calculation is the bottleneck accounting for approximately 70% of the total training time. 3) Weight update is computationally lightweight and takes the same amount of time in both SSCL and SCL.

Analysis on training memory. Fig. 2(b) presents the memory usage during training. The whole training memory usage can be categorized into: model parameter size, intermediate activation storage of each layer, optimizer states which store the parameter gradients and related variables, and replay data that stores a number of data from prior tasks for continual learning. First and foremost, we observe that intermediate activation storage is the primary bottleneck, consuming approximately 70% of the total memory usage. Additionally, optimizer states require twice the memory of model parameters due to the momentum schedule in modern optimizers (e.g., SGD), which doubles the gradient storage. In contrast, replay data occupies a minimal amount of memory compared to other types.

In summary, gradient calculation during the backward pass and intermediate activation storage are the main bottlenecks for

[1]We calculate the memory cost due to lack of well-supported tool to measure fine-grained memory usage in current machine learning framework.

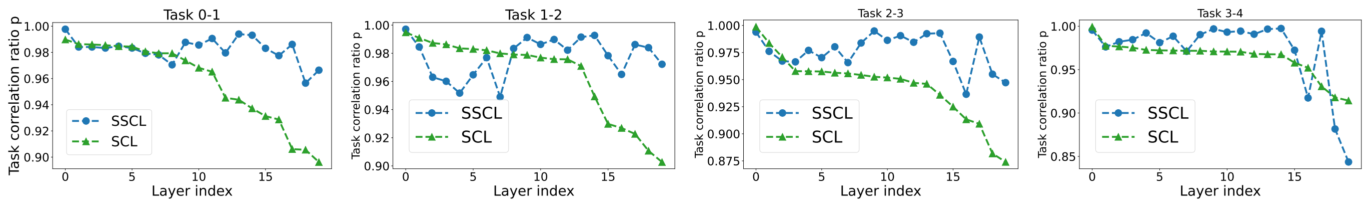


Fig. 3: Layer-wise correlation ratio between two tasks on Split CIFAR-10 using ResNet18 for SSCL and SCL respectively.

both training time and memory usage. Meanwhile, layer freezing—by freezing unimportant layers during training—proves to be an effective technique to address these issues, as frozen layers do not require gradient calculations or storage of corresponding activations for the backward pass. These observations motivated us to further explore how to perform layer freezing in self-supervised continual learning.

C. Proposed Method

In this work, we propose progressive task-correlated layer freezing (PTLF) for self-supervised continual learning to reduce training time and memory consumption while maintaining accuracy. Fig. 1 presents the overview of the proposed method. Specifically, for training each task in continual learning, following the standard practice [16]–[18], [31], we employ the memory replay method [10] to store a uniform sample of data from each previous task using a fixed buffer size. The replayed data are then combined with the current task's data during training to optimize the self-supervised loss as shown in Eq. 2. More importantly, the proposed method progressively freezes portions of important layers from prior tasks based on their task correlations. The detailed method will be illustrated below.

*1) **Task correlation indicates the layers to be frozen**:* As revealed by prior works [16]–[18], [31], the learned representations of SSCL are more general and robust to catastrophic forgetting than the representations of SCL. We conjecture the reason behind is the learned intermediate features for each layer between the current task and prior tasks are highly correlated with each other. Accordingly, if the new task t has strong similarity with old tasks in some layers, it is possible that not updating these layers will not significantly affect the learning performance of this new task. Thus inspired, we raise the question: *Can we leverage the generality of the learned representations from SSL and freeze the highly correlated layers during training for each task to improve the training efficiency?*

To answer this question, motivated by prior gradient orthogonal-projection based methods [14], [15] on SCL, we first investigate the correlation of tasks according to gradient projection. Specifically, to formally characterize the correlation between the current task and prior tasks, we define the **task correlation ratio** in layer-wise as:

$$p_t^l = \frac{\|\mathrm{Proj}_{S_t^l}(\nabla \mathcal{L}_t(\boldsymbol{w}_{t-1}^l))\|_2}{\|\nabla \mathcal{L}_t(\boldsymbol{w}_{t-1}^l))\|_2} \tag{3}$$

where $\mathrm{Proj}_{S_t^l}$ denotes the projection on the input subspace S_t^l of prior tasks $(1, 2, ..., t-1)$ on l^{th} layer, and $\boldsymbol{w}_{t-1}^l$ represents the l^{th} layer weight in the model before learning task t. Here $\mathrm{Proj}_S(A) = AB(B)'$ for some matrix A and B is the bases for S. Accordingly, for each task t, we can generate a set of task correlation ratios $P_t = [p_t^1, p_t^2, ..., p_t^S]$ where S is the total number of layers. Due to the fact that the gradient lies in the span of the input [13], if the task correlation ratio $p_t^l \in (0, 1)$ has a large value, it implies that the current task t and prior tasks may have sufficient common bases in l^{th} layer between their input subspaces and hence are strongly correlated. To quantitatively evaluate the task correlation on SSCL, we conduct the experiments on three settings (i.e., Split CIFAR-10, Split CIFAR-100, Split TinyImageNet) by using SimSiam for continual learning. As shown in 3, we observe that:

Observations: *1) the variance of the task correlation ratio in SSCL is smaller than the counterparts in SCL; 2) the correlation ratios of SSCL are larger than the counterparts in SCL for most layers; 3) the correlation ratios of SCL consistently follow an ascending order, while the counterparts in SSCL are more varied that are usually higher for top and middle layers.*

The first two observations help to further explain that the learned representations of SSCL are more general than SCL. Moreover, the third observation indicates that following ascending order to freeze layer in supervised learning is a good choice [32], [36], [39] since the correlation ratios of the top layers are always larger than the later ones. However, for the SSCL, layer freezing needs to consider task correlations between tasks. In the experiments, we also evaluate that task-correlated layer freezing could show better accuracy than conventional layer freezing in ascending order in the Section V.

*2) **Progressive layer freezing via the defined task correlation**:* First of all, we calculate the task correlation ratios P_t in one-shot before training each task. As shown in Eq. 3, the input subspace of prior tasks is needed for this calculation. To obtain it, we use replayed data from prior tasks to perform a forward pass on the current model and calculate the bases of the subspace by applying Singular Value Decomposition (SVD) to the intermediate activations in each layer.

Based on the obtained task-correlation ratios, we develop progressive task-correlated freezing in SSCL to progressively increase the number of frozen layers with the highest correlation ratios during training for each task. In practice, we define the initial freeze ratio as r_i and final freeze ratio as r_f, which denote the ratio of the number of frozen layers to the number of layers in the neural network. The total number of training epochs is N and the current training epoch is n. To schedule the freeze ratio during training for each task, we consider two options to progressively increase the freeze ratio: linear

TABLE I: Accuracy and forgetting of the learned representations on Split CIFAR-10, Split CIFAR-100 and Split Tiny-ImageNet on ResNet-18 architecture with KNN classifier. All the values are measured by computing mean and standard deviation across three trials. Note that, we use the layer freezing ratio as 0.4 by default for all our results.

	Method	SPLIT CIFAR-10					SPLIT CIFAR-100					SPLIT TINY-IMAGENET				
		Accuracy (%)	Forgetting (%)	Time (ms)	Memory (MB)	FLOPs (e^{11})	Accuracy (%)	Forgetting (%)	Time (ms))	Memory (MB)	FLOPs (e^{11})	Accuracy (%)	Forgetting (%)	Time (ms)	Memory (MB)	FLOPs (e^{11})
Simsiam	Finetune	90.11	5.43	-	-	-	75.42	10.19	-	-	-	71.07	9.48	-	-	-
	PNN [6]	90.93	-	46	819	8.54	66.58	-	47	819	8.54	62.15	-	100	2628	34.1
	SI [7]	92.75	1.81	46	689	8.54	80.08	5.54	47	689	8.54	72.34	8.26	101	2498	34.1
	DER [10]	91.22	4.63	47	689	8.54	77.27	9.31	47	689	8.54	71.90	8.36	101	2400	34.1
	LUMP [18]	91.00	2.92	45	689	8.54	**82.30**	4.71	45	689	8.54	76.66	3.54	102	2498	34.1
	LUMP-Ours	91.03	**1.61**	**38**	**417**	**5.81**	82.24	**2.73**	**39**	**422**	**5.93**	**76.68**	**2.33**	**82**	**1506**	**23.2**
	Multitask	95.76	-	1x	1x	1x	86.31	-	1x	1x	1x	82.89	-	1x	1x	1x
BarlowTwin	Finetune	87.72	4.08	1x	1x	1x	71.97	9.45	1x	1x	1x	66.28	8.89	1x	1x	1x
	PNN [6]	87.52	-	148	819	8.54	57.93	-	148	681	8.54	48.70	-	272	2510	34.1
	SI [7]	**90.21**	2.03	153	681	8.54	75.04	7.43	152	681	8.54	56.96	17.04	278	2490	34.1
	DER [10]	88.67	2.41	142	681	8.54	73.48	7.98	151	681	8.54	68.56	7.87	256	2490	34.1
	LUMP [18]	89.72	1.13	149	681	8.54	80.24	3.53	146	681	8.54	72.17	2.43	274	2490	34.1
	LUMP-Ours	89.73	**0.92**	**123**	**439**	**5.90**	**80.54**	**2.24**	**116**	**412**	**5.68**	**73.56**	**1.74**	**230**	**1606**	**22.7**
	Multitask	95.48	-	-	-	-	87.16	-	-	-	-	82.42	-	-	-	-

scheduling and cosine annealing. The experimental results show that both scheduling methods achieve same performance, and we adopt cosine annealing by default. Following that, once getting the freeze ratio for the current epoch, we adopt the following strategies to progressively and accumulately freeze the layers: 1) the layers with the highest task-correlation ratio under the current freeze ratio r_n will be frozen; 2) the frozen layers of prior epochs will be unchanged, and we will gradually increase the number of frozen layers according to the freeze ratio difference $(r_n - r_{n-1})$. This can be achieved by using a TopK function to select the layers to be frozen according to the layer-wise task correlation Importantly, one practical reason that we choose layer-wise freezing is that it could enable actual training speedup in GPU by using general deep learning frameworks (e.g, Tensorflow, Pytorch). In our experiments, we set the initial and final freeze ratios as 0 and 0.4 for all tasks by default.

IV. Experimental Results

A. Experimental Setup

Baseline. We compare with multiple self-supervised continual learning baselines across different categories of continual learning methods on CIFAR-10, CIFAR-100 and Tiny-ImageNet dataset, respectively. As a general method to improve memory efficiency for SSCL, we mainly adopt the proposed PTFL on prior SOTA SSCL works CaSSLe [17] and LUMP [18] on class incremental learning and task incremental learning, respectively. In addition, following [18], we also report several self-supervised variants of SCL methods. Specifically, FINETUNE is a vanilla supervised learning method trained on a sequence of tasks without regularization or episodic memory and MULTITASK optimizes the model on complete data. SI [7] is a regularization-based CL method, PNN [6] is a architecture-based method, and DER [10] is a memory replay method which adapts knowledge distillation by memory replay to match the network logits sampled through the optimization trajectory during continual learning.

Metrics. Following [18], two metrics are used to evaluate the performance: *Accuracy*, the average final accuracy over all tasks, and *Forgetting*, which measures the forgetting of each task between its maximum accuracy and accuracy at the completion of training. Furthermore, we utilize three metrics to measure training efficiency: Training time, we report the training time ratio compared to LUMP baseline which is measured on NVIDIA RTX A6000 GPU; memory, which includes model parameter size, training activation storage, and memory replay buffer size; Flops, which calculate the number of computational operations during backward.

B. Main Results

1) Task-incremental learning: As shown in Table I, we evaluate the performance of various SSCL methods, by using SimSiam [1] and BarlowTwin [4] SSL frameworks on Split CIFAR-10, Split CIFAR-100 and Split Tiny-ImageNet, respectively. In all experiments, we set the initial and final freeze ratios as 0 and 0.4 for all tasks by default. For SimSiam framework, our method achieves $1.18\times$, $1.15\times$, and $1.2\times$ training time speedup, $1.65\times$, $1.61\times$, and $1.6\times$ memory reduction, and $1.46\times$, $1.44\times$, and $1.46\times$ backward FLOPs reduction on three datasets respectively. Similarly, for the BarlowTwin framework, our method achieves $1.21\times$, $1.23\times$, and $1.19\times$ training time speedup, $1.55\times$, $1.67\times$, and $1.55\times$ memory reduction, and $1.44\times$, $1.50\times$, and $1.50\times$ backward FLOPs reduction on three datasets respectively. Importantly, *our method clearly mitigates the forgetting issue in comparison to all prior methods.* For example, compared to LUMP on Split CIFAR-100 and Split Tiny-ImageNet for Simsiam, we reduce the forgetting by 1.31%, 1.98% and 1.21% respectively with similar accuracy.

2) Class-incremental learning: Moreover, we adopt the proposed PTLF on CaSSLe on ImageNet-100 dataset for class incremental learning by using two backbone SSL frameworks, BarlowTwin and MoCoV2. As shown in Table II, similar results can be found that the proposed method shows clear gain to improve training efficiency and maintain accuracy by mitigating catastrophic forgetting.

3) The setting of 5-dataset: To further evaluate the effectiveness of the proposed method, we conduct the experiments on the more challenging setting of 5 datasets as shown in

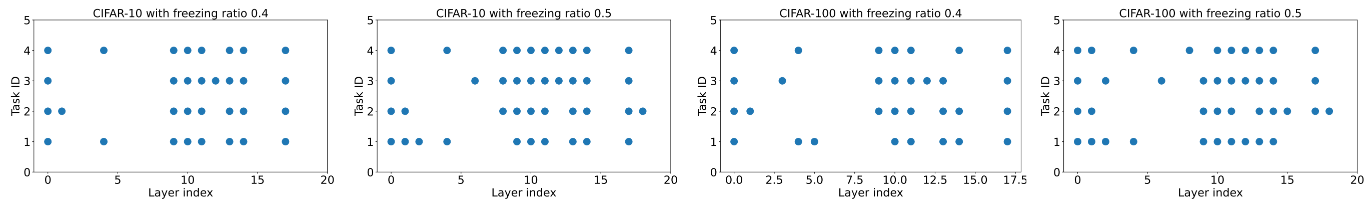


Fig. 4: The final selection of updated layer for each task where the freezing ratio is 0.4. Note that, each blue point means the index of the frozen layer. For Split CIFAR-100 20 tasks setup, we show the first five tasks for simplification.

TABLE II: Accuracy, forgetting, training time, and training memory cost on ImageNet-100 with linear evaluation.

Setting		ImageNet-100				
		Accuracy (%)	Forgeting (%)	Time (ms)	Memory (MB)	FLOPs (e^{11})
BarlowTwin	CaSSLe	68.2	1.3	630	9857	106
	CaSSLe-Ours	**68.5**	**0.7**	**523**	**5271**	**75.5**
MoCoV2	CaSSLe	**68.0**	2.2	729	11025	150
	CaSSLe-Ours	67.9	**1.4**	**616**	**7232**	**82.5**

TABLE III: Accuracy and forgetting on the setting of 5-dataset.

	CIFAR-100	CIFAR10	MNIST	FMNIST	SVHN	Avg Acc	Forgetting
LUMP	56.43	74.12	85.36	83.48	80.73	76.02	11.20
LUMP-Ours	56.43	74.01	85.33	83.50	80.74	76.0	6.82

the Table III. Compared to LUMP, we achieve the same accuracy while reducing forgetting. Notably, we find that the indices of frozen layers are varied for different tasks compared to the setting of in-class splits (e.g., Split CIFAR-10). This observation demonstrates that our method can automatically freeze partial layers based on task correlations that cannot be achieved by manual layer selection.

4) The phenomenon of maintaining accuracy by mitigating catastrophic forgetting: In all experiments, it's interesting to find that the proposed PTLF can persist the final average accuracy while showing clear forgetting reduction compared to baseline methods like LUMP and CaSSLe. This observation suggests that PTLF might result in lower peak accuracy during training for certain tasks. However, it effectively retains crucial knowledge for each task. It's important to note that achieving higher peak accuracy for individual tasks does not necessarily equate to achieving a higher final average accuracy. In the context of continual learning, it is crucial to holistically assess both accuracy and forgetting.

TABLE IV: The ablation study on the proposed method in comparison to layer freezing in ascending layer index (i.e., "Bottom Layer") order by using BarlowTwin as backbone.

Setting	Split CIFAR-10		Split CIFAR-100	
	Forgetting	Accuracy	Forgetting	Accuracy
Bottom Layers	0.96	89.24	2.57	78.87
Ours	**0.92**	**90.03**	**2.24**	**80.54**

V. Ablation Study and Anaysis

a) Task-correlated layer freezing VS ascending order layer freezing: To evaluate the effectiveness of the proposed progressive layer freezing via task correlation, we compare it to ascending order layer freezing which is commonly used in supervised learning settings to improve the training efficiency for a single task. Specifically, for a fair comparison, we also adopt the same cosine annealing to progressively freeze layers under the same freezing ratio (i.e., 0.4). As shown in the Table IV, our method can consistently achieve better accuracy with similar forgetting compared to layer freezing in ascending order. The results demonstrate that the task correlation between tasks needs to be considered in SSCL.

b) The effectiveness of the layer freezing ratio: To understand the impact of the freezing ratio, we evaluate the learning performance for four different values of p (i.e., 0.7, 0.6, 0.5, 0.4) on SPLIT CIFAR-10 dataset by using BarlowTwin method as shown in Table V. It can be seen that with the freezing ratio increasing, the learning performance in terms of both accuracy and forgetting gradually degrades, but to a slight extent. As a benefit, the computational costs in terms of both training time and memory significantly decrease. It makes sense that by fixing more layers, it has a certain degree of negative effect on learning each new task, meanwhile, it indicates less forgetting.

TABLE V: The ablation study on layer-wise freezing ratios.

Freeze ratio	Accuracy	Forgetting	Time	Memory
0.4	89.73	0.86	123	439
0.5	89.01	0.75	118	320
0.6	88.40	0.66	110	249
0.7	87.62	0.60	104	209

c) Self-supervised continual learning is robust to layer freezing decision: We analyze the layer freezing decision by using the freezing ratio of 0.4 and 0.5 for each task on Split CIFAR-10 and Split CIFAR-100 respectively. As shown in Fig 4, there are mainly two observations: 1) for inter-tasks, the indexes of layer freezing decisions are highly similar, which means that the selected frozen layers at the first task will not be updated across all the rest tasks. It further helps to show that the learned representation by SSCL is general and robust; 2) for intra-task, it is interesting to see that and a large number of front layers are updated while the first layer (e.g., index as 0) remain frozen during training. We conjecture the reason is that the first layer learn the low-level general features.

VI. Conclusion

In this work, we first investigate the task correlation of SSCL and find that intermediate features are highly correlated between tasks. Based on this, we propose a progressive task-correlated layer freezing method that freezes gradually partial layers with the highest correlation ratios for each task. Extensive experiments across multiple datasets clearly show that our method can significantly improve training computation and memory efficiency meanwhile mitigating catastrophic forgetting, compared to the SoTA SSCL methods.

References

[1] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, "A simple framework for contrastive learning of visual representations," in *International conference on machine learning*, pp. 1597–1607, PMLR, 2020.

[2] X. Chen and K. He, "Exploring simple siamese representation learning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 15750–15758, 2021.

[3] K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick, "Momentum contrast for unsupervised visual representation learning," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 9729–9738, 2020.

[4] J. Zbontar, L. Jing, I. Misra, Y. LeCun, and S. Deny, "Barlow twins: Self-supervised learning via redundancy reduction," in *International Conference on Machine Learning*, pp. 12310–12320, PMLR, 2021.

[5] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska, *et al.*, "Overcoming catastrophic forgetting in neural networks," *Proceedings of the national academy of sciences*, vol. 114, no. 13, pp. 3521–3526, 2017.

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

[7] F. Zenke, B. Poole, and S. Ganguli, "Continual learning through synaptic intelligence," in *International Conference on Machine Learning*, pp. 3987–3995, PMLR, 2017.

[8] J. Yoon, E. Yang, J. Lee, and S. J. Hwang, "Lifelong learning with dynamically expandable networks," *arXiv preprint arXiv:1708.01547*, 2017.

[9] R. Aljundi, M. Lin, B. Goujaud, and Y. Bengio, "Gradient based sample selection for online continual learning," *Advances in neural information processing systems*, vol. 32, 2019.

[10] P. Buzzega, M. Boschini, A. Porrello, D. Abati, and S. Calderara, "Dark experience for general continual learning: a strong, simple baseline," *Advances in neural information processing systems*, vol. 33, pp. 15920–15930, 2020.

[11] G. Zeng, Y. Chen, B. Cui, and S. Yu, "Continual learning of context-dependent processing in neural networks," *Nature Machine Intelligence*, vol. 1, no. 8, pp. 364–372, 2019.

[12] M. Farajtabar, N. Azizan, A. Mott, and A. Li, "Orthogonal gradient descent for continual learning," in *International Conference on Artificial Intelligence and Statistics*, pp. 3762–3773, PMLR, 2020.

[13] G. Saha, I. Garg, and K. Roy, "Gradient projection memory for continual learning," *arXiv preprint arXiv:2103.09762*, 2021.

[14] S. Lin, L. Yang, D. Fan, and J. Zhang, "Trgp: Trust region gradient projection for continual learning," *arXiv preprint arXiv:2202.02931*, 2022.

[15] S. Lin, L. Yang, D. Fan, and J. Zhang, "Beyond not-forgetting: Continual learning with backward knowledge transfer," *arXiv preprint arXiv:2211.00789*, 2022.

[16] E. Fini, V. G. T. da Costa, X. Alameda-Pineda, E. Ricci, K. Alahari, and J. Mairal, "Self-supervised models are continual learners," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9621–9630, 2022.

[17] A. Gomez-Villa, B. Twardowski, L. Yu, A. D. Bagdanov, and J. van de Weijer, "Continually learning self-supervised representations with projected functional regularization," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 3867–3877, 2022.

[18] D. Madaan, J. Yoon, Y. Li, Y. Liu, and S. J. Hwang, "Representational continuity for unsupervised continual learning," in *International Conference on Learning Representations*, 2021.

[19] D. Hu, S. Yan, Q. Lu, H. Lanqing, H. Hu, Y. Zhang, Z. Li, X. Wang, and J. Feng, "How well does self-supervised pre-training perform with streaming data?," in *International Conference on Learning Representations*, 2022.

[20] C. I. Tang, L. Qendro, D. Spathis, F. Kawsar, C. Mascolo, and A. Mathur, "Kaizen: Practical self-supervised continual learning with continual fine-tuning," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pp. 2841–2850, 2024.

[21] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz, "mixup: Beyond empirical risk minimization," *arXiv preprint arXiv:1710.09412*, 2017.

[22] J.-B. Grill, F. Strub, F. Altché, C. Tallec, P. Richemond, E. Buchatskaya, C. Doersch, B. Avila Pires, Z. Guo, M. Gheshlaghi Azar, *et al.*, "Bootstrap your own latent-a new approach to self-supervised learning," *Advances in neural information processing systems*, vol. 33, pp. 21271–21284, 2020.

[23] R. Aljundi, F. Babiloni, M. Elhoseiny, M. Rohrbach, and T. Tuytelaars, "Memory aware synapses: Learning what (not) to forget," in *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 139–154, 2018.

[24] S.-W. Lee, J.-H. Kim, J. Jun, J.-W. Ha, and B.-T. Zhang, "Overcoming catastrophic forgetting by incremental moment matching," *Advances in neural information processing systems*, vol. 30, 2017.

[25] J. Yoon, S. Kim, E. Yang, and S. J. Hwang, "Scalable and order-robust continual learning with additive parameter decomposition," in *International Conference on Learning Representations*, 2020.

[26] J. Serra, D. Suris, M. Miron, and A. Karatzoglou, "Overcoming catastrophic forgetting with hard attention to the task," in *International Conference on Machine Learning*, pp. 4548–4557, PMLR, 2018.

[27] L. Yang, S. Lin, J. Zhang, and D. Fan, "Grown: Grow only when necessary for continual learning," *arXiv preprint arXiv:2110.00908*, 2021.

[28] M. Riemer, I. Cases, R. Ajemian, M. Liu, I. Rish, Y. Tu, and G. Tesauro, "Learning to learn without forgetting by maximizing transfer and minimizing interference," *arXiv preprint arXiv:1810.11910*, 2018.

[29] Y. Guo, M. Liu, T. Yang, and T. Rosing, "Improved schemes for episodic memory-based lifelong learning," *Advances in Neural Information Processing Systems*, vol. 33, pp. 1023–1035, 2020.

[30] A. Chaudhry, M. Ranzato, M. Rohrbach, and M. Elhoseiny, "Efficient lifelong learning with a-gem," *arXiv preprint arXiv:1812.00420*, 2018.

[31] D. Rao, F. Visin, A. Rusu, R. Pascanu, Y. W. Teh, and R. Hadsell, "Continual unsupervised representation learning," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[32] Y. Liu, S. Agarwal, and S. Venkataraman, "Autofreeze: Automatically freezing model blocks to accelerate fine-tuning," *arXiv preprint arXiv:2102.01386*, 2021.

[33] C. He, S. Li, M. Soltanolkotabi, and S. Avestimehr, "Pipetransformer: Automated elastic pipelining for distributed training of transformers," *arXiv preprint arXiv:2102.03161*, 2021.

[34] Y. Wang, D. Sun, K. Chen, F. Lai, and M. Chowdhury, "Efficient dnn training with knowledge-guided layer freezing," *arXiv preprint arXiv:2201.06227*, 2022.

[35] G. Aguilar, Y. Ling, Y. Zhang, B. Yao, X. Fan, and C. Guo, "Knowledge distillation from internal representations," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, pp. 7350–7357, 2020.

[36] G. Yuan, Y. Li, S. Li, Z. Kong, S. Tulyakov, X. Tang, Y. Wang, and J. Ren, "Layer freezing & data sieving: Missing pieces of a generic framework for sparse training," *arXiv preprint arXiv:2209.11204*, 2022.

[37] Z. Wu, Y. Xiong, S. X. Yu, and D. Lin, "Unsupervised feature learning via non-parametric instance discrimination," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3733–3742, 2018.

[38] G. M. Van de Ven and A. S. Tolias, "Three scenarios for continual learning," *arXiv preprint arXiv:1904.07734*, 2019.

[39] A. Brock, T. Lim, J. M. Ritchie, and N. Weston, "Freezeout: Accelerate training by progressively freezing layers," *arXiv preprint arXiv:1706.04983*, 2017.