SynCo: Synthetic Hard Negatives in Contrastive Learning for Better Unsupervised Visual Representations

Supplementary Material

7. Algorithm

Algorithm 1 provides the pseudo-code of SynCo.

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

```
# f_q, f_k: encoder networks for query and key
# queue: dictionary as a queue of K keys (CxK)
# m: momentum
# t: temperature
# hard_neg_functions: list of functions to generate
# synthetic negatives (type 1 to 6)

f_k.params = f_q.params # initialize
for x in loader: # load a minibatch x with N samples
   x_q = aug(x) # a randomly augmented version
   x_k = aug(x) # another randomly augmented version

   q = f_q.forward(x_q) # queries: NxC
   k = f_k.forward(x_k) # keys: NxC
   k = k.detach() # no gradient to keys

   # positive logits: Nx1
   l_pos = bmm(q.view(N,1,C), k.view(N,C,1))

   # negative logits: NxK
   l_neg = mm(q.view(N,C), queue.view(C,K))

   # find indices of the top-(N_hard) hard negatives
   idxs_hard = topk(l_neg, k=N_hard)

   # generate hard negatives
   for func in hard_neg_functions:
      # generate hard negatives of type i
      s_neg = func(q, queue, idxs_hard)
      # compute logits for synthetic negatives
      l_syn = bmm(q.view(N,C), s_neg.view(N,C))
      # append hard negatives logits
      l_neg = cat([l_neg, l_syn], dim=1)

   # logits: Nx(1+K+N_hard)
   logits = cat([l_pos, l_neg], dim=1)

   # contrastive loss
   labels = zeros(N) # positives are the 0-th
   loss = CrossEntropyLoss(logits/t, labels)

   # SGD update: query network
   loss.backward()
   update(f_q.params)

   # momentum update: key network
   f_k.params = m*f_k.params+(1-m)*f_q.params

   # update dictionary
   enqueue(queue, k) # enqueue the current minibatch
   dequeue(queue) # dequeue the earliest minibatch
```

`bmm`: batch matrix multiplication; `mm`: matrix multiplication; `cat`: concatenation; `topk`: returns the indices of the top-k values;

8. Implementation Details

We implement SynCo in PyTorch following the implementation of MoCo[1]. Specifically, we follow the same setting as MoCo-v2.

[1]Available at: https://github.com/facebookresearch/moco.

8.1. Pretraining

Datasets. We evaluate the proposed method on ImageNet ILSVRC-2012[2] [6], which includes 1000 classes and is commonly used in previous self-supervised methods [3, 4, 26, 29]. The dataset consists of 1.28 million training images and 50,000 validation images. We also conduct ablation studies on ImageNet-100 [13], a subset of 100 classes derived from ImageNet ILSVRC-2012, with 126,689 training images and 5,000 validation images. Both datasets are well-balanced in class distribution, and the images contain iconic views of objects, as is common in vision tasks [9, 26].

Augmentations. Each input image is transformed twice to generate two different views. For SynCo, we use the same augmentation as used in [5] and [12] for a fair comparison. We transform each input image with two sampled augmentations to produce two distorted versions of the input. The augmentation pipeline consists of random cropping, resizing to 224×224, randomly flipping the images horizontally, applying color distortion, optionally converting to grayscale, adding Gaussian blurring.

Architecture. Both the encoder f_q and f_k consist of a backbone and a projection head. The encoder f_k is updated by the moving average of f_q. As our base encoder, we adopt ResNet-50 (2048 output units). The projection head is a 2-layer MLP, following [5]: the hidden layers of the MLP are 2048-d and are with ReLU [18]; the output layer of the MLP is 128-d, without ReLU.

Optimization. We follow the same setting as [5]. We utilize the SGD optimizer [20] with a base learning rate of 0.03 ($= 0.03 \times \text{batch\_size}/256$), where we scale the learning rate with the batch size as in [3], and a weight decay of 10^{-4}. The training schedule begins with a warm-up period during the first 10 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. The batch size for ImageNet is set to 256 distributed over 4 NVIDIA A40 GPUs. The total training duration is set to 200/800 epochs for ImageNet. For pretraining, SynCo takes approximately 43 hours (1.8 days) and 8 kWh of power for 100 epochs.

[2]Available at: https://www.image-net.org/.

Hyperparameters. We empirically set SynCo's hyperparameters to $\sigma = 0.01$, $\delta = 0.01$, and $\eta = 0.01$. A thorough analysis of these hyperparameters revealed no significant difference in performance when these values are varied within reasonable bounds (also see Section 9.3), indicating that our method is robust to a range of practical settings. For hard negative generation, we select the top $N = 1024$ hardest negatives and set $N_1 = N_2 = N_3 = 256$ and $N_4 = N_5 = N_6 = 64$ to maintain a balanced total number of generated hard negatives. A detailed analysis of the choice of N_i, $i = 1, \ldots, 6$ is provided in Section 9.3. We tested various similarity functions, including cosine similarity, Euclidean, and Mahalanobis distances, for generating gradient-based synthetic hard negatives. Our results revealed no significant differences in model performance across these similarity measures. Therefore, we opted to use the dot product similarity function, which simplifies computation and aligns with the InfoNCE loss used in SynCo's contrastive learning framework.

8.2. Linear Evaluation

We follow the linear evaluation protocol of [11] and as in [3, 8, 14, 15, 22], which consists in training a linear classifier on top of the frozen representation, i.e., without updating the network parameters nor the batch statistics. At training time, we apply spatial augmentations, i.e., random crops with resize to 224×224 pixels, and random flips. At test time, images are resized to 256 pixels along the shorter side using bicubic resampling, after which a 224×224 center crop is applied. In both cases, we normalize the color channels by subtracting the average color and dividing by the standard deviation, after applying the augmentations. We optimize the cross-entropy loss using SGD with Nesterov momentum over 100 epochs, using a batch size of 256 and a momentum of 0.9. We use a learning rate of 30.0 for ImageNet ILSVRC-2012 and 10.0 for ImageNet-100. We train using 4 NVIDIA A40 GPUs.

8.3. Semi-supervised Training

We follow the semi-supervised learning protocol of [3, 15, 27]. We first initialize the network with the parameters of the pretrained representation, and fine-tune it with a subset of ImageNet ILSVRC-2012 labels. At training time, we apply spatial augmentations, i.e., random crops with resize to $224\times$ 224 pixels and random flips. At test time, images are resized to 256 pixels along the shorter side using bicubic resampling, after which a 224×224 center crop is applied. In both cases, we normalize the color channels by subtracting the average color and dividing by the standard deviation (computed on ImageNet), after applying the augmentations. We optimize the cross-entropy loss using SGD with Nesterov momentum. We used a batch size of 256, a momentum of 0.9. Similar to [1], we sweep over the learning rates $\{0.01, 0.02, 0.05, 0.1, 0.005\}$ and the number of epochs $\{30, 60\}$. We train using 4 NVIDIA A40 GPUs.

8.4. Object Detection

We follow the object detection protocol of [5, 11] and as used in [12]. We first initialize the network with the parameters of the pretrained representation, and fine-tune it on PASCAL VOC and COCO datasets. During training, we apply spatial augmentations, specifically random resizing and random horizontal flipping. During testing, images are resized to a fixed size of 800 pixels along the shorter side. The R50-C4 backbones, similar to those used in Detectron2 [25], conclude at the `conv4` stage. Subsequently, the box prediction head is composed of the `conv5` stage, which includes global pooling, followed by a BN layer. We train using 8 NVIDIA RTX 6000 GPUs.

PASCAL VOC object detection. We use a Faster R-CNN [19] with the SGD optimizer at a base learning rate of 0.02, a momentum of 0.1, and a weight decay of 0.0001, and a batch size of 16. The model is trained for 24,000 iterations using a step learning rate scheduler, where the learning rate is reduced at 18,000 and 22,000 iterations. Images are scaled to 480×800 pixels during training and resized to 800 pixels on the longer side for inference.

COCO object detection. We use a Mask R-CNN [10] with the SGD optimizer at a base learning rate of 0.02, a momentum of 0.1, and a weight decay of 0.0001, and a batch size of 16. The model is trained for 180,000 iterations using a step learning rate scheduler, where the learning rate is reduced at 120,000 and 160,000 iterations. A warm-up period is applied for the first 100 iterations. Images are resized to 640×800 pixels during training and normalized to 800 pixels on the longer side for inference.

8.5. Alignment and Uniformity

We follow the protocol of [12] but training the network 100 epochs on ImageNet-100. We calculate the alignment and uniformity based on [24]. The alignment loss $\mathcal{L}_{\text{align}}$ and uniformity loss $\mathcal{L}_{\text{uniform}}$ are computed as follows:

$$\mathcal{L}_{\text{align}}(\mathbf{x}, \mathbf{y}) = \mathbb{E}_{(\mathbf{x},\mathbf{y})\sim p_{\text{data}}} \left[\|\mathbf{x} - \mathbf{y}\|_2^{\alpha}\right] \tag{11}$$

$$\mathcal{L}_{\text{uniform}}(\mathbf{x}) = \log \mathbb{E}_{\mathbf{x},\mathbf{y}\sim p_{\text{data}}} \left[\exp(-t\|\mathbf{x} - \mathbf{y}\|_2^2)\right] \tag{12}$$

where $\mathbf{x}$ and $\mathbf{y}$ are the embeddings from the network, α is a hyperparameter typically set to 2, and t controls the sharpness of the distribution, also set to 2. Here, p_{data} represents the empirical distribution of the data, from which pairs of embeddings $(\mathbf{x}, \mathbf{y})$ are sampled. We implement these losses in PyTorch following the original implementation[3].

[3]Available at: https://github.com/Ssnl/align_uniform.

8.6. ImageNet-100 Subsets

The list of classes from ImageNet-100[4] is randomly sampled from the original ImageNet ILSVRC-2012 dataset and is the same as that used in [21].

8.7. Image Augmentations

During self-supervised training, SynCo uses the same augmentation as [5]. The augmentation parameters are detailed in Table 5.

Table 5. Parameters used to generate image augmentations.

Parameter	MoCo-v2 $\mathcal{T}$
Random crop probability	1.0
Horizontal flip probability	0.5
Vertical flip probability	0.8
Brightness adjustment max intensity	0.4
Contrast adjustment max intensity	0.4
Saturation adjustment max intensity	0.2
Hue adjustment max intensity	0.1
Color dropping probability	0.2
Gaussian blurring probability	0.5
Solarization probability	0.0

9. Additional Results

In this section we provide additional results of SynCo.

9.1. Transferring to Detection

We evaluate the SynCo representation using a pretrained ResNet-50 model trained for 800 epochs on VOC dataset. The results are shown in Table 6. SynCo demonstrates faster training, achieving better results at lower epochs compared to MoCo-v2. At 200 epochs, SynCo already surpasses MoCo-v2 in terms of AP_{50} and AP_{75}. However, when training is extended to 800 epochs, MoCo-v2 and SynCo perform on par, with both methods reaching similar performance.

Table 6. Results for object detection on PASCAL VOC. The values in bold indicate the maximum of each column.

Method	Epochs	AP	AP_{50}	AP_{75}
Supervised	90	53.5	81.3	58.8
MoCo [11]	200	55.9	81.5	62.6
MoCo-v2 [5]	200	57.0	82.4	63.6
MoCo-v2 [5]	800	**57.4**	82.5	**64.0**
SynCo (ours)	200	57.2	82.6	63.9
SynCo (ours)	800	**57.4**	**82.8**	**64.0**

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

9.2. Class Concentration Analysis

To quantify the overall structure of the learned latent space, we examine the relationship between within-class and between-class distances. Figure 6 compares the distribution of ratios between inter-class and intra-class ℓ_2-distances of representations learned by different MoCo-based contrastive learning methods on the ImageNet validation set. A higher mean ratio indicates that the representations are better concentrated within their corresponding classes while maintaining better separation between different classes, suggesting improved linear separability (following Fisher's linear discriminant analysis principles [7]).

Table 7. Statistical summary of the ratio between inter-class and intra-class distances for different MoCo-based methods. Higher mean indicates better class separation while lower standard deviation suggests more consistent feature learning across different classes.

Method	Epochs	Mean ↑	Median	Std ↓
Supervised	90	1.381	1.369	**0.110**
MoCo-v1 [11]	200	1.012	0.999	0.115
MoCo-v2 [5]	200	1.061	0.971	0.358
MoCo-v2 [5]	800	1.146	1.043	0.375
PCL-v1 [17]	200	0.930	0.869	0.312
PCL-v2 [17]	200	0.988	0.866	0.419
SynCo (ours)	200	1.104	1.001	0.383
SynCo (ours)	800	**1.384**	1.282	0.361

As shown in Table 7, SynCo trained for 800 epochs achieves the highest mean ratio (1.384) among all MoCo-based methods, approaching and slightly surpassing the supervised baseline (1.381). A higher mean ratio indicates better class separability, which is crucial for downstream classification tasks. This superior performance can be attributed to SynCo's synthetic hard negative generation strategies, which help create more discriminative feature representations.

The standard deviation of the ratio distribution provides insight into the consistency of learned features across different classes. Lower standard deviation suggests more uniform feature learning across all classes. While the supervised baseline achieves the lowest standard deviation (0.110), among MoCo-based methods, MoCo-v1 shows comparable consistency (0.115).

Notably, both SynCo variants (200 and 800 epochs) consistently outperform their MoCo-v2 counterparts at equivalent training epochs in terms of mean ratio (1.104 vs 1.061 at 200 epochs, and 1.384 vs 1.146 at 800 epochs), demonstrating the effectiveness of synthetic hard negatives in learning more discriminative features. The improvement in class concentration metrics aligns with SynCo's superior performance on downstream tasks, particularly in scenarios requiring fine-grained discrimination between similar classes. By focusing exclusively on methods built upon the MoCo framework,

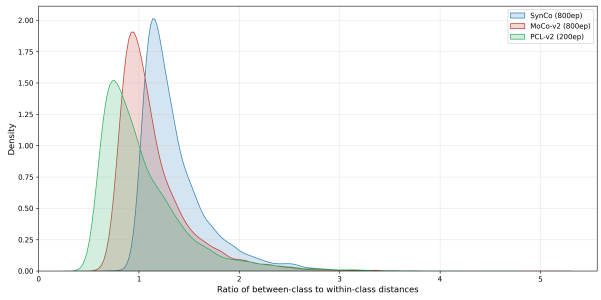


Figure 6. Distribution of the ratio between inter-class and intra-class distances for different MoCo-based methods. Higher values indicate better class separation. For clarity, we only show MoCo-v2 [5] (800 epochs), PCL-v2 [17] (200 epochs), and SynCo (800 epochs).

this comparison ensures a fair evaluation of SynCo's contributions to contrastive learning.

9.3. Ablations on CIFAR-100

We perform additional ablation studies on CIFAR-100 [16] for 32×32 images for 100-way classification. We use the same settings as previously discussed with the following differences. We adopt a ResNet-18 (512 output units) [9] architecture without the final classification layer, replacing the original 7×7 convolutional layer (`conv1`) with a 3×3 convolution that has a stride of 1 and removing the initial max pooling layer (`maxpool`). The batch size for CIFAR-10/100 is set to 256, using a single NVIDIA RTX 6000 GPU, and the total training duration is set to 1,000 epochs. Unless stated otherwise, we use $K = 16$k. We report both top-1 and top-5 accuracies as percentages on the test set. When training a linear classifier on top of frozen features, we use a learning rate of 3.0.

Ablation on parameters. We evaluate the impact of the parameters σ, δ, and η on SynCo's performance, specifically focusing on type 4, type 5, and type 6 negatives. To determine the optimal settings, we empirically test three sets of values for each parameter: $0.1, 0.05, 0.01$. The results, illustrated in Figure 7, indicate that training SynCo with different values of these parameters yields similar performance across all configurations.

Effect of hard negative sampling. We evaluate SynCo by first training without hard negatives (equivalent to MoCo-v2) and then by incorporating each type of hard negative individually, as well as in combination. Additionally, we test different configurations of the number of hard negatives (N_1 through N_6) to find the optimal settings. The results in Figure 8 show that any incorporation of hard negatives accelerates convergence and improves top-1 accuracy, regardless of type. Increasing the total number of hard negatives beyond $N = 1024$ (e.g., to $N = 2048$) does not further enhance performance, consistent with findings in MoCHI.

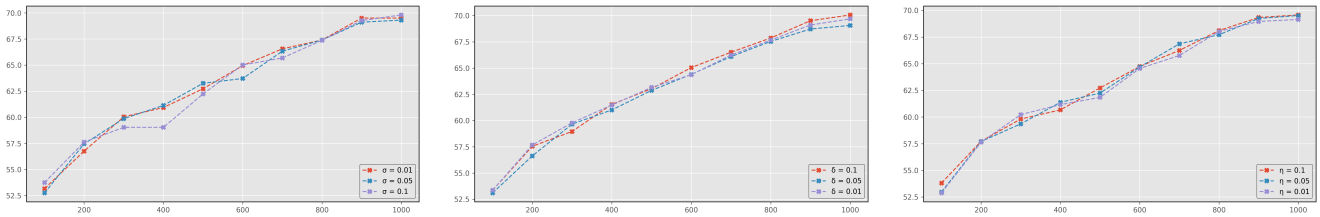
Ablation on queue size. We evaluate the performance of SynCo across various queue sizes. The results, shown in Figure 9, compare the top-1 accuracy of SynCo and MoCo-v2 across these different queue sizes. SynCo initially performs on par with MoCo-v2, with a minimal performance gap, suggesting that excessively challenging negatives may initially hinder learning efficacy. As the queue size increases, both SynCo and MoCo-v2 show comparable performance, converging further as the queue size maxes out.

Ablation on batch size. We evaluate the effect of varying batch sizes on the performance of SynCo. We tested batch sizes of $64, 128, 256, 512, 1024$, and 4096. The results are shown in Figure 10. SynCo consistently outperforms MoCo-v2 across all batch sizes, even at the smallest batch size of 64. However, larger batch sizes generally lead to degraded performance for both methods, likely due to the dilution of gradient signals when averaging over larger batches.

10. Additional Discussion

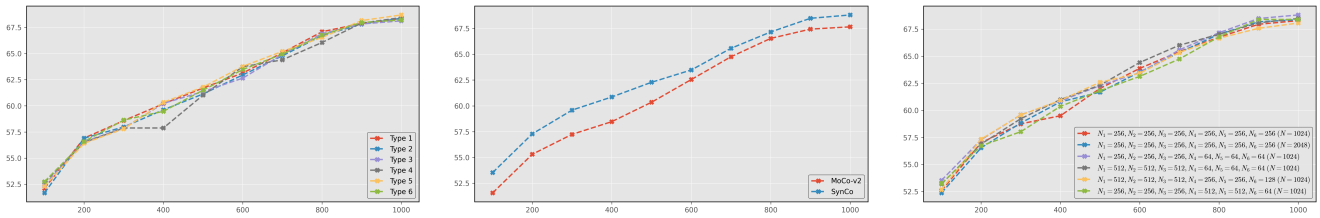
Intuition of synthetic hard negatives. Each SynCo strategy improves model generalization through challenging contrasts. Type 1 interpolates between query and hard negatives, increasing sample diversity throughout training. Type 2 extrapolates beyond the query, pushing representation space boundaries and improving robustness to difficult contrasts. Type 3 combines pairs of hard negatives, encouraging more generalized and robust feature learning. Type 4 injects Gaussian noise, promoting invariance to minor feature fluctuations and enhancing generalization. Type 5 modifies embeddings based on similarity gradients, refining discriminatory power by directing the model towards harder negatives. Type 6 applies adversarial perturbations, creating the most challenging contrasts to distinguish deceptively similar samples.

Using hard negatives for model regularization. SynCo addresses existing limitations by generating hard negatives on-the-fly, reducing computational overhead while maintaining diverse contrasts. It regularizes the network through synthetic hard negatives, aligning with vicinal risk minimization [2]. This encourages learning robust features over memorization, addressing poor generalization common in empirical risk minimization [23, 28]. The diverse synthetic negatives create a comprehensive learning environment, improving generalization across datasets and tasks. This approach reduces overfitting and enhances robustness to data variations, leading to more robust representations [28].



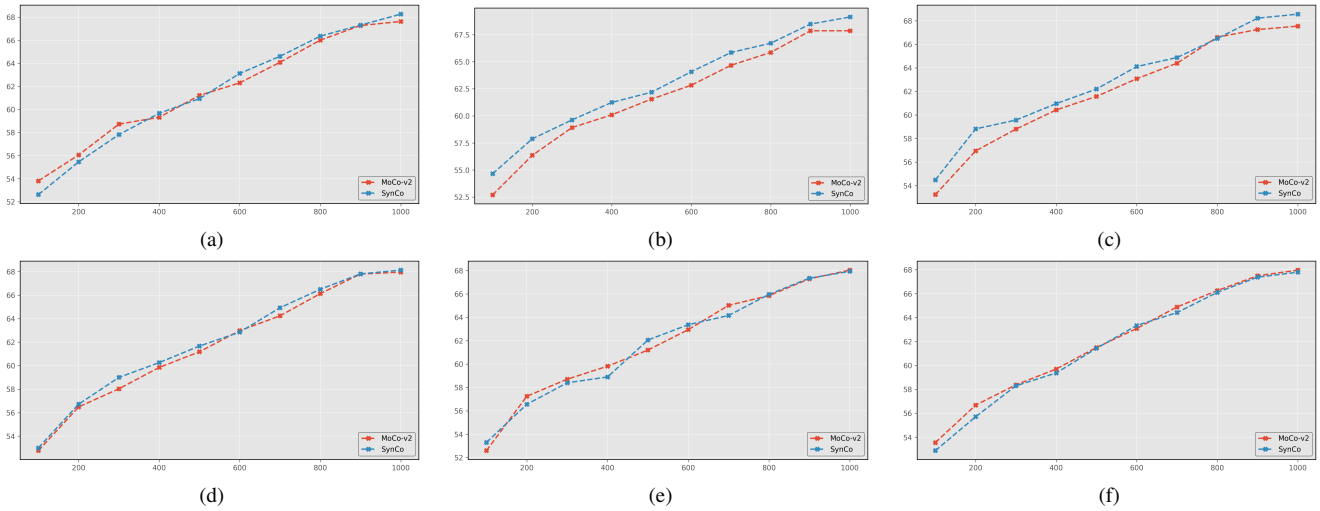
(a) (b) (c)

Figure 7. Top-1 accuracy on CIFAR-100, evaluated every 100 epochs over 1000 epochs of training with varying parameter values. (a) Performance with different σ values. (b) Performance with different δ values. (c) Performance with different η values.



(a) (b) (c)

Figure 8. Top-1 accuracy on CIFAR-100, evaluated every 100 epochs over 1000 epochs of training. (a) Performance of SynCo with one type of hard negative at a time. (b) Comparison of SynCo without hard negatives (equivalent to MoCo-v2) and with all hard negatives combined. (c) Performance of SynCo with varying numbers of hard negatives N_1 through N_6. Numbers in parentheses represent the maximum N chosen from the queue $\hat{\mathcal{Q}}$.



(a) (b) (c)

(d) (e) (f)

Figure 9. Top-1 accuracy on CIFAR-100, evaluated every 100 epochs over 1000 epochs of training, comparing SynCo and MoCo-v2. (a) With queue size $K = 1024$. (b) With $K = 4096$. (c) With $K = 8192$. (d) With $K = 16384$. (e) With $K = 32768$. (f) With $K = 65536$.

Considerations for parameter tuning and optimal use of synthetic negatives. In our experiments, we searched for optimal parameters for each type of synthetic negative. We used all types of synthetic negatives to demonstrate overall improvements. However, incorporating fewer synthetic negatives, rather than all, could potentially lead to higher accuracy. Here, our focus was on proposing the concept of synthetic negatives rather than searching for the optimal combination. The optimal combination of synthetic negatives depends on the specific dataset and task. We also did not exhaustively search for the most effective number of synthetic negatives to generate. Instead, we conducted initial experiments to assess their effectiveness.

Implications of synthetic hard negatives in broader contexts. The introduction of synthetic hard negatives in con-

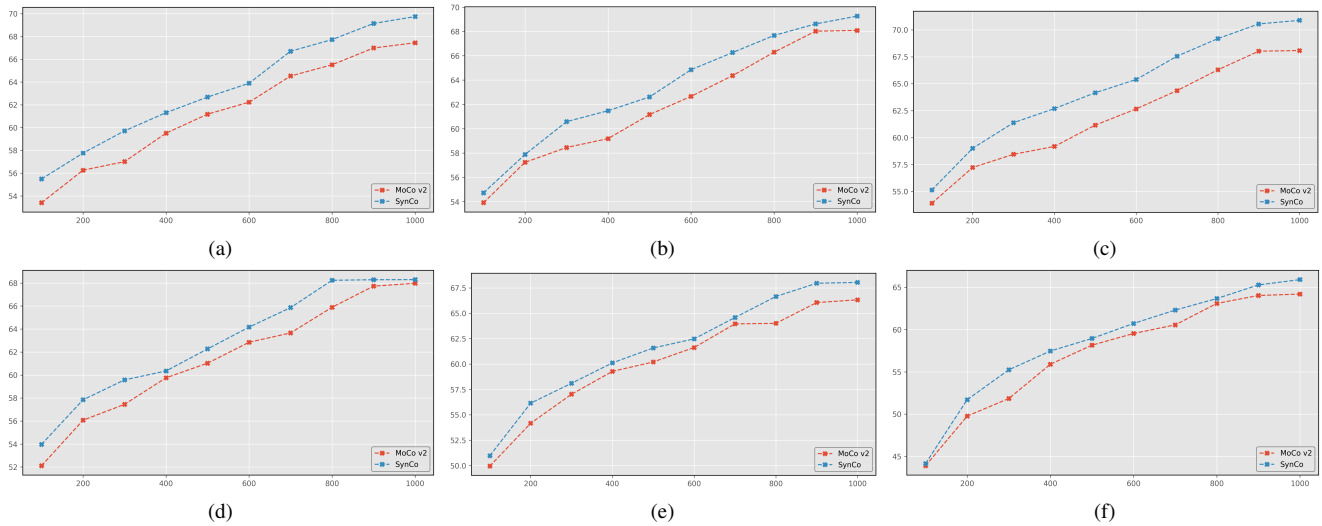


Figure 10. Top-1 accuracy on CIFAR-100, evaluated every 100 epochs over 1000 epochs of training, comparing SynCo with MoCo-v2. (a) With batch size of 64. (b) With 128. (c) With 256. (d) With 512. (e) With 1024. (f) With 2048.

trastive learning not only improves model performance in traditional image classification and detection tasks but also holds potential for applications beyond the current scope. Synthetic hard negatives can be adapted for various modalities, including text, audio, and multi-modal learning environments. For instance, in natural language processing, generating challenging negative samples could enhance tasks such as sentence similarity, text classification, and language translation. Similarly, in audio processing, synthetic hard negatives might improve tasks like speaker recognition or audio event detection. Moreover, the adaptability of synthetic hard negatives opens up possibilities for future research into domain adaptation and transfer learning. By incorporating domain-specific hard negatives, models can better generalize across different domains, addressing the challenge of domain shift in practical applications. This adaptability also suggests that synthetic hard negatives could be a crucial component in developing more robust, generalizable machine learning systems across various fields.

11. Broader Impact

The presented research should be categorized as research in the field of unsupervised learning. This work may inspire new algorithms, theoretical, and experimental investigation. The algorithm presented here can be used for many different vision applications and a particular use may have both positive or negative impacts, which is known as the dual use problem. Besides, as vision datasets could be biased, the representation learned by SynCo could be susceptible to replicate these biases.

12. Checkpoint Availability

The pre-trained model checkpoints for models trained on the ImageNet ILSVRC-2012 dataset are available for download: 200-epoch model (top-1 linear evaluation accuracy 68.1%) and 800-epoch model (top-1 linear evaluation accuracy 70.6%).

References

[1] Mathilde Caron, Ishan Misra, Julien Mairal, Priya Goyal, Piotr Bojanowski, and Armand Joulin. Unsupervised learning of visual features by contrasting cluster assignments. In *Advances in Neural Information Processing Systems*, pages 9912–9924. Curran Associates, Inc., 2020. 2

[2] Olivier Chapelle, Jason Weston, and Léon Bottou. Vicinal risk minimization. In *Proceedings of the 13th International Conference on Neural Information Processing Systems*, pages 416–422, 2000. 4

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

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

[5] Xinlei Chen, Haoqi Fan, Ross Girshick, and Kaiming He. Improved baselines with momentum contrastive learning, 2020. 1, 2, 3, 4

[6] 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

[7] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer, New York, second edition, 2009. 3

[8] 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. 2

[9] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015. 1, 4

[10] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn, 2018. 2

[11] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning, 2020. 2, 3

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

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

[14] Alexander Kolesnikov, Xiaohua Zhai, and Lucas Beyer. Revisiting self-supervised visual representation learning, 2019. 2

[15] Simon Kornblith, Jonathon Shlens, and Quoc V. Le. Do better imagenet models transfer better?, 2019. 2

[16] Alex Krizhevsky. Learning multiple layers of features from tiny images. pages 32–33, 2009. 4

[17] Junnan Li, Pan Zhou, Caiming Xiong, and Steven C. H. Hoi. Prototypical contrastive learning of unsupervised representations, 2021. 3, 4

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

[19] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks, 2016. 2

[20] Sebastian Ruder. An overview of gradient descent optimization algorithms, 2017. 1

[21] Yonglong Tian, Dilip Krishnan, and Phillip Isola. Contrastive multiview coding, 2020. 3

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

[23] Vladimir N. Vapnik. *Statistical Learning Theory*. Wiley, 1998. 4

[24] Tongzhou Wang and Phillip Isola. Understanding contrastive representation learning through alignment and uniformity on the hypersphere. In *International Conference on Machine Learning*, pages 9929–9939. PMLR, 2020. 2

[25] Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick. Detectron2. https://github.com/facebookresearch/detectron2, 2019. 2

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

[27] Xiaohua Zhai, Joan Puigcerver, Alexander Kolesnikov, Pierre Ruyssen, Carlos Riquelme, Mario Lucic, Josip Djolonga, Andre Susano Pinto, Maxim Neumann, Alexey Dosovitskiy, Lucas Beyer, Olivier Bachem, Michael Tschannen, Marcin Michalski, Olivier Bousquet, Sylvain Gelly, and Neil Houlsby. A large-scale study of representation learning with the visual task adaptation benchmark, 2020. 2

[28] Hongyi Zhang, Moustapha Cisse, Yann N. Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412*, 2017. 4

[29] 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