Unsupervised Training of Vision Transformers with Synthetic Negatives

Supplementary Material

6. Implementation Details

We use 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 with synthetic negatives, our method reduces to MoBY when no synthetic negatives are generated.

6.1. Pretraining

Datasets. We evaluate our method on ImageNet ILSVRC-2012 [3], which includes 1000 classes and is commonly used in previous self-supervised methods [1, 2, 18, 19]. We also conduct ablation studies on ImageNet-100 [9], a subset of 100 classes derived from ImageNet. Both datasets are well-balanced in class distribution, and the images contain iconic views of objects, as is common in vision tasks [18].

Architecture. Our online encoder, f_q, consists of a backbone, a projection head [1], and an extra prediction head [6]; the target encoder, f_k, has the backbone and projection head, but not the prediction head. The target is updated by the moving average of f_q [6, 7]. For the backbone, we adopt the ViT-Small [4, 16] or SWIN-Tiny [12] architecture without the final classification layer. Both the projection head and the prediction head are two-layer MLPs. The hidden layers of both MLPs are 4096-d and are with ReLU [14]; the output layers of both MLPs are 256-d, without ReLU. All layers in both MLPs have batch normalization [8].

Optimization. We follow the same setting as [17]. We utilize the AdamW optimizer [13] 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 zero 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 [6]. 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. The augmentation parameters are detailed in Table 2.

Table 2. Parameters used to generate image augmentations.

Parameter	BYOL	
	$\mathcal{T}$	$\mathcal{T}'$
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 negative generation. We follow the same setting as [5]. 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, as defined in Section 7. We implement a cooldown period for the last 100 epochs where *no* synthetic negatives are generated[1].

6.2. Linear Evaluation

We follow the linear evaluation protocol of [7] and as in [10, 11], 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

[1] As shown in [5], training with synthetic negatives for longer epochs can harm performance, potentially making the learning task too difficult to solve as the model converges.

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.

6.3. ImageNet-100 Subsets

The list of classes from ImageNet-100 is randomly sampled from the original ImageNet ILSVRC-2012 dataset and is the same as that used in [15]. The list is shown in Table 3.

Table 3. The list of classes from ImageNet-100, which are randomly sampled from the original ImageNet ILSVRC-2012 dataset.

List of ImageNet-100 classes			
n02869837	n01749939	n02488291	n02107142
n13037406	n02091831	n04517823	n04589890
n03062245	n01773797	n01735189	n07831146
n07753275	n03085013	n04485082	n02105505
n01983481	n02788148	n03530642	n04435653
n02086910	n02859443	n13040303	n03594734
n02085620	n02099849	n01558993	n04493381
n02109047	n04111531	n02877765	n04429376
n02009229	n01978455	n02106550	n01820546
n01692333	n07714571	n02974003	n02114855
n03785016	n03764736	n03775546	n02087046
n07836838	n04099969	n04592741	n03891251
n02701002	n03379051	n02259212	n07715103
n03947888	n04026417	n02326432	n03637318
n01980166	n02113799	n02086240	n03903868
n02483362	n04127249	n02089973	n03017168
n02093428	n02804414	n02396427	n04418357
n02172182	n01729322	n02113978	n03787032
n02089867	n02119022	n03777754	n04238763
n02231487	n03032252	n02138441	n02104029
n03837869	n03494278	n04136333	n03794056
n03492542	n02018207	n04067472	n03930630
n03584829	n02123045	n04229816	n02100583
n03642806	n04336792	n03259280	n02116738
n02108089	n03424325	n01855672	n02090622

7. Synthetic Hard Negatives

Applying the formulation from [5], we implement *six* functions $\mathcal{F}_1, \mathcal{F}_2, ..., \mathcal{F}_6$ for generating synthetic negatives $N_1, N_2, \ldots, N_6$, each providing a different instantiation of $\mathcal{F}$ in Equation (2):

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

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

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

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

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

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

where $\mathbf{n}_i, \mathbf{n}_j \in \mathcal{Q}^N$ 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 [5].

8. 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. 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:

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

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

Broader Impact

The improvements demonstrated by SynBY have potential implications beyond the specific task of image classification. Our approach contributes to more data-efficient deep learning. This is particularly valuable in domains where labeled data is scarce or expensive to obtain.

References

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

[2] Xinlei Chen and Kaiming He. Exploring simple siamese representation learning, 2020. 1

[3] 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. 1

[4] 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

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

[6] 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

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

[8] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift, 2015. 1

[9] Prannay Khosla, Piotr Teterwak, Chen Wang, Aaron Sarna, Yonglong Tian, Phillip Isola, Aaron Maschinot, Ce Liu, and Dilip Krishnan. Supervised contrastive learning, 2021. 1

[10] Alexander Kolesnikov, Xiaohua Zhai, and Lucas Beyer. Revisiting self-supervised visual representation learning, 2019. 1

[11] Simon Kornblith, Jonathon Shlens, and Quoc V. Le. Do better imagenet models transfer better?, 2019. 1

[12] 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. 1

[13] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization, 2019. 1

[14] Vinod Nair and Geoffrey Hinton. Rectified linear units improve restricted boltzmann machines vinod nair. pages 807–814, 2010. 1

[15] Yonglong Tian, Dilip Krishnan, and Phillip Isola. Contrastive multiview coding, 2020. 2

[16] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention, 2021. 1

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

[18] Jure Zbontar, Li Jing, Ishan Misra, Yann LeCun, and Stéphane Deny. Barlow twins: Self-supervised learning via redundancy reduction, 2021. 1

[19] 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. 1