Unsupervised Training of Vision Transformers with Synthetic Negatives

Nikolaos Giakoumoglou Andreas Floros Kleanthis Marios Papadopoulos Tania Stathaki
Imperial College London
{nikos, andreas.floros18, kleanthis-marios.papadopoulos18, t.stathaki}@imperial.ac.uk

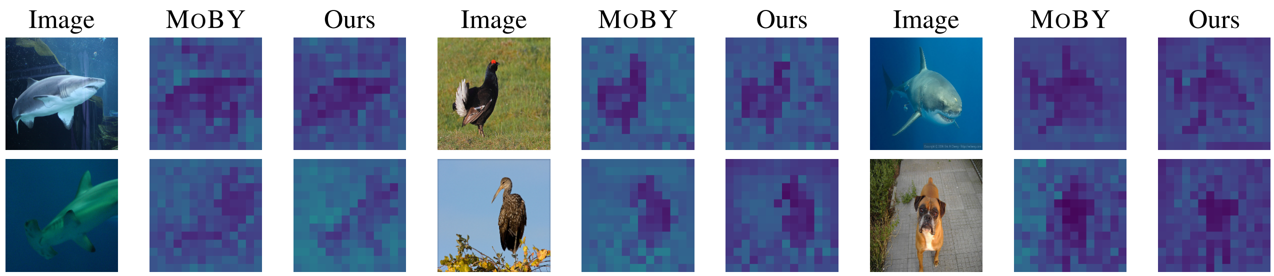


Figure 1. Self-attention patterns of the average attention head of DeiT-S from the last transformer layer for MoBY and our approach.

Abstract

This paper does not introduce a novel method per se. Instead, we address the neglected potential of hard negative samples in self-supervised learning. Previous works explored synthetic hard negatives but rarely in the context of vision transformers. We build on this observation and integrate synthetic hard negatives to improve vision transformer representation learning. This simple yet effective technique notably improves the discriminative power of learned representations. Our experiments show performance improvements for both DeiT-*S and* Swin-*T architectures*[1].

1. Introduction

Computer vision has recently witnessed two major advances. Self-supervised learning [4, 12] has fundamentally transformed how machines learn from visual data without labels. Concurrently, vision transformer architectures [9, 29] have reshaped the field by applying attention mechanisms to image understanding tasks. Self-supervised methods have proven remarkably effective for building robust visual representations [19], often referred to as *"the dark matter of intelligence"* that underpins broader machine comprehension. As Yann LeCun aptly noted, *"if AI is a cake, self-supervised learning is the bulk of the cake"*. The emergence of transformer models has complemented this progress by providing architectures capable of capturing complex relationships within visual data [9].

[1]Code will be made available upon acceptance.

Despite their effectiveness, contrastive learning approaches face a persistent challenge regarding the quality of negative examples [15]. Standard techniques rely on randomly sampling negatives from a batch [4, 6] or memory bank [12, 30], but these negatives are often too easy to distinguish, limiting the discriminative power of learned representations [10, 15].

In this work, we address this limitation by integrating synthetic hard negatives into self-supervised vision transformer training. Building upon existing momentum-based frameworks [11, 12, 31], we generate challenging negative examples that force the model to learn more discriminative features [10, 15]. Inspired by recent advances in synthetic contrastive learning [10], our approach synthesizes hard negatives *"on-the-fly"* in the feature space, creating examples that improve representation quality while maintaining stability. The key insight of our approach is that synthetic negatives provide a controlled way to increase the difficulty of the learning task [10], pushing the model to develop more robust representations.

Our main **contributions** include exploring the previously uninvestigated application of synthetic negatives in vision transformers. Specifically:

- We demonstrate that synthetic hard negatives can effectively enhance vision transformer representations.
- Our experiments reveal that most configuration settings provide sufficient contrast for the model to learn highly discriminative features.
- Our approach seamlessly integrates with existing contrastive learning frameworks.

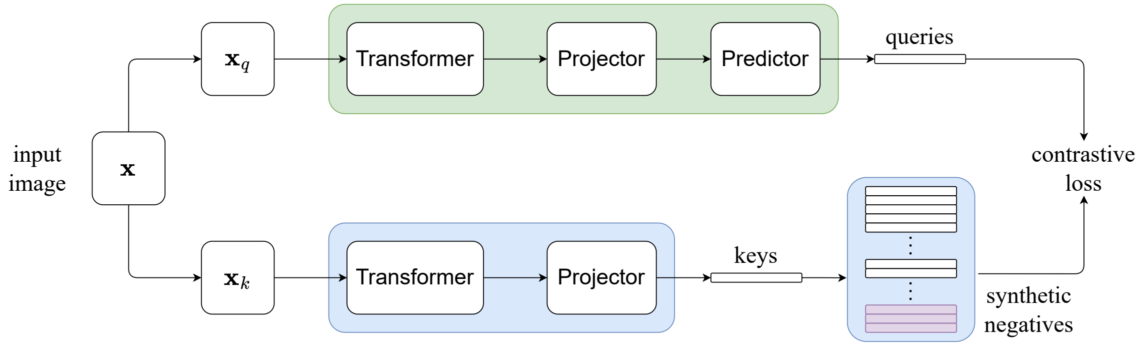


Figure 2. SYNBY framework overview. Our approach incorporates synthetic hard negatives into the MOBY framework.

2. Related Work

Self-supervised visual representation learning. Self-supervised learning has emerged as a powerful approach to learn visual representations without manual annotations. Within this paradigm, contrastive learning has shown particular promise and has been widely adopted in various forms [4, 6, 12, 26]. SIMCLR [4] demonstrated the effectiveness of a simple framework using data augmentation, large batch sizes, and nonlinear projection heads. MOCO [12] introduced a momentum encoder and queue-based mechanism, enabling contrastive learning with smaller batch sizes.

Hard negatives in contrastive learning. The quality of negative samples in contrastive learning has been a focus of extensive research [1, 7, 10, 15, 24, 30]. These studies aim to select informative negative samples and address false negatives in instance discrimination tasks. Recent work [15] explored mixing of hard negatives to create challenging contrasts, showing that harder examples lead to improved representations. Subsequent works developed this direction, with newer approaches [10] proposing systematic methods for generating synthetic hard negatives in the feature space.

Self-supervised transformers for vision. Self-supervised learning for vision transformers has rapidly evolved [2, 13]. Self-distillation methods operate without labels [3], while masked modeling approaches draw inspiration from NLP techniques [2, 13]. MOCO-V3 [6] adapted momentum-based frameworks for transformers, addressing instability through fixed patch projection and batch normalization. Other contrastive methods like MOBY [31] implemented asymmetric drop path rates and fewer stability "*tricks*".

3. Background

In this section, we introduce contrastive learning basics (Section 3.1) and our framework for generating synthetic hard negatives (Section 3.2), see Figure 2.

3.1. Contrastive Learning

Contrastive learning aims to learn representations by comparing similar and dissimilar samples. Given an image $\mathbf{x}$ and two distribution of image augmentations $\mathcal{T}$ and $\mathcal{T}'$, two augmented views of the same image are created $\mathbf{x}_q = t_q(\mathbf{x})$ and $\mathbf{x}_k = t_k(\mathbf{x})$, where $t_q \sim \mathcal{T}$ and $t_k \sim \mathcal{T}'$. These views are encoded by *online* and *target* encoders, f_q and f_k, respectively, producing vectors $\mathbf{q} = f_q(\mathbf{x}_q)$ and $\mathbf{k} = f_k(\mathbf{x}_k)$. The learning objective is to minimize the InfoNCE loss [28]:

$$\mathcal{L}(\mathbf{q}, \mathbf{k}, \mathcal{Q}) = -\log \frac{\exp(\mathbf{q}^T \cdot \mathbf{k}/\tau)}{\exp(\mathbf{q}^T \cdot \mathbf{k}/\tau) + \sum\limits_{\mathbf{n} \in \mathcal{Q}} \exp(\mathbf{q}^T \cdot \mathbf{n}/\tau)} \tag{1}$$

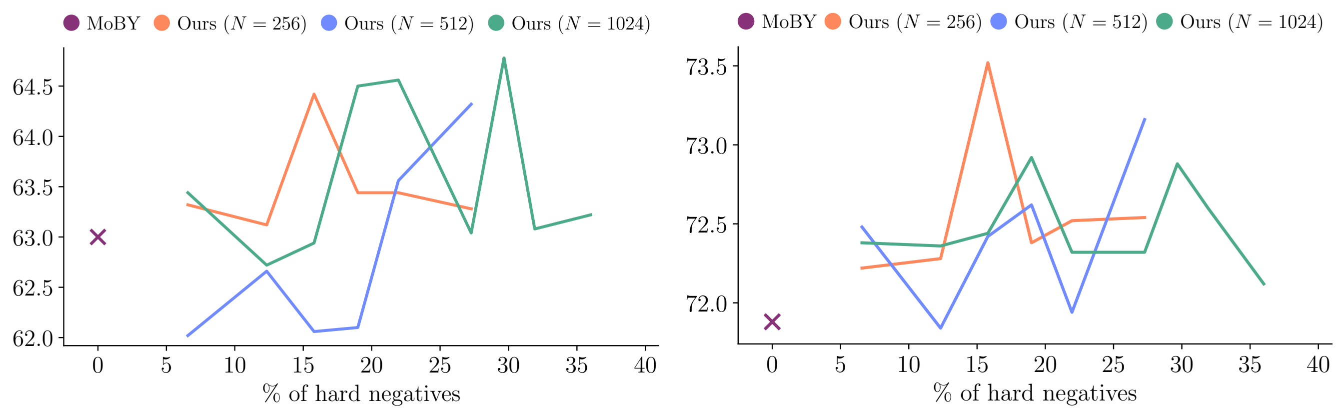
Here, $\mathcal{Q} = \{\mathbf{n}_1, \mathbf{n}_2, \ldots, \mathbf{n}_K\}$ is a set of K negative samples and τ is a temperature parameter. Negative samples are mined either from the batch [4, 6] or from a memory bank [12, 22, 26]. The encoder can be updated via momentum $\theta_k \leftarrow m \cdot \theta_k + (1 - m) \cdot \theta_q$ or through weight sharing in siamese networks ($f_k \equiv f_q$).

3.2. Synthetic Hard Negatives

Synthetic negatives provide challenging examples that help models learn more discriminative features. Let $\hat{\mathcal{Q}}^N = \text{TopK}(\{\text{sim}(\mathbf{q}, \mathbf{n}) \mid \mathbf{n} \in \mathcal{Q}\}, N)$ be the subset containing the $N < K$ hardest negatives, where $\text{sim}(\mathbf{a}, \mathbf{b})$ is the cosine similarity of ℓ_2 normalized features. The synthetic hard negatives can be abstractly represented as a function:

$$\mathbf{s} = \frac{\mathbf{s}'}{\|\mathbf{s}'\|_2} \quad \text{where} \quad \mathbf{s}' = \mathcal{F}(\mathbf{q}, \hat{\mathcal{Q}}^N; \xi) \tag{2}$$

where $\mathbf{s}'$ is the raw synthetic negative, $\mathbf{s}$ is the normalized synthetic negative, $\|\cdot\|_2$ denotes the ℓ_2 norm, and ξ represents the parameters that control the synthesis process (see Section 8). The synthetic hard negatives $\{\mathbf{s}_1, \mathbf{s}_2, \ldots\}$ are incorporated with real negatives $\mathcal{Q}$ in the contrastive loss of Equation (1), exposing the model to more challenging contrasts.



(a) DEIT-S (b) SWIN-T

Figure 3. Ablation study of different hardness selection values and synthetic negative percentages (see Section 9 for details).

4. Experiments

We develop our approach in PyTorch, building upon the implementation of MOBY [31] and SYNCO [10]. We refer to our approach as SYNBY.

4.1. Experimental Details

We pretrain SYNBY on ImageNet ILSVRC-2012 [8] and its smaller ImageNet-100 subset [16] using a DEIT-Small [9, 27] or SWIN-Tiny [20] encoder. Our implementation builds upon MOBY [31]. The encoder f_q consists of a backbone, a projection head [4], and an extra prediction head [11]; the encoder f_k has the backbone and projection head, but not the prediction head. For training, we use the AdamW optimizer [21] with a base learning rate of 0.03, weight decay of 10^{-4}, and batch size of 512. The momentum parameter starts at $m_{\text{start}} = 0.99$ and increases to 1 following a cosine schedule. For synthetic negatives, we select the top $N = 256$ hardest negatives. We use a temperature $\tau = 0.2$ for the contrastive loss of Equation (1) and a queue size $K = 4096$. We implement a cooldown period for the last 100 epochs where *no* synthetic negatives are generated. For ImageNet linear evaluation, we train a linear classifier on frozen features for 100 epochs. See Section 7 for details.

4.2. Linear Evaluation on ImageNet

Table 1 shows top-1 accuracy of our method after pretraining for 300 epochs on ImageNet ILSVRC-2012. SYNBY outperforms the MOBY baseline by 0.2% on both architectures. These results surpass other self-supervised methods like MOCO-V3 and DINO, demonstrating that synthetic hard negatives consistently improve representation quality. While a gap remains compared to supervised training, our approach improves performance *without* requiring additional labeled data or computational overhead.

Table 1. Comparison of various self-supervised learning methods on DEIT-S and SWIN-T architectures.

Method	Arch.	Params (M)	Top-1 (%)
Supervised	DEIT-S	22	79.8
Supervised	SWIN-T	29	81.3
MOCO-V3 [6]	DEIT-S	22	72.5
DINO [3]	DEIT-S	22	72.5
MOBY [31]	DEIT-S	22	72.8
MOBY [31]	SWIN-T	29	75.0
SYNBY (ours)	DEIT-S	22	73.0
SYNBY (ours)	SWIN-T	29	**75.2**

Visualizing attention. Figure 1 shows self-attention patterns comparing SYNBY and MOBY. Our method produces more focused attention maps with finer-grained patterns highlighting semantically meaningful regions, suggesting synthetic hard negatives help develop more discriminative features that target relevant visual elements.

4.3. Ablation Study

We perform ablations studies of SYNBY on ImageNet-100 pretraining for 100 epochs.

Synthetic negatives. We observe architectural differences in how DEIT and SWIN transformers respond to synthetic negatives (see Figure 3). DEIT benefits from mining negatives at either low (256) or high (1024) hardness levels, while SWIN performs well across all hardness levels. Additionally, DEIT achieves better results with moderately hard negatives at medium or high proportions, whereas SWIN performs consistently well with all proportions. This likely stems from SWIN's inductive biases requiring less aggressive negative samples than DEIT's pure transformer architecture.

Table 2. Ablation study on applying tricks of MoCo-v3.

Fixed Patch Embedding	Replace LN before MLP with BN	Top-1 (%)	
		DeiT-S	Swin-T
		66.7	67.5
✓		66.4	67.2
	✓	67.2	67.9

Table 3. Ablation study on the drop path rates.

Online dpr	*Target* dpr	Top-1 Acc. (%)	
		DeiT-S	Swin-T
0.1	0.1	61.9	74.3
0.05	0.0	65.0	75.3
0.1	0.0	65.0	75.4
0.2	0.0	64.7	72.7

Table 4. Ablation study on queue size K.

K	Top-1 Acc. (%)	
	DeiT-S	Swin-T
1024	64.5	72.5
2048	64.5	72.5
4096	64.7	72.7
8192	63.6	72.3
16384	62.6	71.6

Table 5. Ablation study on temperature τ.

τ	Top-1 Acc. (%)	
	DeiT-S	Swin-T
0.07	59.3	61.5
0.1	61.5	69.2
0.2	64.5	72.7
0.3	64.0	71.7

Table 6. Ablation study on momentum m_{start}.

m_{start}	Top-1 Acc. (%)	
	DeiT-S	Swin-T
0.99	64.5	72.7
0.993	65.2	72.2
0.996	63.8	72.4
0.999	60.3	68.6

Applying MoCo-v3 tricks. Our experiments reveal that synthetic negatives provide sufficient regularization, eliminating the need for additional stabilization techniques from MoCo-v3. As shown in Table 2, fixing the patch embedding has minimal impact on performance, suggesting our synthetic negatives already provide comparable regularization. This allows for a simpler implementation without compromising performance. Notably, replacing Layer Normalization (LN) with Batch Normalization (BN) before MLP blocks yields improvements.

Asymmetric drop path rates. The asymmetric configuration of drop path rates (dpr) significantly impacts model performance (Table 3). Unlike MoBY which uses 0.2 for the online encoder, we find a smaller rate of 0.05 is optimal when combined with synthetic negatives. This suggests the synthetic negatives provide additional regularization, reducing the need for aggressive drop path. Applying drop path only to the online encoder while keeping the target encoder stable yields the best balance.

Other hyper-parameters. The default hyperparameters from MoBY work effectively with our synthetic negative approach. As shown in Tables 4 to 6, performance remains stable across different queue sizes (best at 4096), temperatures (optimal at 0.2), and momentum values (best at 0.99). This demonstrates that synthetic negatives can be incorporated without extensive re-tuning of existing parameters. This suggests our synthetic negative generation technique integrates seamlessly with established contrastive learning frameworks, requiring minimal adaptation effort.

5. Conclusion

In this paper, we explored synthetic negatives in vision transformer pretraining. We found that synthetic negatives provide enough regularization that we do not need high drop path rates, while still requiring an asymmetric drop path rate configuration for improved performance. Importantly, our approach requires minimal adjustments to current frameworks, working in a "*plug-and-play*" manner with existing architectures. The experimental results demonstrate that SynBY further improves representation learning with minimal computational *overhead*, showing consistent gains across different transformer architectures.

Limitations. Our ablation studies were conducted on ImageNet-100, which may not fully capture the behavior on larger-scale datasets.

Future work. Synthetic hard negatives have proven effective for vision transformers, but their application could be extended to multimodal models. Exploring their integration into vision-language frameworks like CLIP represents a promising direction, potentially enhancing cross-modal contrastive learning through more challenging negative examples.

Acknowledgments

We acknowledge the computational resources and support provided by the Imperial College Research Computing Service (http://doi.org/10.14469/hpc/2232), which enabled our experiments.

References

[1] Sanjeev Arora, Hrishikesh Khandeparkar, Mikhail Khodak, Orestis Plevrakis, and Nikunj Saunshi. A theoretical analysis of contrastive unsupervised representation learning, 2019. 2

[2] Hangbo Bao, Li Dong, Songhao Piao, and Furu Wei. Beit: Bert pre-training of image transformers, 2022. 2

[3] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers, 2021. 2, 3

[4] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations, 2020. 1, 2, 3, 6

[5] Xinlei Chen and Kaiming He. Exploring simple siamese representation learning, 2020. 6

[6] Xinlei Chen, Saining Xie, and Kaiming He. An empirical study of training self-supervised vision transformers, 2021. 1, 2, 3

[7] Ching-Yao Chuang, Joshua Robinson, Yen-Chen Lin, Antonio Torralba, and Stefanie Jegelka. Debiased contrastive learning. In *Advances in Neural Information Processing Systems*, pages 8765–8775, 2020. 2

[8] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, K. Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009. 3, 6

[9] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale, 2021. 1, 3, 6

[10] Nikolaos Giakoumoglou and Tania Stathaki. Synco: Synthetic hard negatives in contrastive learning for better unsupervised visual representations, 2024. 1, 2, 3, 7

[11] Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre H. Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Daniel Guo, Mohammad Gheshlaghi Azar, Bilal Piot, Koray Kavukcuoglu, Rémi Munos, and Michal Valko. Bootstrap your own latent: A new approach to self-supervised learning, 2020. 1, 3, 6

[12] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning, 2020. 1, 2, 6, 7

[13] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners, 2021. 2

[14] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift, 2015. 6

[15] Yannis Kalantidis, Mert Bulent Sariyildiz, Noe Pion, Philippe Weinzaepfel, and Diane Larlus. Hard negative mixing for contrastive learning, 2020. 1, 2

[16] Prannay Khosla, Piotr Teterwak, Chen Wang, Aaron Sarna, Yonglong Tian, Phillip Isola, Aaron Maschinot, Ce Liu, and Dilip Krishnan. Supervised contrastive learning, 2021. 3, 6

[17] Alexander Kolesnikov, Xiaohua Zhai, and Lucas Beyer. Revisiting self-supervised visual representation learning, 2019. 7

[18] Simon Kornblith, Jonathon Shlens, and Quoc V. Le. Do better imagenet models transfer better?, 2019. 7

[19] Xiao Liu, Fanjin Zhang, Zhenyu Hou, Liming Mian, Zhaoyu Wang, Jing Zhang, and Jie Tang. Self-supervised learning: Generative or contrastive. *IEEE Transactions on Knowledge and Data Engineering*, 35(1):21–40, 2021. 1

[20] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows, 2021. 3, 6

[21] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization, 2019. 3, 6

[22] Ishan Misra and Laurens van der Maaten. Self-supervised learning of pretext-invariant representations, 2019. 2

[23] Vinod Nair and Geoffrey Hinton. Rectified linear units improve restricted boltzmann machines vinod nair. pages 807–814, 2010. 6

[24] Joshua Robinson, Ching-Yao Chuang, Suvrit Sra, and Stefanie Jegelka. Contrastive learning with hard negative samples. In *International Conference on Learning Representations*, 2021. 2

[25] Yonglong Tian, Dilip Krishnan, and Phillip Isola. Contrastive multiview coding, 2020. 7

[26] Yonglong Tian, Chen Sun, Ben Poole, Dilip Krishnan, Cordelia Schmid, and Phillip Isola. What makes for good views for contrastive learning? In *Advances in Neural Information Processing Systems*, pages 6827–6839. Curran Associates, Inc., 2020. 2

[27] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention, 2021. 3, 6

[28] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding, 2019. 2

[29] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008, 2017. 1

[30] Zhirong Wu, Yuanjun Xiong, Stella Yu, and Dahua Lin. Unsupervised feature learning via non-parametric instance-level discrimination, 2018. 1, 2

[31] Zhenda Xie, Yutong Lin, Zhuliang Yao, Zheng Zhang, Qi Dai, Yue Cao, and Han Hu. Self-supervised learning with swin transformers, 2021. 1, 2, 3, 6

[32] Jure Zbontar, Li Jing, Ishan Misra, Yann LeCun, and Stéphane Deny. Barlow twins: Self-supervised learning via redundancy reduction, 2021. 6

[33] Shaofeng Zhang, Lyn Qiu, Feng Zhu, Junchi Yan, Hengrui Zhang, Rui Zhao, Hongyang Li, and Xiaokang Yang. Align representations with base: A new approach to self-supervised learning. In *The IEEE / CVF Computer Vision and Pattern Recognition Conference*, 2022. 6

Unsupervised Training of Vision Transformers with Synthetic Negatives

Supplementary Material

Contents

6. Algorithm

Algorithm 1 provides the pseudo-code of our method.

7. Implementation Details

We develop our approach in PyTorch, building upon the implementation of MoBY. While MoBY integrates two self-supervised learning methods, MoCo-v2 and BYOL, our method combines SynCo with BYOL. Since SynCo extends MoCo-v2 by introducing synthetic hard negatives, our method reduces to MoBY when no synthetic negatives are generated. We refer to our approach as SynBY.

7.1. Pretraining

Datasets. We evaluate our method on ImageNet ILSVRC-2012 [8], which includes 1000 classes and is commonly used in previous self-supervised methods [4, 5, 32, 33]. We also conduct ablation studies on ImageNet-100 [16], a subset of 100 classes derived from ImageNet.

Architecture. Our encoder f_q consists of a backbone, a projection head [4], and an extra prediction head [11]; the encoder f_k has the backbone and projection head, but not the prediction head. The encoder f_k is updated by the moving average of f_q [11, 12]. As our base encoder, we adopt ViT-Small [9, 27] or Swin-Tiny [20] architecture without the final classification layer. Both the projection head and the prediction head are 2-layer MLPs. The hidden layers of both MLPs are 4096-d and are with ReLU [23]; the output layers of both MLPs are 256-d, without ReLU. All layers in both MLPs have batch normalization [14].

Optimization. We follow the same setting as [31]. We utilize the AdamW optimizer [21] with a base learning rate of 0.03 and a weight decay of 10^{-4}. The training schedule begins with a warm-up period during the first 30 epochs in which the learning rate linearly increases from 0 to the base learning rate. Following this, the learning rate gradually decreases to zero following a cosine decay schedule without restarts. For the target network, the exponential moving average parameter m starts from $m_{\text{start}} = 0.99$ and is increased to one during training. Specifically, we set $m \triangleq 1 - (1 - m_{\text{start}}) \cdot \left(\cos\left(\frac{\pi k}{K}\right) + 1\right)/2$, with k the current training step and K the maximum number of training steps. We use a batch size of 512 split over 4 NVIDIA L40S GPUs.

Augmentation. We use the same set of image augmentations as in BYOL [11]. First, a random patch of the image is selected and resized to 224×224 with a random horizontal flip, followed by a color distortion, consisting of a random sequence of brightness, contrast, saturation, hue adjustments, and an optional grayscale conversion. Finally Gaussian blur and solarization are applied to the patches.

Table 7. Parameters used to generate image augmentations.

Parameter	BYOL $\mathcal{T}$	BYOL $\mathcal{T}'$
Random crop probability	1.0	1.0
Flip probability	0.5	0.5
Flip probability	0.8	0.8
Brightness adjustment max intensity	0.4	0.4
Contrast adjustment max intensity	0.4	0.4
Saturation adjustment max intensity	0.2	0.2
Hue adjustment max intensity	0.1	0.1
Color dropping probability	0.2	0.2
Gaussian blurring probability	1.0	0.1
Solarization probability	0.0	0.2

Hard negatives generation. We follow the same setting as [10]. Specifically, we set $\alpha_{\text{max}} = 0.5$, $\beta_{\text{max}} = 1.5$, γ_k is randomly sampled from a uniform distribution in the range $(0, 1)$, $\sigma = 0.01$, $\delta = 0.01$, and $\eta = 0.01$ (see ξ in Equation (2)). For hard negative generation, we select the top $N = 256$ hardest negatives and set $N_i = 128$ ($i = 1, 2, \ldots, 6$) to maintain a balanced total number of generated hard negatives (see Section 8). We implement a cooldown period for the last 100 epochs where *no* synthetic negatives are generated[2].

7.2. Linear Evaluation

We follow the linear evaluation protocol of [12] and as in [17, 18], which consists in training a linear classifier on top of the frozen features pretrained with our SYNBY method without updating the backbone network parameters or batch statistics. During training, we apply spatial augmentations including random crops with resize to 224×224 pixels and horizontal flips. At test time, images are resized to 256 pixels along the shorter side using bicubic resampling, followed by a 224×224 center crop. For both stages, we normalize color channels by subtracting the mean and dividing by the standard deviation after applying augmentations. We optimize the cross-entropy loss using SGD with Nesterov momentum of 0.9 over 100 epochs with a batch size of 512. We use a base learning rate of 1.0, scaled linearly according to batch size. We employ a cosine learning rate schedule with 5 warm-up epochs and set weight decay to 0.0. We keep the backbones frozen throughout training. Importantly, we do *not* apply any other regularization techniques such as gradient clipping or logits regularization, as these can mask the true quality of learned representations.

7.3. ImageNet-100 Subsets

The list of classes from ImageNet-100[3] is randomly sampled from the original ImageNet ILSVRC-2012 dataset and is the same as that used in [25].

8. Synthetic Hard Negatives

Applying the formulation from [10], we implement *six* functions $\mathcal{F}_1, \mathcal{F}_2, ..., \mathcal{F}_6$ for generating synthetic negatives $N_1, N_2, ..., N_6$, each providing a different instantiation of $\mathcal{F}$ in Equation 2:

$$\mathcal{F}_1(\mathbf{q}, \hat{\mathcal{Q}}^N; \alpha) = \alpha \cdot \mathbf{q} + (1 - \alpha) \cdot \mathbf{n}_i \quad (3)$$

$$\mathcal{F}_2(\mathbf{q}, \hat{\mathcal{Q}}^N; \beta) = \mathbf{n}_i + \beta \cdot (\mathbf{n}_i - \mathbf{q}) \quad (4)$$

$$\mathcal{F}_3(\mathbf{q}, \hat{\mathcal{Q}}^N; \gamma) = \gamma \cdot \mathbf{n}_i + (1 - \gamma) \cdot \mathbf{n}_j \quad (5)$$

$$\mathcal{F}_4(\mathbf{q}, \hat{\mathcal{Q}}^N; \sigma) = \mathbf{n}_i + \mathcal{N}(\mathbf{0}, \sigma^2 \cdot \mathbf{I}) \quad (6)$$

$$\mathcal{F}_5(\mathbf{q}, \hat{\mathcal{Q}}^N; \delta) = \mathbf{n}_i + \delta \cdot \nabla_{\mathbf{n}_i} \text{sim}(\mathbf{q}, \mathbf{n}_i) \quad (7)$$

$$\mathcal{F}_6(\mathbf{q}, \hat{\mathcal{Q}}^N; \eta) = \mathbf{n}_i + \eta \cdot \text{sign}(\nabla_{\mathbf{n}_i} \text{sim}(\mathbf{q}, \mathbf{n}_i)) \quad (8)$$

where $\mathbf{n}_i, \mathbf{n}_j \in \hat{\mathcal{Q}}^N$ are randomly selected negative examples from the set of hardest negatives. The parameters controlling generation include: $\alpha \in (0, 0.5)$ for interpolation coefficient, $\beta \in (1, 1.5)$ for extrapolation magnitude, $\gamma \in (0, 1)$ for mixing weight between negatives, and $\sigma = \delta = \eta = 0.01$ for noise and perturbation strengths. In these equations, $\mathbf{I}$ is the identity matrix, $\mathcal{N}(\mathbf{0}, \sigma^2 \cdot \mathbf{I})$ represents Gaussian noise with zero mean and variance σ^2, $\text{sim}(\cdot, \cdot)$ is the cosine similarity function, $\nabla_{\mathbf{n}_i}$ denotes the gradient with respect to $\mathbf{n}_i$, and $\text{sign}(\cdot)$ returns the element-wise sign of the gradient. These functions produce synthetic negatives through interpolation, extrapolation, feature mixing, noise injection, gradient-based perturbation, and sign-based adversarial perturbation, respectively [10].

9. Ablations

The effectiveness of synthetic negatives depends critically on the selection of appropriate configuration parameters, particularly the hardness selection value N and the number of synthetic negatives generated from each strategy. To systematically explore this parameter space, we conducted extensive ablation studies with different configurations (Figure 3). The hardness selection value N determines how many of the most challenging negative samples from the queue are considered for synthetic negative generation. We experimented with three different values: $N \in 256, 512, 1024$. A smaller N restricts the selection to only the most similar (and thus most challenging) negatives, while a larger N includes a broader range of negative samples in the synthesis process. For each synthetic negative generation function $\mathcal{F}_i$, the parameter N_i controls how many synthetic negatives are created using that particular strategy. To maintain tractable experimental complexity, we grouped parameters as $N_1 = N_2 = N_3$ and $N_4 = N_5 = N_6$, and tested various combinations: $(N_1, N_4) \in \{ (64, 32), (128, 64), (128, 128), (256, 64), (256, 128), (256, 256), (512, 64), (512, 128), (512, 256) \}$. The proportion of synthetic negatives relative to real negatives can be quantified as:

$$p = \frac{\sum_{i=1}^{6} N_i}{K + \sum_{i=1}^{6} N_i} \quad (9)$$

where $K = 4096$ is the queue size and $\sum_{i=1}^{6} N_i$ represents the total number of synthetic negatives.

[2] As shown in [10], training with synthetic negatives for longer epochs can harm performance, potentially making the learning task too difficult to solve as the model converges.

[3] Available at: https://github.com/HobbitLong/CMC/blob/master/imagenet100.txt.

Algorithm 1 Pseudocode of SYNBY in a PyTorch-like style.

```
# f_q, f_k: transformer-based encoders (online/target)
# g_q, g_k: projector networks (online/target)
# h_q: predictor network
# odpr: online drop path rate
# tdpr: target drop path rate
# m: momentum coefficient
# t: temperature coefficient
# F: list of functions to generate synthetic negatives
# N_hard: number of hardest negatives to select
# queue1, queue2: feature queues for storing negative samples

# define the loader
loader = get_data_loader()

# online and target networks
f_online = lambda x: h_q(g_q(f_q(x, drop_path_rate=odpr)))
f_target = lambda x: g_k(f_k(x, drop_path_rate=tdpr))

for x in loader: # load a batch of images
   # get two augmented views
   v1, v2 = augment(x), augment(x)

   # forward pass
   q1, q2 = f_online(v1), f_online(v2) # queries: NxC

   with no_grad():
      # momentum update of target network
      f_target = m * f_target + (1. - m) * f_online

      # compute target features
      k1, k2 = f_target(v1), f_target(v2) # keys: NxC

   # positive logits: Nx1
   l_pos1 = bmm(q1.view(N,1,C), k2.view(N,C,1))
   l_pos2 = bmm(q2.view(N,1,C), k1.view(N,C,1))

   # negative logits: NxK
   l_neg1 = mm(q1.view(N,C), queue2.view(C,K))
   l_neg2 = mm(q2.view(N,C), queue1.view(C,K))

   # find indices of the top hardest negatives
   idxs_hard1 = topk(l_neg1, k=N_hard)
   idxs_hard2 = topk(l_neg2, k=N_hard)

   # apply all synthetic negative generation functions
   for func in F:
      # generate synthetic negatives for view 1
      s_neg1 = func(q1, queue2, idxs_hard1)
      l_syn1 = mm(q1.view(N,C), s_neg1.transpose(0,1))
      l_neg1 = cat([l_neg1, l_syn1], dim=1)

      # generate synthetic negatives for view 2
      s_neg2 = func(q2, queue1, idxs_hard2)
      l_syn2 = mm(q2.view(N,C), s_neg2.transpose(0,1))
      l_neg2 = cat([l_neg2, l_syn2], dim=1)

   # logits: Nx(1+K+N_syn) where N_syn depends on enabled types
   logits1 = cat([l_pos1, l_neg1], dim=1)
   logits2 = cat([l_pos2, l_neg2], dim=1)

   # symmetric contrastive loss
   labels = zeros(N, dtype=long) # positives are the 0-th
   loss = CrossEntropyLoss(logits1/t, labels) + CrossEntropyLoss(logits2/t, labels)

   # SGD update: online network
   loss.backward()
   update(f_online)

   # update queue
   enqueue_dequeue(queue1, k1)
   enqueue_dequeue(queue2, k2)
```