SYNCHRONIZED CONTRASTIVE PRUNING FOR EFFICIENT SELF-SUPERVISED LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Various self-supervised learning (SSL) methods have demonstrated strong capability in learning visual representations from unlabeled data. However, the current state-of-the-art (SoTA) SSL methods largely rely on heavy encoders to achieve comparable performance as the supervised learning counterpart. Despite the well-learned visual representations, the large-sized encoders impede the energy-efficient computation, especially for resource-constrained edge computing. Prior works have utilized the sparsity-induced asymmetry to enhance the contrastive learning of dense models, but the generality between asymmetric sparsity and self-supervised learning has not been fully discovered. Furthermore, transferring the supervised sparse learning to SSL is also largely under-explored. To address the research gap in prior works, this paper investigates the correlation between in-training sparsity and SSL. In particular, we propose a novel sparse SSL algorithm, embracing the benefits of contrastiveness while exploiting high sparsity during SSL training. The proposed framework is evaluated comprehensively with various granularities of sparsity, including element-wise sparsity, GPU-friendly $N{:}M$ structured fine-grained sparsity, and hardware-specific structured sparsity. Extensive experiments across multiple datasets are performed, where the proposed method shows superior performance against the SoTA sparse learning algorithms with various SSL frameworks. Furthermore, the training speedup aided by the proposed method is evaluated with an actual DNN training accelerator model.

1 INTRODUCTION

The early empirical success of deep learning was primarily driven by supervised learning with massive labeled data, e.g., ImageNet (Krizhevsky et al., 2012). To overcome the labeling bottleneck of deep learning, learning visual representations without label-intensive datasets has been widely investigated (Chen et al., 2020a; He et al., 2020; Grill et al., 2020; Zbontar et al., 2021). The recent self-supervised learning (SSL) methods have shown great success and achieved comparable performance to the supervised learning counterpart. The common property of various SSL designs is utilizing different augmentations from the original images to generate contrastiveness, which requires duplicated encoding with wide and deep models (Meng et al., 2022). The magnified training effort and extensive resource consumption make the SSL-trained encoder infeasible for on-device computing (e.g., mobile devices). The contradiction between label-free learning and extraordinary computation cost limits further applications of SSL, also necessitating efficient sparse training techniques for self-supervised learning.

For supervised learning, sparsification (a.k.a. pruning) has been widely explored, aiming to reduce computation and memory costs by removing unimportant parameters during training or fine-tuning. Conventional supervised pruning explores weight sparsity based on a pre-trained model followed by additional fine-tuning to recover the accuracy (Han et al., 2016). For self-supervised learning, recent work (Chen et al., 2021) also sparsified a pre-trained dense SSL model for the downstream tasks with element-wise pruning. In addition to the fine-grained sparsity, MCP (Pan et al., 2022) exploited the filter-wise sparsity on the MoCo-SSL (He et al., 2020) model. Both of these sparse SSL works (Chen et al., 2021; Pan et al., 2022) exploit sparsity based on the pre-trained dense model. However, compared to supervised learning, obtaining the pre-trained model via SSL requires a significant amount of additional training effort ($\sim$200 epochs vs. $\sim$1,000 epochs). Therefore, exploring post-training sparsity via fine-tuning is not an ideal solution for efficient SSL.

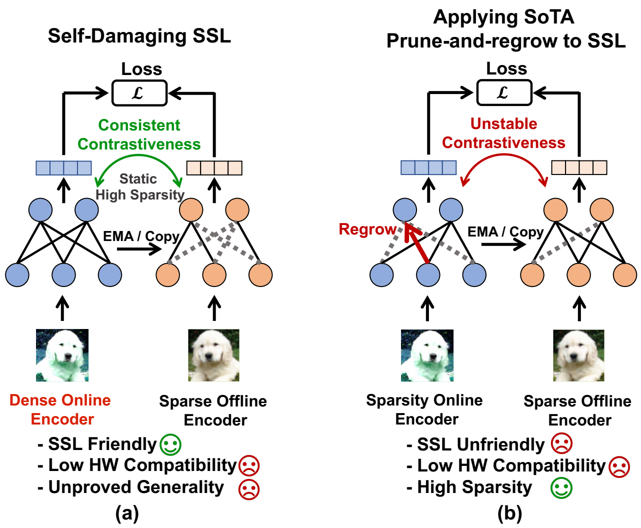

Figure 1: (a) Applying self-damaging scheme (Jiang et al., 2021) to SSL. (b) Applying prune-and-regrow scheme (Liu et al., 2021) to SSL. (c) Proposed contrastiveness-aware sparse training.

On the other hand, sparsifying the model during supervised training (Dettmers & Zettlemoyer, 2019; Evci et al., 2020) has emerged as a promising technique to elevate training efficiency while obtaining a sparse model. To accurately localize the unimportant parameters, prior works investigated various types of importance metrics, including gradient-based pruning (Dettmers & Zettlemoyer, 2019) and the "prune-regrow" scheme (Liu et al., 2021). Compared to the post-training sparsification methods, in-training sparsification for supervised training has achieved memory/computation reduction as well as speedup of the training process. However, exploiting in-training sparsity for SSL models that are trained from scratch is still largely under-explored.

In general, eliminating the unimportant parameters from the model often leads to biased performance across different classes (Hooker et al., 2019), due to the distorted memorization caused by the sparsified model architectures. As an exception, the sparsified "self-damaging" encoder in SD-CLR (Jiang et al., 2021) creates the asymmetric learners SSL, where the enhanced contrastiveness leads to improved performance with the non-salient samples. Nevertheless, such sparsity-aided contrastive learning mainly focuses on the performance enhancement of the SSL-trained dense model (i.e., SimCLR (Chen et al., 2020a)), and whether such an asymmetric learning scheme works in other SSL methods remains unclear. The expensive self-supervised learning and under-explored sparse contrastiveness inspire us to investigate the following question: *How to efficiently sparsify the model during self-supervised training with the awareness of contrastiveness?*

In this work, for the first time, we collectively investigate this question from the perspectives of both self-supervised learning and sparse training. We first discover the challenges of the sparsity-induced asymmetric SSL (Jiang et al., 2021), where the sparsity-induced "sparse-dense" asymmetric SSL is not universally applicable for various SSL schemes. Applying SoTA sparse training techniques of supervised training, such as "prune-and-regrow" (Liu et al., 2021), towards SSL is also challenging, because the pruning candidate in both encoders is frequently swapped by regrowing and the unsynchronized sparsity, leading to the oscillated encoder architecture during training. When such a technique is applied for sparse SSL training, it will further destabilize self-supervised learning. To address these challenges, we present **Sync**hronized **C**ontrastive **P**runing (SyncCP), a novel sparse training algorithm designed for self-supervised learning. To maximize the energy efficiency of SSL training, SyncCP exploits in-training sparsity in both encoders. To avoid the "architecture oscillation" caused by the "prune-and-regrow" scheme, the proposed SyncCP algorithm sparsifies SSL while synchronizing the architecture asymmetry between encoders. Unlike the self-damaging SimCLR (Jiang et al., 2021; Chen et al., 2020a), the proposed SyncCP gradually exploits high in-training sparsity with contrastive synchronization and optimally-triggered sparsification, maximizing the training efficiency without hurting the contrastiveness of SSL. Furthermore, SyncCP is compatible with various granularities of sparsity, including element-wise pruning, GPU-friendly N:M sparsity, and structured pruning designed for a custom hardware accelerator. As shown in

Figure 1, SyncCP collectively achieves high training efficiency, SSL compatibility, and outstanding hardware feasibility. We validated the proposed method against previous SoTA sparsification algorithms on CIFAR-10, CIFAR-100, and ImageNet datasets. Across various SSL frameworks, SynCP consistently achieves SoTA accuracy in all experiments.

2 Related Works

2.1 Contrastive Self-supervised Learning

Self-supervised learning recently has gained popularity due to its ability to learn visual representation without labor-intensive labeling. Specifically, pioneering research works (He et al., 2020; Chen et al., 2020a) utilize the contrastive learning scheme (Hadsell et al., 2006) that aims to group the correlated positive samples while repelling the mismatched negative samples (Oord et al., 2018). The performance of the contrastive learning-based approaches largely depends on the contrastiveness between the positive and negative samples, which requires large-sized batches to support. As indicated by SimCLR (Chen et al., 2020a), the performance of SSL is sensitive to the training batch size, and the inflated batch size elevates training cost. MoCo (He et al., 2020; Chen et al., 2020b) alleviates such issue with queue-based learning and momentum encoder, where the extensive queue-held negative samples provide proficient contrastiveness, and the slow-moving average momentum encoder derives consistent negative pairs. BYOL (Grill et al., 2020) simplifies and outperforms the prior works by only learning positive samples, while the online latent features are projected by an additional predictor q_θ:

$$\text{online prediction} = q_\theta(g_\theta(f_\theta(X))) \tag{1}$$

$$\text{offline target} = g_\xi(f_\xi(X')) \tag{2}$$

Where f and g represent the encoder and projector of online (θ) and offline (ξ) paths with augmented input X and X', respectively. The predictor q_θ generates an alternative view of the projected positive samples, and the offline momentum encoder provides consistent encoding for contrastive learning. Overall, salient and consistent contrastiveness is essential to contrastive self-supervised learning.

2.2 Sparse Training

DNN sparsification has been widely investigated under the supervised learning domain, which can be generally categorized based on the starting point of sparsification. Early works mainly focus on post-training sparsification (Han et al., 2016; Evci et al., 2020; Jayakumar et al., 2020), which removes the weights from the pre-trained model and then recovers the accuracy with subsequent fine-tuning. Other works exploit weight sparsity prior to the training process (Wang et al., 2019; Lee et al., 2018), and the resultant model is trained with the sparsified architecture.

In contrast to post-training or pre-training sparsification, exploiting sparsity during training generates the compressed model with one-time training from scratch, eliminating the requirement of a pre-trained model or extensive searching process. With the full observability of the training process, the magnitude of the gradient can be used to evaluate the model reflection with the exploited sparsity. Motivated by this, GraNet (Liu et al., 2021) utilizes the "prune-and-regrow" technique to periodically remove the unimportant non-zero weights from the sparse model and then regrow certain pruning candidates back. Given the targeted sparsity s_t and total prune ratio r_t at iteration t, unimportant weights w are removed based on the Top-K magnitude scores:

$$w^{'} = \text{TopK}(|w|, r_t) \tag{3}$$

Subsequently, the sensitive weights are re-grown back based on the reflection of gradient g^t:

$$w = w^{'} + \text{TopK}(g^t_{i!=w', r_t - s_t}) \tag{4}$$

Since the gradient g_t indicates the instant model sensitivity at iteration t, the optimal sparse model architecture can be varied between two adjacent pruning steps.

2.3 Contrastive Learning with Sparsity-induced Asymmetry

As introduced in Section 2.1, salient and consistent contrastiveness is essential for contrastive SSL, where the contrastiveness can be constructed via negative samples or the auxiliary predictors (Grill et al., 2020). Inspired by (Hooker et al., 2019), SDCLR (Jiang et al., 2021) amplifies the contrastiveness by pruning one encoder of SimCLR (Chen et al., 2020a) while keep the identical twin dense. Such "sparsity-induced asymmetry" elevates the performance of SSL with the improved performance of the dense model on the long-tailed data samples. However, SDCLR (Jiang et al., 2021) is not designed for model compression or efficiency improvements. Furthermore, the generality of such sparsity-induced asymmetry remains under-explored in other SSL frameworks.

3 CHALLENGE OF SPARSE SELF-SUPERVISED LEARNING

3.1 LIMITATIONS OF SPARSITY-INDUCED ASYMMETRY

It has been shown in SDCLR (Jiang et al., 2021) that the sparsity-induced "sparse-dense" asymmetry is beneficial to contrastive SSL. SDCLR (Jiang et al., 2021) is specifically built upon the SimCLR (Chen et al., 2020a) framework with shared encoders, where the pruned architectures have the dense twin in the mirrored contrastive encoder. However, the generality of sparsity-induced asymmetry remains unproved in other SSL methods, which motivates us to investigate the following question:

Question 1: *For contrastive self-supervised learning with non-identical encoders, will the sparsity-induced asymmetric encoders still result in elevated performance for contrastive learning?*

To answer the above question, we use MoCo-V2 (Chen et al., 2020b) and follow the procedure of SDCLR (Jiang et al., 2021) to generate a highly-sparse online encoder prior to the training process. Mathematically, we have:

$$\text{online output} = g_\theta(\mathbf{M}_\theta \cdot f_\theta(X))) \tag{5}$$

$$\text{offline output} = g_\xi(f_\xi(X')) \tag{6}$$

The online encoder mask $\mathbf{M}_\theta$ produces a sparse online encoder with initialized element-wise sparsity (Han et al., 2016) at 90%, while the offline encoder is updated by exponential moving average (EMA). Following the setup of SDCLR (Jiang et al., 2021), the sparsity is periodically updated at the beginning of each epoch. Table 1 summarizes the linear evaluation accuracy on the CIFAR-10 dataset with different static online sparsity values. As opposed to the performance of SimCLR in (Jiang et al., 2021), directly applying the high sparsity-based perturbation to MoCo-V2 (Chen et al., 2020b) is challenging, and leads to considerable performance degradation.

Table 1: Largely degraded performance of MoCo-V2 (Chen et al., 2020b) with self-damaging SSL (Jiang et al., 2021) on CIFAR-10 dataset.

ResNet-18	**Dense Model Acc. = 92.09%**			
Encoder	**Online**	**Momentum**	**Online**	**Momentum**
Fixed Sparsity	90%	0%	50%	0%
Linear Eval. Acc (%)	88.72 (-3.41%)	87.68 (-4.31%)	92.10 (+0.01%)	92.07 (-0.02%)

Summarizing these empirical results, our main observation is:

Observation 1: *Compared to the online encoder, the EMA-updated momentum encoder has the delayed architecture, which makes it unqualified to be the "competitor" as SDCLR (Jiang et al., 2021). The directly-applied high sparsity overshoots the asymmetric learning, leading to degraded self-supervised learning.*

3.2 FREQUENT ARCHITECTURE CHANGING HINDERS SELF-SUPERVISED LEARNING

As depicted in Eq. 4, the "prune-and-regrow" scheme such as GraNet (Liu et al., 2021) uses instant gradient magnitude to indicate the model sensitivity after magnitude pruning, removing the unimportant and insensitive weights while gradually achieving high sparsity. **Observation 1** demonstrates the incompatibility of the directly-applied high sparsity in SSL, then the following question raises:

Question 2: *If we apply the gradually-increased sparsity for both encoders, will the "prune-and-regrow" scheme also be feasible for self-supervised learning?*

Table 2: Sparse training with "prune-and-regrow" scheme on BYOL (Grill et al., 2020).

BYOL (Grill et al., 2020)	**CIFAR-10 Acc (%)**	
ResNet-18	**Dense Model Acc. = 92.42%**	
Online Encoder Sparsity	0%→80%	0%→90%
Online Linear Eval. Acc (%)	91.20±0.02	90.13±0.06
Momentum Encoder Sparsity	0%→50%	0%→60%
Momentum Linear Eval Acc. (%)	91.31±0.07	90.09±0.04

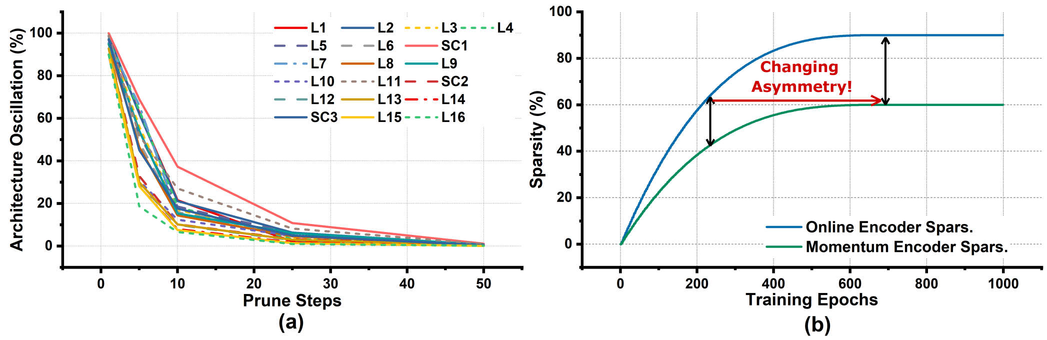

Figure 2: (a) Layer-wise oscillation at different steps of pruning. "SC" stands for the shortcut connection of ResNet-18 model. (b) Gradually-increased sparsity of GraNet (Liu et al., 2021) leads to inconsistent asymmetry.

To address **Question 2**, we use the SoTA GraNet (Liu et al., 2021) as the example algorithm to exploit in-training sparsity on both encoders of BYOL (Grill et al., 2020), where the regrowing process is only applied to the online encoder. Starting with the dense models, we gradually prune the online and offline encoders to 90% and 60% sparsity in an element-wise fashion with periodically-updated sparsity. For sparse SSL training, the results of such gently-increased sparsity scheme reported in Table 2 outperforms those by (Jiang et al., 2021) (Table 1) by a significant margin. However, the state-of-the-art supervised pruning algorithm still incurs 2.3% linear evaluation accuracy degradation with SSL on the CIFAR-10 dataset.

Compared to the self-damaging SSL with fixed sparsity (Jiang et al., 2021), the "prune-and-regrow" method keeps swapping the pruning candidates to minimize the model sensitivity, oscillating the encoder architecture during training. We quantifying such architecture oscillation by XORing the masks generated from magnitude pruning $\mathbf{M}_{MP}$ and gradient-based regrow $\mathbf{M}_g$:

$$\mathbf{G}_{cor} = \mathbf{M}_{MP} \oplus \mathbf{M}_g \in \{0, 1\} \tag{7}$$

Under the same sparsity ratio, the number of "1"s in $\mathbf{G}_{cor}$ indicates the amount of architecture oscillation caused by the gradient-based regrow. During the early stage of training, almost all the magnitude pruning candidates are replaced by the regrowing process, as shown in Figure 2(a). The high degree of architecture oscillation implies drastic changes in the sparse model architecture. In the meantime, gradually sparsifying two encoders with different target sparsity further destroys the consistency of self-supervised learning, as shown in Figure 2(b). As a result, we have the following observation for **Question 2**:

Observation 2: *Sparsifying the model with frequently changing architecture hinders the contrastiveness and consistency of self-supervised learning and leads to degraded encoder performance.*

As shown in **Observation 1** and **Observation 2**, high sparsity-induced asymmetry is not directly-applicable to sparse SSL, while the consistency requirements of SSL negates the plausibility of gradual sparsification. The dilemma between self-supervised learning and sparse training derives the following challenge:

How can we efficiently sparsify the model during self-supervised training while maximizing the benefits of the sparsity-induced asymmetry?

4 METHOD

To address the above challenge, we propose **Sync**hronized **C**ontrastive **P**runing (SyncCP), which successfully alleviates the contradiction between the needs of high sparsity and the requirements of consistency in self-supervised learning.

4.1 SYNCHRONIZED SPARSIFICATION (SYNCS)

The rationale behind the sparsity-induced asymmetric SSL is that the perturbation generated by the pruned encoder elevates the difference between contrastive features. As indicated by **Observation 1** and Table 1, the high sparsity-induced asymmetry is not universally applicable, but the SSL can be rewarded from the asymmetry incurred by lower sparsity (e.g., 50%), where the SSL-trained sparse and dense encoders exhibit negligible accuracy degradation compared to the baseline. Motivated by this, we propose the *Synchronized Sparsification (SyncS)* technique to exploit sparsity in both contrastive encoders. Given the online and offline (momentum) encoder θ and ξ, the in-training sparsification can be expressed as:

$$\text{online output} = g_\theta(\mathbf{M}_\theta \cdot f_\theta(X))) \tag{8}$$

$$\text{offline output} = g_\xi(\mathbf{M}_\xi \cdot f_\xi(X')) \tag{9}$$

Where $\mathbf{M}_\theta$ and $\mathbf{M}_\xi$ represent the online and offline (momentum) sparse masks with sparsity s_θ and s_ξ. The proposed SyncS scheme gradually exploits the sparsity in both encoders while maintaining a consistent sparsity gap Δ_s between them during SSL training. At each pruning step t, we have:

$$s_\theta^t = s_\theta^f + (s_\theta^i - s_\theta^f)(1 - \frac{t - t_0}{n\Delta t})^3 \tag{10}$$

$$s_\xi^t = s_\xi^f + (s_\xi^i - s_\xi^f)(1 - \frac{t - t_0}{n\Delta t})^3 \tag{11}$$

$$\text{s.t } |s_\theta^t - s_\xi^t| = \Delta_s, \text{ for } t \in \{t_0, t_0 + \Delta t, ..., t_0 + n\Delta t\} \tag{12}$$

The synchronized sparsification with the constraints of Δ_s prevents the exceeding asymmetry between contrastive encoders while minimizing the distortion caused by the changing sparsity. In practice, Δ_s is treated as a tunable hyperparameter initialized by *Erdos Renyi Kernel* (ERK) (Evci et al., 2020), and impacts the final sparsity of both online and offline encoders. To guarantee the consistency of the contrastive sparsity, both s_θ and s_ξ are initialized with respect to Δ_s.

4.2 CONTRASTIVE SPARSIFICATION INDICATOR (CSI)

Achieving high sparsity requires gentle sparsification, but as presented in **Observation 2**, the inconsistent architecture difference deteriorates the contrastiveness of SSL. On the other hand, the popular EMA-based update (He et al., 2020) allows the momentum encoder to generate consistent latent representation, but the lagged architecture makes the momentum encoder become an unqualified "competitor" to the online encoder, which violates the findings of (Jiang et al., 2021). To address such conflict, we propose the *Contrastive Sparsification Indicator* (CSI), a simple-yet-effective method that automatically selects the starting point of sparsification based on the learning progress of SSL.

During the self-supervised training, CSI first generates the pseudo pruning decisions of both encoders based on element-wise magnitude pruning with respect to the target sparsity s_θ^f and s_ξ^f:

$$\mathbf{M}_\theta^* = \mathbb{1}\{|w_\theta| \in \text{TopK}(|w_\theta|, s_\theta^f)\} \tag{13}$$

$$\mathbf{M}_\xi^* = \mathbb{1}\{|w_\xi| \in \text{TopK}(|w_\xi|, s_\xi^f)\} \tag{14}$$

Where $\mathbb{1}$ represents the indicator function, and the resultant pseudo masks of $\mathbf{M}_\theta^*$ and $\mathbf{M}_\xi^*$ will not be applied to the weights. Subsequently, CSI XORs the pseudo pruning masks to generate G (Eq. 15), and the number of "1"s in $\mathbf{G}$ is equivalent to the architecture inconsistency $\mathbf{I}$ (Eq. 16). Instead of using cosine similarity, the bit-wise XOR can be easily implemented on hardware to quantify the architecture difference during training.

$$\mathbf{G} = \mathbf{M}_\theta^* \oplus \mathbf{M}_\xi^* \tag{15}$$

$$\mathbf{I} = 1 - \frac{\sum \mathbb{1}\{\mathbf{G} = 0\}}{|\mathbf{G}|} \tag{16}$$

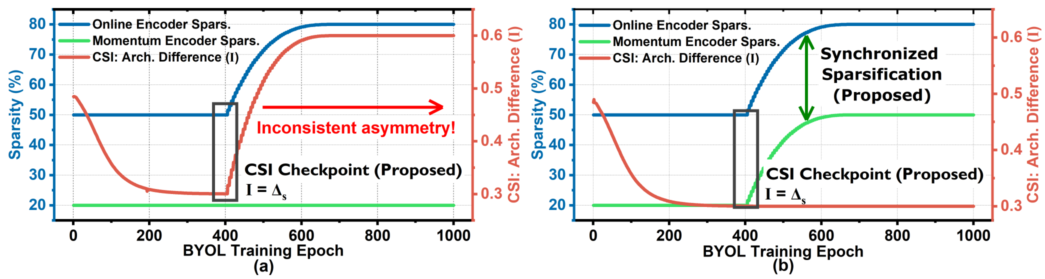

Figure 3: Sparse BYOL training process (a) without SyncS and (b) with SyncS.

Table 3: Performance comparison of BYOL on CIFAR-10 dataset with/without SyncS.

Method	Online Encoder Spars.	Momentum Encoder Spars.	Online Linear Eval. Acc. (%)
CSI + SyncS	50%→80%	20%→50%	**92.24%**
CSI Only	50%→80%	20%→20% (Fixed)	91.64%

Given the sparsity gap Δ_s defined by SyncS, CSI activates the sparsity increment when $\mathbf{I}$ equals to Δ_s, and this moment is defined as the *CSI checkpoint*. In other words, when the architecture difference between online and offline encoders is mainly caused by the sparsity difference, it is the optimal moment to start exploiting the in-training sparsity with the gradually-increased sparsity. With the ability to automatically select the starting point of sparsification, the proposed CSI method automatically sparsifies the model with the full awareness of the SSL process. For the SSL framework with shared encoder (Zbontar et al., 2021), the architecture inconsistency $\mathbf{I}$ is computed based on the sparse architecture of two consecutive epochs, and the sparsification process is activated when $\mathbf{I}$ is less than a pre-defined threshold τ (e.g., $\tau = 0.9$).

Figure 3 shows the sparsification scheme with and without SyncS. As summarized in Table 3, holding the sparse momentum architecture after the CSI checkpoint interrupts the consistency between the online and momentum encoders. Although the momentum encoder retains the low sparsity at 20%, the absence of consistent asymmetry from synchronized contrastive pruning (SCP) causes the degraded model performance.

On top of the proposed SyncS and CSI schemes, we adopt the prune-and-regrow scheme (Liu et al., 2021) with modifications to exploit sparsity during SSL training. To further alleviate the contrastiveness oscillation caused by changing sparsity, we slowly average the gradient magnitude by exponential moving average (EMA) with gentle momentum, instead of using the instant score. The detailed pseudo code of the proposed algorithm is summarized in Appendix A.

5 Experimental Results

In this section, we validate the proposed sparse training scheme and compare it with the current SoTA sparse training schemes. Unlike the work by (Jiang et al., 2021), the proposed scheme exploits in-training sparsity in both contrastive paths (encoders) and aims to achieve energy-efficient self-supervised learning. The proposed method is applied to multiple mainstream SSL frameworks, including EMA-based methods (Chen et al., 2020b; Grill et al., 2020) and SSL with shared encoder (Zbontar et al., 2021). The linear evaluation accuracy and training cost reduction are reported for multiple datasets, including CIFAR-10, CIFAR-100, and ImageNet-2012. Furthermore, this work exploits in-training sparsity with various sparsity granularities, including element-wise sparsity, N:M sparsity (Zhou et al., 2020), and structural sparsity for a custom hardware accelerator.

CIFAR-10 and CIFAR-100 Table 4 summarizes the linear evaluation accuracy of the proposed method on CIFAR-10 and CIFAR-100 datasets with element-wise sparsity. We use ResNet-18 ($1\times$) as the backbone and train the model from scratch by 1,000 epochs. Following the typical high sparsity results reported with supervised learning, we report the model performance with 80% and 90%

Table 4: Linear evaluation comparison on CIFAR-10/100 datasets with element-wise sparsity.

Dataset		**CIFAR-10 Acc (%)**			**CIFAR-100 Acc (%)**		
Encoder		**ResNet-18 (1×)**			**ResNet-18 (1×)**		
Element-wise Sparsity		0%	80%	90%	0%	80%	90%
MoCo-V2 (Chen et al., 2020b)	**This work**	92.09	**91.77±0.08**	**91.31±0.04**	67.72	67.56±0.04	66.78±0.07
	GraNet-MoCo (Liu et al., 2021)	92.09	90.66±0.07	90.05±0.08	67.72	67.17±0.05	64.92±0.06
	SD-MoCo (Jiang et al., 2021)	92.09	90.26±0.05	87.68±0.06	67.72	65.04±0.04	61.33±0.05
BYOL (Grill et al., 2020)	**This work**	92.42	**92.26±0.06**	**92.03±0.05**	68.80	**68.69±0.06**	**67.73±0.04**
	GraNet-BYOL (Liu et al., 2021)	92.42	91.20±0.02	90.13±0.03	68.80	67.17±0.05	65.85±0.08
	SD-BYOL (Jiang et al., 2021)	92.42	90.33±0.07	87.38±0.04	68.80	66.13±0.08	62.20±0.10
Barlow Twins (Zbontar et al., 2021)	**This work**	91.74	**91.67±0.09**	**90.84±0.07**	68.62	**68.75±0.13**	**68.48±0.12**
	GraNet-Barlow (Liu et al., 2021)	91.74	91.23±0.03	90.44±0.12	68.62	68.40±0.10	68.15±0.14
	SD-Barlow (Jiang et al., 2021)	91.74	90.09±0.03	88.41±0.07	68.62	66.42±0.07	61.77±0.04

Table 5: Linear evaluation accuracy comparison on CIFAR-10/100 datasets with N:M structured-fine-grained sparsity.

Datasets	**CIFAR-10 Acc (%)**		**CIFAR-100 Acc (%)**	
Encoder	**ResNet-18 (1×)**		**ResNet-18 (1×)**	
N:M **Sparse Pattern**	2:4	1:4	2:4	1:4
MoCo-V2 (Chen et al., 2020b)	91.99±0.07	91.53±0.04	67.58±0.05	67.11±0.05
BYOL (Grill et al., 2020)	92.61±0.05	91.83±0.02	68.69±0.02	68.09±0.07
Barlow Twins (Zbontar et al., 2021)	91.68±0.04	90.97±0.03	68.26±0.07	68.19±0.06
Inference time reduction (s)	1.40×	2.08×	1.40×	2.08×

target sparsity. To sparsify both encoders during SSL training, we initialize the sparsity of online and offline (momentum) encoders as 30% and 0%, where the Δ_s is set to 30%. The initialized sparse encoders reduce the overall memory footprint throughout the entire training process. We rigorously transfer the SoTA GraNet Liu et al. (2021) to SSL based on its open-sourced implementation, the proposed method outperforms GraNet-SSL with 1.26% and 1.86% accuracy improvements on CIFAR-10 and CIFAR-100 datasets, respectively. In all experiments, we report the average accuracy with its variation in 3 runs.

In addition to element-wise sparsity, the recent Nvidia Ampere architecture is equipped with the Sparse Tensor Cores to accelerate the inference computation on GPU with N:M structured fine-grained sparsity (Zhou et al., 2020), where the N dense elements remain within an M-sized group. Powered by the open-sourced Nvidia-ASP library, SyncCP sparsifies BYOL training (Grill et al., 2020) by targeting 100% N:M sparse groups in online encoders. Starting from scratch, the percentage of the N:M sparse groups is initialized as 30% and 0% in online and momentum encoders with Δ_s=30%. After the CSI checkpoint, the percentage of N:M groups gradually increases. Appendix A describes the detailed pruning algorithm of N:M sparsification. Table 5 summarizes linear evaluation accuracy and inference time reduction on the CIFAR-10 and CIFAR-100 datasets. The resultant model achieves up to 2.08× inference acceleration with minimum accuracy degradation. The inference time is measured on an Nvidia 3090 GPU with FP32 data precision.

ImageNet-2012 Since the BYOL (Grill et al., 2020) learning scheme achieves the best performance with CIFAR datasets, we further evaluate the proposed method with ResNet-50 on ImageNet based on the BYOL framework (Grill et al., 2020). Following the typical high sparsity results re-

Table 6: ImageNet-2012 accuracy and training cost comparison with SoTA works on ResNet-50 with BYOL (Grill et al., 2020).

ImageNet		Top-1 Accuracy (%)	FLOPS (Training)	Top-1 Accuracy (%)	FLOPS (Training)
Dense Baseline		66.16	5.29e+18 (1×)	66.16	5.29e+18 (1×)
Element-wise Sparsity		80%		90%	
BYOL (Grill et al., 2020)	**This work**	**64.89**	**0.64×**	**63.76**	**0.58×**
	GraNet-BYOL (Liu et al., 2021)	63.65	0.59×	62.55	0.51×
	SD-BYOL (Jiang et al., 2021)	61.64	0.68×	59.04	0.63×

ported in Table 4, we report the model performance with 80% and 90% element-wise sparsity. The data augmentation setup is adopted from the open-sourced library (Costa et al., 2022). Starting from scratch, the model is trained by 200 epochs, where both online and momentum encoders are initialized by ERK with Δ_s = 30%. While we believe a more fine-grained hyperparameter tuning and extended training efforts could lead to better accuracy, we choose the above scheme for simplicity and reproducibility. Table 6 shows the comparison of linear evaluation accuracy on ImageNet-2012 dataset. Compared to the self-damaging scheme (Jiang et al., 2021) and GraNet (Liu et al., 2021), the proposed algorithm achieves the same highly-sparse network with 4.72% and 1.21% Top-1 inference accuracy improvements, respectively. GraNet exploits in-training sparsity throughout the entire training process, but the inconsistent contrastiveness hampers the model performance. On the other hand, the dense encoder limits the efficiency of the self-damaging scheme (Jiang et al., 2021) scheme, and the static high sparsity degrades the model performance.

Table 7: Hardware training acceleration of the proposed structured sparse SSL training algorithm.

BYOL+ResNet-18		Top-1 Accuracy (%)	Training Speed-up	Top-1 Accuracy (%)	Training Speed-up
Dense Baseline		92.42	1×	92.42	1×
Target Structured Sparsity		80%		90%	
BYOL (Grill et al., 2020)	**This work**	92.16	1.74×	91.77	1.91×

Hardware-based Structured Pruning The hardware practicality of element-wise sparsification is often limited by the irregularity of fine-grained sparsity and index requirement. To that end, we employ structured sparsity based on group-wise EMA scores towards achieving actual hardware training acceleration. The encoders are initialized by ERK with 30% and 0% sparse groups while keeping Δ_s = 30%. The structured sparsity starts to gradually increase after the CSI checkpoint. We adopt the training accelerator specifications from (Venkataramanaiah et al., 2022) and choose K_l (# of output channels) $\times$ C_l (# of input channels) = 8×8 as the sparse group size. Table 7 evalautes the training speedup of BYOL (Grill et al., 2020) aided by the structured sparse training. The proposed algorithm achieves up to 1.91× training acceleration with minimal accuracy degradation.

6 Conclusion

In this paper, we propose a novel sparse training algorithm designed for self-supervised learning (SSL). As one of the first studies in this area, we first point out the imperfections of the sparsity-induced asymmetric self-supervised learning, as well as the incompatibility of the supervised sparse training algorithm in SSL. Based on the well-knit conclusions, we propose a contrastiveness-aware sparse training algorithm, consisting of synchronized contrastive pruning (SCP) and contrastive sparsification indicator (CSI). The proposed method outperforms the SoTA sparse training algorithm on both CIFAR and ImageNet-2012 datasets with various mainstream SSL frameworks. We also demonstrate the actual training and inference hardware acceleration with structured sparsity and N:M structured fine-grained pattern.

REFERENCES

Tianlong Chen, Jonathan Frankle, Shiyu Chang, Sijia Liu, Yang Zhang, Michael Carbin, and Zhangyang Wang. The Lottery Tickets Hypothesis for Supervised and Self-supervised Pre-training in Computer Vision Models. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 16306–16316, 2021.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A Simple Framework for Contrastive Learning of Visual Representations. In *International Conference on Machine Learning (ICML)*, pp. 1597–1607. PMLR, 2020a.

Xinlei Chen, Haoqi Fan, Ross Girshick, and Kaiming He. Improved Baselines with Momentum Contrastive learning. *arXiv preprint arXiv:2003.04297*, 2020b.

Victor Guilherme Turrisi Da Costa, Enrico Fini, Moin Nabi, Nicu Sebe, and Elisa Ricci. solo-learn: A Library of Self-supervised Methods for Visual Representation Learning. *Journal of Machine Learning Research*, 23(56):1–6, 2022.

Tim Dettmers and Luke Zettlemoyer. Sparse Networks From Scratch: Faster Training without losing performance. *arXiv preprint arXiv:1907.04840*, 2019.

Utku Evci, Trevor Gale, Jacob Menick, Pablo Samuel Castro, and Erich Elsen. Rigging the lottery: Making all tickets winners. In *International Conference on Machine Learning (ICML)*, 2020.

Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Guo, Mohammad Gheshlaghi Azar, et al. Bootstrap your own latent-a new approach to self-supervised learning. *Advances in Neural Information Processing Systems (NeurIPS)*, 33:21271–21284, 2020.

Raia Hadsell, Sumit Chopra, and Yann LeCun. Dimensionality reduction by learning an invariant mapping. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1735–1742, 2006.

Song Han, Huizi Mao, and William J Dally. Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding. *International Conference on Learning Representations (ICLR)*, 2016.

Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 9729–9738, 2020.

Sara Hooker, Aaron Courville, Gregory Clark, Yann Dauphin, and Andrea Frome. What do Compressed Deep Neural Networks Forget? *arXiv preprint arXiv:1911.05248*, 2019.

Siddhant Jayakumar, Razvan Pascanu, Jack Rae, Simon Osindero, and Erich Elsen. Top-kast: Top-k always sparse training. *Advances in Neural Information Processing Systems (NeurIPS)*, 33: 20744–20754, 2020.

Ziyu Jiang, Tianlong Chen, Bobak J Mortazavi, and Zhangyang Wang. Self-damaging Contrastive Learning. In *International Conference on Machine Learning (ICML)*, pp. 4927–4939, 2021.

Alexander Kolesnikov, Xiaohua Zhai, and Lucas Beyer. Revisiting Self-supervised Visual Representation Learning. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1920–1929, 2019.

Simon Kornblith, Jonathon Shlens, and Quoc V Le. Do Better ImageNet Models Transfer Better? In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2661–2671, 2019.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. ImageNet Classification with Deep Convolutional Neural Networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2012.

Namhoon Lee, Thalaiyasingam Ajanthan, and Philip Torr. SNIP: Single-shot Network Pruning based on Connection Sensitivity. In *International Conference on Learning Representations (ICLR)*, 2018.

Shiwei Liu, Tianlong Chen, Xiaohan Chen, Zahra Atashgahi, Lu Yin, Huanyu Kou, Li Shen, Mykola Pechenizkiy, Zhangyang Wang, and Decebal Constantin Mocanu. Sparse Training via Boosting Pruning Plasticity with Neuroregeneration. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

Jian Meng, Li Yang, Jinwoo Shin, Deliang Fan, and Jae-sun Seo. Contrastive Dual Gating: Learning Sparse Features With Contrastive Learning. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 12257–12265, 2022.

Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation Learning with Contrastive Predictive Coding. *arXiv preprint arXiv:1807.03748*, 2018.

Siyuan Pan, Yiming Qin, Tingyao Li, Xiaoshuang Li, and Liang Hou. Momentum Contrastive Pruning. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2647–2656, 2022.

Shreyas K. Venkataramanaiah, Jian Meng, Han-Sok Suh, Injune Yeo, Jyotishman Saikia, Sai Kiran Cherupally, Yichi Zhang, Zhiru Zhang, and Jae sun Seo. A 28nm 8-bit Floating-Point Tensor Core based CNN Training Processor with Dynamic Activation/Weight Sparsification. In *IEEE European Solid-State Circuits Conference (ESSCIRC)*, 2022.

Chaoqi Wang, Guodong Zhang, and Roger Grosse. Picking Winning Tickets Before Training by Preserving Gradient Flow. In *International Conference on Learning Representations (ICLR)*, 2019.

Yang You, Igor Gitman, and Boris Ginsburg. Large Batch Training of Convolutional Networks. *arXiv preprint arXiv:1708.03888*, 2017.

Jure Zbontar, Li Jing, Ishan Misra, Yann LeCun, and Stéphane Deny. Barlow Twins: Self-supervised Learning via Redundancy Reduction. In *International Conference on Machine Learning (ICML)*, pp. 12310–12320, 2021.

Aojun Zhou, Yukun Ma, Junnan Zhu, Jianbo Liu, Zhijie Zhang, Kun Yuan, Wenxiu Sun, and Hongsheng Li. Learning N:M Fine-grained Structured Sparse Neural Networks From Scratch. In *International Conference on Learning Representations (ICLR)*, 2020.

A APPENDIX A

A.1 PSEUDO CODE OF SYNCCP WITH ELEMENT-WISE SPARSITY

Algorithm 1: Synchornized Contrastive Pruning (SyncCP)

```
Initialize Sparse online encoder f_θ, Sparse offline encoder f_ξ, EMA updater, Momentum γ,
SyncS density gap Δ_s, CSI threshold τ (Default=Δ_s).
Initial sparsity s_θ^0, s_ξ^0, such that |s_θ^* − s_ξ^*| = Δ_s
Target sparsity s_θ^*, s_ξ^*, such that |s_θ^* − s_ξ^*| = Δ_s
Initial mask M_θ^0, M_ξ^0.
Pruner udpate frequency Φ
while t < Total Iterations do
    Draw augmented data (X, X');
    Forward pass: online encoding = f_θ(M_θ · θ, X) ;
    Forward pass: offline encoding = f_ξ(M_ξ · ξ, X');
    Update Exponential Moving Average (EMA) gradient score based on Eq. 17 ;
    if End Epoch then
        Get pseudo masks M_θ^* and M_ξ^* based on magnitude pruning;
        Compute layer-wise G and I based on Eq. 15 and Eq. 16;
        if I = Δ_s then
            Prune=True
        end
    end
    if t % Φ = 0 then
        if Prune=True then
            Update sparsity s_θ^t, s_ξ^t based on Eq. 10 and Eq. 11;
            Maintain the SyncS constraint Δ_s;
            Inside f_θ and f_ξ, prune s_θ^t, and s_ξ^t elements with least magnitude score;
            Prune extra r_θ^t elements of the unpruned elements, then regrow r_θ elements with
              hights EMA-gradient score;
            Update M_θ^t, M_ξ^t based on Eq. 3 and Eq. 4;
        else
    end
end
```

A.2 EMA-BASED PRUNE AND REGROW

As aforementioned, the findings of **Observation 2** implies the incompatibility of the instant gradient and magnitude score. Together with the proposed SyncS and CSI methods, weight importance is measured by the magnitude score, while the sensitivity of the model is quantified by the gently averaged gradient magnitude with EMA:

$$\bar{g}^t = \gamma \times \bar{g}^{t-1} + (1-\gamma) \times |g|^t \tag{17}$$

Table 8 summarizes the linear evaluation accuracy of ResNet-18 trained by BYOL (Grill et al., 2020). We initialize s_θ^0 and s_ξ^0 as 40% and 10%, where the Δ_s is set to 30%, the EMA momentum is set to 0.1 for gentle gradient score averaging.

Metric	Prune	Regrow	EMA	Online Linear Eval. Acc. (%)
This work	✓	✓	✓	**91.88**
Prune-and-regrow	✓	✓	✗	91.52
Magnitude Pruning	✓	✗	✗	90.99

Table 8: Peformance comparison between different sparsification metrics.

A.3 PSEUDO CODE OF SYNCCP WITH $N{:}M$ SPARSITY

Algorithm 2: Synchornized Contrastive Pruning (SyncCP) with $N{:}M$ Sparsity

Initialize Sparse online encoder f_θ, Sparse offline encoder f_ξ, EMA updater, Momentum γ,
SyncS density gap Δ_s, **CSI** threshold τ (Default=Δ_s).
Group size M, Number of dense element per group N.
Initial percentage p_θ^0 of $N{:}M$ groups in f_θ, Initial percentage p_ξ^0 of $N{:}M$ groups in f_ξ;
Such that $|p_\theta^0 - p_\xi^0| = \Delta_s$;
Target percentage $p_\theta^* = 100\%$, $p_\xi^* = p_\theta^* - \Delta_s$;
Initial mask $\mathbf{M}_\theta^0$, $\mathbf{M}_\xi^0$.
Pruner udpate frequency Φ.
while $t <$ *Total Iterations* **do**
    Draw augmented data (X, X');
    Forward pass: online encoding $= f_\theta(\mathbf{M}_\theta \cdot \theta, X)$;
    Forward pass: offline encoding $= f_\xi(\mathbf{M}_\xi \cdot \xi, X')$;
    Update Exponential Moving Average (**EMA**) weight gradient score based on Eq. 17 ;
    if *End Epoch* **then**
        Get pseudo masks $\mathbf{M}_\theta^*$ and $\mathbf{M}_\xi^*$ based on magnitude pruning;
        Compute layer-wise $\mathbf{G}$ and $\mathbf{I}$ based on Eq. 15 and Eq. 16;
        if $\mathbf{I} = \Delta_s$ **then**
            Prune=True
        end
    end
    if t % $\Phi = 0$ **then**
        if ***Prune***=*True* **then**
            Update sparsity p_θ^t, p_ξ^t based on Eq. 10 and Eq. 11;
            Maintain the **SyncS** constraint $p_\theta^t, p_\xi^t = p_\theta^t - \Delta_s$;
            Inside f_θ and f_ξ, pick p_θ^t, and p_ξ^t M-sized groups with least sum of magnitude score;
            Inside each group, prune the N-M elements with smallest magnitude score;
            Update $\mathbf{M}_\theta^t$, $\mathbf{M}_\xi^t$ based on Figure 4;
        else
    end
end

Input:

- Threshold value `thre` calculated based on p and all the group-wise weight weight magnitude scores.

Output: Binary pruning mask.

```
ngroup = weight.numel()/M
# reshape the weight score of current layer
weight = weight.abs().permute(0,2,3,1).reshape(ngroup, M)

# sort the least significant elements inside each group
index = torch.argsort(weight, dim=1)[:, :int(M-N)]
# initialize mask
w_b = torch.ones(weight.shape)
# fine-grained sparsification for all groups
w_b = w_b.scatter_(dim=1, index=index, value=0)

# pick the least significant groups based on p
gsum = weight.sum(dim=1)
gindex = gsum.gt(thre)

# recover the significant groups
w_b[index, :] = 1

# reshape the mask back as the size of weight
w_b = w_b.reshape(weight.shape)
```

(a)

Input:

- Threshold value `gthre` calculated based on p and all the group-wise EMA weight **gradient** scores.
- Sparse weight mask `m` with sparsity = `r` + `s`

Output: Binary weight mask.

```
# total number of groups
ngroup = gema.numel()/M
# reshape the ema gradient score of current layer
gema = gema.permute(0,2,3,1).reshape(ngroup, M)

# only regrow the weight groups that already sparse
msum = torch.sum(m, dim=1)
sidx = msum.eq(self.N).float()

# sum the gradient of the elements inside the sparsified groups
gradgrp = gradgrp*sidx[:, None]
gsum = torch.sum(gradgrp, dim=1)

# sort the response
y, idx = torch.sort(gsum.flatten(), descending=True)
# regrow the group
m[idx[:total_regrowth]] = 1.0

# reshape the binary mask back (permutation is required in practice)
m = m.reshape(weight.shape)
```

(b)

Figure 4: Group-wise (a) prune and (b) regrow algorithm based on EMA gradient score. SyncCP sparsifies $M - N$ elements inside each group, while keep p^t $N{:}M$ groups inside f.

A.4 Pseudo Code of SyncCP with structured sparsity

Algorithm 3: Synchornized Contrastive Pruning (SyncCP) with Structured Sparsity

Initialize Sparse online encoder f_θ, Sparse offline encoder f_ξ, EMA updater, Momentum γ,
SyncS density gap Δ_s, **CSI** threshold τ (Default=Δ_s).
Group size K_l (# of output channels) $\times$ C_l (# of input channels) = $g \times g$.
Initial percentage p_θ^0 of sparse groups in f_θ, Initial percentage p_ξ^0 of sparse groups in f_ξ;
Such that $|p_\theta^* - p_\xi^*| = \Delta_s$;
Initial structured sparsity s_θ^0, s_ξ^0, such that $|s_\theta^* - s_\xi^*| = \Delta_s$
Target structured sparsity s_θ^*, s_ξ^*, such that $|s_\theta^* - s_\xi^*| = \Delta_s$
Initial mask $\mathbf{M}_\theta^0$, $\mathbf{M}_\xi^0$.
Pruner udpate frequency Φ.
while $t <$ *Total Iterations* **do**
    Draw augmented data (X, X');
    Forward pass: online encoding $= f_\theta(\mathbf{M}_\theta \cdot \theta, X)$;
    Forward pass: offline encoding $= f_\xi(\mathbf{M}_\xi \cdot \xi, X')$;
    Update Exponential Moving Average (**EMA**) gradient score based on Eq. 17;
    if *End Epoch* **then**
        Get pseudo masks $\mathbf{M}_\theta^*$ and $\mathbf{M}_\xi^*$ based on magnitude pruning;
        Compute layer-wise $\mathbf{G}$ and $\mathbf{I}$ based on Eq. 15 and Eq. 16;
        if $\mathbf{I} = \Delta_s$ **then**
            Prune=True
        end
    end
    if t % $\Phi = 0$ **then**
        if ***Prune***=*True* **then**
            Update structured sparsity s_θ^t, s_ξ^t based on Eq. 10 and Eq. 11;
            Maintain the **SyncS** constraint s_θ^t, $s_\xi^t = s_\theta^t - \Delta_s$;
            Inside f_θ and f_ξ, pick s_θ^t, and s_ξ^t groups with least sum of magnitude score;
            Outside the sparsified groups of f_θ, prune r_θ^t more groups with least sum of magnitude score;
            Among the sparsified groups f_θ, regrow the r_θ^t groups back with highest sum of EMA gradient score;
            Update $\mathbf{M}_\theta^t$, $\mathbf{M}_\xi^t$;
        end
    end
end

B Detailed Experimental Setup of SyncCP

B.1 Linear Evaluation Protocol

As in (Kolesnikov et al., 2019; Kornblith et al., 2019; Chen et al., 2020a), we use the standard linear evaluation protocol on CIFAR-10/100 and ImageNet-2012 datasets, which training a linear classifier on top of the frozen SSL-trained encoder. During linear evaluation, we apply spatial augmentation and random flips. The linear classifier is optimized by SGD with cross-entropy loss.

B.2 CIFAR-10/100 experiments

The training hyper-parameters of the compared individual sparse training works are same for CIFAR-10 and CIFAR-100. We provide the detailed training setup of different self-supervised learning frameworks as follow:

MoCo-V2 The ResNet-18 ($\times$) encoder is trained by MoCo-V2 (Chen et al., 2020b) from scratch by 1,000 epochs with SGD optimizer and 256 batch size. The learning rate is set to 0.3 with Cosine learning rate decay and 10 epochs warmup. The detailed data augmentation is summarized in Table 9.

Parameter	X	X'
Random crop size	32×32	32×32
Horizontal flip probability	0.5	0.5
Color jitter probability	0.8	0.8
Brightness adjustment probability	0.4	0.4
Contrast adjustment probability	0.4	0.4
Saturation adjustment probability	0.2	0.2
Hue adjustment probability	0.1	0.1
Gaussian blurring probability	0.0	0.0
Solarization probability	0.0	0.0

Table 9: Detailed image augmentation settings for MoCo-V2 (Chen et al., 2020b) on CIFAR-10/100.

BYOL The ResNet-18 ($\times$) encoder is trained by BYOL (Grill et al., 2020) from scratch by 1,000 epochs with LARS-SGD optimizer (You et al., 2017). The predictor is constructed with 4096 hidden features and 256 output dimension. We use 256 batch size along with 1.0 learning rate. The Cosine learning rate scheduler is used with 10 epochs warmup training. The detailed data augmentation is summarized in Table 10.

Parameter	X	X'
Random crop size	32×32	32×32
Horizontal flip probability	0.5	0.5
Color jitter probability	0.8	0.8
Brightness adjustment probability	0.4	0.4
Contrast adjustment probability	0.4	0.4
Saturation adjustment probability	0.2	0.2
Hue adjustment probability	0.1	0.1
Gaussian blurring probability	0.0	0.0
Solarization probability	0.0	0.2

Table 10: Detailed image augmentation settings for BYOL (Chen et al., 2020b) on CIFAR-10/100.

Barlow Twins The ResNet-18 ($\times$) encoder is trained by Barlow Twins (Zbontar et al., 2021) from scratch by 1,000 epochs with LARS-SGD optimizer (You et al., 2017). We use 256 batch size along with 0.3 learning rate and $1e-4$ weight decay. The Cosine learning rate scheduler is used with 10 epochs warmup training. The detailed data augmentation is summarized in Table 11.

Parameter	X	X'
Random crop size	32×32	32×32
Horizontal flip probability	0.5	0.5
Color jitter probability	0.8	0.8
Brightness adjustment probability	0.4	0.4
Contrast adjustment probability	0.4	0.4
Saturation adjustment probability	0.2	0.2
Hue adjustment probability	0.1	0.1
Gaussian blurring probability	0.0	0.0
Solarization probability	0.0	0.2

Table 11: Detailed image augmentation settings for Barlow Twins (Zbontar et al., 2021) on CIFAR-10/100.

B.3 IMAGENET EXPERIMENTS

Starting from scratch, the proposed SyncCP algorithm exploits in-training sparsity with the BYOL framework (Grill et al., 2020) on ImageNet-2012 dataset. The ResNet-50 encoder is trained by LARS-SGD (You et al., 2017) with 0.45 learning rate and a momentum of 0.9. We uses 0.1 for the for EMA-averaged gradient score. We use 128 batch size along with 1e-6 weight decay. The detailed image augmentations are summarized in Table 12.

Parameter	X	X'
Random crop size	224×224	224×224
Horizontal flip probability	0.5	0.5
Color jitter probability	0.8	0.8
Brightness adjustment probability	0.4	0.4
Contrast adjustment probability	0.4	0.4
Saturation adjustment probability	0.2	0.2
Hue adjustment probability	0.1	0.1
Gaussian blurring probability	1.0	0.1
Solarization probability	0.0	0.2

Table 12: Detailed image augmentation settings for BYOL (Grill et al., 2020) on ImageNet.