SynCo: Synthetic Hard Negatives for Contrastive Visual Representation Learning

Supplementary Material

Contents

8. Algorithm

Algorithm 1 provides the pseudo-code of SynCo, followed by the detailed implementation of the six distinct types of synthetic hard negatives used in our approach (Algorithms 2 to 7).

9. Implementation Details

We implement SynCo in PyTorch [70] following the implementation of MoCo[1]. Specifically, we follow the same setting as MoCo-v2.

9.1. Pretraining

Datasets. We evaluate the proposed method on ImageNet ILSVRC-2012[2] [22], which includes 1000 classes and is commonly used in previous self-supervised methods [14, 16, 100, 103]. The dataset consists of 1.28 million training images and 50,000 validation images. We also conduct ablation studies on ImageNet-100 [50], a subset of 100 classes derived from ImageNet, 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 [40, 100].

Augmentation. Each input image is transformed twice to generate two different views. For SynCo, we use the same augmentation as used in [17] and [49] 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 [17]: the hidden layers of the MLP are 2048-d and are with ReLU [69]; the output layer of the MLP is 128-d, without ReLU.

Optimization. We follow the same setting as [17]. We utilize the SGD optimizer [75] 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 [14], 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

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

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

is set to 256 distributed over 4 NVIDIA L40 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.

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 11), 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 11. 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. For detailed configuration of SynCo pretraining, including architecture and optimization parameters, see Table 5.

Parameter	**Value**
Architecture	
Backbone	ResNet-50
Projection head	2-layer MLP
Projection head activation	ReLU
Optimization	
Optimizer	SGD
Momentum	0.9
Base learning rate	0.03
Weight decay	10^{-4}
Warm-up	10 epochs
Batch size	256
Training epochs	200/800 epochs
Training time	$\sim$43 hours/100 epochs
MoCo	
Queue size K	65536
Momentum m	0.999
Temperature τ	0.2
SynCo	
Hardest negatives N	1024
Synthetic N_i, $i = 1, 2, 3$	256
Synthetic N_i, $i = 4, 5, 6$	64
Hyperparameters σ, δ, η	0.01

Table 5. **Architecture and optimization hyperparameters used for SynCo pretraining.** This table lists all architectural, optimization, MoCo, and SynCo-specific settings used during contrastive pretraining.

9.2. Linear Evaluation

We follow the linear evaluation protocol of [42] and as in [14, 35, 52, 54, 85], 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 do not use any regularization methods such as weight decay, gradient clipping [21], tclip [4], or logits regularization. We use a learning rate of 30.0 for ImageNet ILSVRC-2012 and 10.0 for ImageNet-100. We train using 4 NVIDIA L40 GPUs.

9.3. Semi-supervised Training

We follow the semi-supervised learning protocol of [14, 35, 54, 101]. 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 1024, a momentum of 0.9. We do not use any regularization methods such as weight decay, gradient clipping [21], tclip [4], or logits rescaling. Similar to [10], 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 L40 GPUs.

9.4. Object Detection

We follow the object detection protocol of [17, 42]. We first initialize the network with the parameters of the pretrained representation, and fine-tune it on PASCAL VOC [28][3] and COCO [59][4] datasets. During training, we apply spatial

[3]Available at https://host.robots.ox.ac.uk/pascal/VOC/.

[4]Available at https://cocodataset.org/.

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 [94], 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 [73] 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 [41] 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. For the $1\times$ schedule, the model trains for 90,000 iterations with learning rate reductions at 60,000 and 80,000 iterations. For the $2\times$ schedule, it trains for 180,000 iterations with learning rate reductions 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.

9.5. Alignment and Uniformity

We follow the protocol of [49] but training the network 100 epochs on ImageNet-100. We calculate the alignment and uniformity based on [90]. The alignment loss $\mathcal{L}_{\text{align}}$ and uniformity loss $\mathcal{L}_{\text{uniform}}$ are computed as follows:

$$\mathcal{L}_{\text{align}}(\mathbf{x}, \mathbf{y}) = \mathbb{E}_{(\mathbf{x},\mathbf{y})\sim p_{\text{data}}} \left[\|f_q(\mathbf{x}) - f_k(\mathbf{y})\|_2^{\alpha}\right] \tag{12}$$

$$\mathcal{L}_{\text{uniform}}(\mathbf{x}) = \log \mathbb{E}_{\mathbf{x},\mathbf{y}\sim p_{\text{data}}} \left[\exp(-t\|f_q(\mathbf{x}) - f_k(\mathbf{y})\|_2^2)\right] \tag{13}$$

where $\mathbf{x}$ and $\mathbf{y}$ is a pair of positive images, 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[5].

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

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

Table 6. **List of classes from ImageNet-100.** These classes are randomly sampled from the original ImageNet ILSVRC-2012 dataset.

9.6. ImageNet-100 Subsets

The list of classes from ImageNet-100[6] is randomly sampled from the original ImageNet ILSVRC-2012 dataset and is the same as that used in [81]. The list is shown in Table 6.

9.7. Image Augmentations

During self-supervised training, SynCo uses the same augmentation as [17]. The augmentation parameters are detailed in Table 7.

10. Additional Results

In this section we present extended results starting with object detection on PASCAL VOC, where SynCo demonstrates faster training and matches MoCo-v2's performance at 800 epochs. We then analyze the representations learned by SynCo through multiple perspectives. We examine the model's feature space using class concentration metrics, dimensionality reduction techniques (t-SNE, UMAP), and nearest neighbor analysis to understand its semantic organi-

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

Parameter	$\mathcal{T}$
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

Table 7. **Parameters used to generate image augmentations.** This table lists the augmentation probabilities and intensity settings used for MoCo-v2 pretraining.

zation. We investigate the robustness of learned representations under distribution shifts (ImageNet variants), corruptions, and adversarial attacks to assess their generalization capabilities. Through GradCAM visualizations, we also provide insights into which image regions contribute most to the model's feature extraction. These analyses collectively demonstrate SynCo's ability to learn discriminative and robust visual representations.

10.1. Comparison with State-of-the-Art Methods

We present a comprehensive comparison with state-of-the-art self-supervised learning methods in Table 8, including instance discrimination methods (SimCLR [14], PIRL [67]), momentum-based (BYOL [35], MoCo variants), clustering-based (SwAV [10]), redundancy reduction (Barlow Twins [100], VICReg [7]), and various hard negative mining strategies (MoCHI [49], DCL [98], AdCo [48]). It is important to note that methods such as BYOL [35], Barlow Twins [100], SwAV [10], DINO [11], SimCLR-v2 [15], AdCo [48], and VICReg [7] incorporate additional architectural and training *tricks*, including larger projection heads, significantly larger projection dimensions (*e.g.*, DINO with 65k dimensions, Barlow Twins with 8k dimensions compared to our 128 dimensions), multi-crop augmentation strategies [10], and extended training schedules. While these modifications improve performance, they stem from orthogonal architectural choices rather than from core learning mechanisms alone. Therefore, the most fair and direct comparison is against MoCo-based approaches (MoCo [42], MoCo-v2 [17], MoCHI [49], PCL [56], DCL [98]), which share similar architectural choices, projection dimensions, and training procedures, ensuring an equitable evaluation of our contributions to hard negative mining. For more details, see Section 13.

Method	Epochs	Top-1
Supervised	200	76.5
RotNet [33]	200	48.9
PIRL [67]	200	63.6
LA [106]	200	60.2
CMC [81]	200	60.0
InfoMin [82]	200	70.1
InfoMin [82]	800	73.0
SimSiam [16]	100	68.3
MSF [53]	200	72.4
ReSSL [105]	200	69.9
ReSSL [105]‡	200	74.7
AdCo [48]	200	68.6
AdCo [48]‡	800	**75.7**
SimCLR + DCL [98]	200	65.8
SimCLR + DCLW [98]	200	66.9
SimCLR [14]	1000	69.3
BYOL [35]	1000	74.3
W-MSE [26]	400	72.5
DINO [11]‡	800	<u>75.3</u>
SwAV [10]	800	71.8
SwAV [10]‡	800	<u>75.3</u>
Barlow Twins [100]	1000	73.2
ReLIC [68]	1000	70.3
UniGrad [80]	1000	70.3
VICRegL [8]	300	70.4
VICReg [7]	1000	73.2
CLSA [91]	800	72.2
Mixed Barlow Twins [5]	800	72.2
CaCo [92]	800	74.1
All4One [27]	800	66.6
MoCo-based		
MoCo [42]	200	60.7
PCL-v1 [56]	200	61.5
MoCo-v2 [17]	200	67.5
MoCo-v2 [17]	800	<u>71.1</u>
MoCHI [49]	200	66.9
MoCHI [49]	800	68.7
PCL-v2 [56]	200	67.6
MoCo-v2 + DCL [98]	200	67.6
MoCo-v2 + NS [31]	200	67.9
SynCo (ours)	200	68.1
SynCo (ours)	800	70.7
SynCo† (ours)	800	**71.6**

Table 8. **State-of-the-art linear evaluation on ImageNet ILSVRC-2012.** Top-1 accuracy (in %) for all methods, including extended epochs. Epochs indicate pretraining duration. The highest accuracy in each column is **bolded** and the second highest is <u>underlined</u>. **Symbols**: ‡ With multi-crop augmentation. † We stop generating synthetic negatives at epoch 400.

Method	Epochs	AP	AP_{50}	AP_{75}
Supervised	90	53.5	81.3	58.8
MoCo [42]	200	55.9	81.5	62.6
MoCo-v2 [17]	200	57.0	82.4	63.6
MoCo-v2 [17]	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**

Table 9. **Results for object detection on PASCAL VOC.** The values in bold indicate the maximum of each column.

10.2. 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 9. 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.

10.3. Linear Evaluation

Table 10 shows the progression of linear evaluation accuracy over training epochs, comparing MoCo-v2, SynCo, and SynCo† (which stops generating synthetic negatives after epoch 400). The results demonstrate that SynCo achieves faster convergence in the early training stages compared to MoCo-v2, reaching higher accuracy with fewer epochs. This initial acceleration can be attributed to the synthetic hard negatives providing more informative gradient signals that help the model learn discriminative features more efficiently.

Performance plateau. However, a critical observation emerges in the later stages of training: while standard SynCo continues to generate synthetic negatives throughout the entire 800-epoch training process, its performance begins to plateau and even slightly decline after epoch 400. In contrast, SynCo†, which stops generating synthetic negatives after epoch 400, shows the best final performance. This phenomenon can be explained through the lens of proxy task difficulty modulation.

Early vs. late stage training. As training progresses, the model becomes increasingly proficient at distinguishing between positive and negative pairs, making the original contrastive task easier. However, SynCo continues to generate synthetic hard negatives that are specifically designed to be challenging, effectively maintaining or even increasing the difficulty of the proxy task. While this sustained difficulty can be beneficial in early training stages—preventing the model from prematurely converging to suboptimal solutions—it becomes counterproductive in later stages when the model needs to consolidate and refine its learned representations.

Cooldown. The continued generation of synthetic hard negatives in later epochs creates an overly challenging proxy task that may force the model to focus on increasingly subtle and potentially noisy distinctions rather than learning robust, generalizable features. This aligns with our analysis in the main paper, which shows that SynCo consistently achieves lower proxy task accuracy (indicating higher difficulty) compared to individual synthetic negative types. The SynCo† variant effectively provides a "cooling down" period where the model can stabilize its representations without the additional challenge of synthetic negatives, leading to better downstream performance. This finding suggests that dynamically modulating the difficulty of contrastive learning through controlled synthetic negative generation—rather than maintaining constant difficulty—is crucial for optimal representation learning.

10.4. Fine-tuning

We also evaluate SynCo's performance when fine-tuning with 100% of the labeled ImageNet data. As shown in Table 11, SynCo demonstrates consistent improvements over MoCo-based methods across all training data fractions. With the full dataset (100% labels), SynCo achieves 79.0% top-1 accuracy, outperforming MoCo-v2 (77.0%) by **+2.0%** and MoCHI (78.0%) by **+1.0%**. When stopping synthetic negative generation, we achieve even better results, *i.e.*, 79.9% top-1 accuracy. This comprehensive evaluation across 1%, 10%, and 100% of labeled data demonstrates that SynCo's synthetic hard negatives provide robust improvements regardless of the amount of available supervision, with particularly pronounced benefits in low-data regimes where the quality of learned representations becomes even more critical.

10.5. 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 5 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 [30]).

As shown in Table 12, 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 bet-

Method	100	200	300	400	500	600	700	800
MoCo-v2 [17]	64.2	67.5	68.5	69.2	70.3	70.6	70.9	71.1
SynCo (ours)	65.8	68.1	68.7	69.7	70.3	70.4	70.6	70.7
SynCo† (ours)	**65.8**	**68.1**	**68.7**	**69.7**	**70.5**	**70.8**	**71.3**	**71.6**

Table 10. **Top-1 accuracy (%) during pretraining from 100 to 800 epochs.** Comparison of MoCo-v2, SynCo, and SynCo† (synthetic negatives disabled after 400 epochs).

Method	1%	10%	100%
MoCo-v2 [17]	48.2	66.1	77.0
MoCHI [49]	50.4	65.7	78.0
SynCo (ours)	50.8	66.6	79.0

Table 11. **Top-1 accuracy under different fine-tuning data fractions.** We compare MoCo-v2, MoCHI, and SynCo when fine-tuned on 1%, 10%, and 100% of ImageNet.

Method	Epochs	Mean ↑	Median ↑	Std ↓
Supervised	90	1.381	**1.369**	**0.110**
MoCo [42]	200	1.012	0.999	0.115
MoCo-v2 [17]	200	1.061	0.971	0.358
MoCo-v2 [17]	800	1.146	1.043	0.375
PCL-v1 [56]	200	0.930	0.869	0.312
PCL-v2 [56]	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

Table 12. **Statistical summary of the ratio between inter-class and intra-class distances.** ↑ indicates higher is better, ↓ indicates lower is better. Higher mean indicates better class separation, while lower standard deviation suggests more consistent feature learning across different classes.

ter 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 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

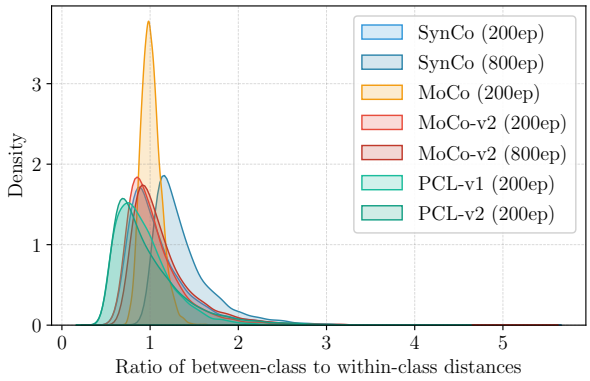


Figure 5. **Distribution of the ratio between inter-class and intra-class distances** for different MoCo-based methods. Higher values indicate better class separation. We show MoCo [42], MoCo-v2 [17] (200 and 800 epochs), PCL-v1 and PCL-v2 [56] (200 epochs), and SynCo (200 and 800 epochs).

on downstream tasks, particularly in scenarios requiring fine-grained discrimination between similar classes. By focusing exclusively on methods built upon the MoCo framework, this comparison ensures a fair evaluation of SynCo's contributions to contrastive learning.

10.6. Robustness and Out-of-Distribution Evaluation

Datasets. We evaluate the robustness and out-of-distribution (OOD) generalization capabilities of SynCo representations. For robustness evaluation, we employ four datasets: ImageNet-v2 [72], which comprises three sets of 10,000 images (matched frequency, threshold 0.7, and top images); ImageNet-A [46], which contains naturally adversarial examples; ImageNet-C [44], which consists of 15 synthetically generated corruptions (*e.g.*, blur, noise, weather); ImageNet-Watermark [58], testing robustness to watermark perturbations. For OOD generalization, we examine performance on five datasets: ImageNet-Sketch [88], containing 50,000 black-and-white sketches; ImageNet-R [45], consisting of 30,000 artistic renditions; ImageNet-O [46], designed for anomaly detection (evaluated using FPR95).

Evaluation protocol. On all datasets, we evaluate the representations of a standard ResNet-50 encoder under a linear

evaluation protocol, where we freeze the pretrained representations and train a linear classifier using the labeled ImageNet training set. The test evaluation is performed zero-shot, i.e., no training is done on the above datasets.

Results and analysis. As shown in Table 13, SynCo demonstrates strong robustness across various distribution shifts, outperforming MoCo and SimCLR in most robustness benchmarks. At 200 epochs, SynCo achieves better results than MoCo and is competitive with MoCo-v2, particularly on ImageNet-C (51.6% top-1 accuracy) and ImageNet-A (3.2% top-1 accuracy). At 800 epochs, SynCo achieves comparable performance to MoCo-v2 across robustness and OOD benchmarks, while surpassing SimCLR on OOD datasets such as ImageNet-Sketch (19.2% top-1 accuracy) and ImageNet-R (28.7% top-1 accuracy). Table 14 further highlights SynCo's strong performance across all corruption categories in ImageNet-C, including noise, blur, weather, and digital corruptions. SynCo consistently outperforms MoCo across these categories, demonstrating its ability to maintain high accuracy under a wide range of corruptions. At 800 epochs, SynCo achieves similar performance to MoCo-v2.

10.7. Adversarial Robustness

Attack methods. We evaluate the adversarial robustness of SynCo by testing against a comprehensive suite of adversarial attacks. Following standard practices in adversarial machine learning [65], we assess model performance against both white-box and black-box attacks on the ImageNet validation set. All attacks are implemented using the torchattacks library [51][7], with evaluations conducted using a ResNet-50 backbone.

Evaluation protocol. Our evaluation includes gradient-based attacks: Fast Gradient Sign Method (FGSM) [34] with $\varepsilon = 8/255$, and Projected Gradient Descent (PGD) [65] with $\varepsilon = 8/255$, $\alpha = 2/255$, and 10 steps. We also evaluate against optimization-based attacks: Carlini & Wagner (C&W) [9] with confidence $\kappa = 0$, 50 optimization steps, learning rate of 0.01, and initial constant $c = 10^{-4}$. Additionally, we test black-box attacks, including score-based and decision-based methods: Square Attack [1] with ℓ_∞ norm and 1,000 queries, and Auto Attack [20] using ℓ_∞ norm. Furthermore, we include advanced perturbation methods: Translation-Invariant FGSM (TIFGSM) [24] with $\varepsilon = 8/255$, $\alpha = 2/255$, and 10 steps, and One-Pixel Attack [79] limited to single-pixel modifications with 10 steps.

Results and analysis. Results in Table 15 highlight SynCo's strong adversarial robustness across a diverse set of attacks. At 200 epochs, SynCo outperforms MoCo and MoCo-v2 on clean accuracy (68.13%) and demonstrates higher resilience against FGSM (24.70%) and PGD (0.33%) attacks, reflecting its ability to withstand gradient-based perturbations better. Furthermore, SynCo achieves comparable results to MoCo-v2 on optimization-based attacks like C&W (17.87%) and Square Attack (14.73%), while surpassing MoCo in all categories. At 800 epochs, SynCo continues to exhibit competitive performance, achieving parity with MoCo-v2 on clean accuracy (70.72% vs. 71.06%) and similar or slightly better robustness to most attacks.

[7] Available at: https://github.com/Harry24k/adversarial-attacks-pytorch.

10.8. Extending SynCo to Vision Transformers

In this section, we describe how we extend SynCo to vision transformers [25]. While SynCo was originally tested for convolutional architectures (specifically ResNet-50), its core principle of enhancing contrastive learning through synthetic hard negatives is architecture-agnostic. To adapt SynCo for vision transformers, we integrate our synthetic negative generation approach into the MoBY framework [96]. We chose MoBY over MoCo-v3 [18] due to computational constraints, as MoCo-v3 requires prohibitively large batch sizes (4096) which were not feasible with our available resources. We follow the established MoBY protocol, which is particularly designed for self-supervised learning with vision transformers. Specifically, both the encoder f_q and f_k consist of a backbone (DeiT-Small [84] or Swin-Tiny [62]) and a projection head [14]. The encoder f_q has an additional prediction head following [35]. The encoder f_k is updated by the moving average of f_q. Similar to the original MoBY implementation, we maintain a dual-encoder architecture, asymmetric network updates through momentum [35], and a queue-based contrastive approach [42]. The primary difference in our implementation is the incorporation of synthetic hard negatives generated in feature space, which creates more challenging examples for the model during training.

Implementation details. For our full ImageNet experiments, we selected the configuration with $N = 256$ hardest negatives and $N_i = 128$, for $i = 1, \ldots, 6$, which offered the best balance of performance and stability. We maintain most of the training hyperparameters from MoBY, including AdamW [64] optimizer with base learning rate of 0.03, weight decay of 10^{-4}, batch size of 512, temperature parameter $\tau = 0.2$, queue size $K = 4096$, and starting momentum $m_{\text{start}} = 0.99$ with cosine schedule. For longer pretraining regimes (like our 300-epoch training on ImageNet), we implement a "cooldown" period for the last 100 epochs where no synthetic negatives are generated. This helps stabilize learning as the model approaches convergence, preventing the learning task from becoming too difficult in later stages.

Method	Epochs		Robustness						Out-Of-Distribution		
		IN	MF	T-0.7	TI	IN-C	IN-A	IN-W	IN-S	IN-R	IN-O
Supervised	90	76.1	63.1	72.3	77.6	39.8	0.0	48.7	24.1	36.2	81.4
SimCLR [14]	1000	69.3	53.2	61.7	68.0	31.1	-	-	3.9	18.3	-
MoCo [42]	200	60.9	45.9	53.8	60.4	33.8	2.5	38.5	10.2	18.2	85.9
MoCo-v2 [17]	200	67.8	54.8	63.0	69.0	51.4	2.8	44.2	17.5	27.8	81.9
MoCo-v2 [17]	800	71.1	58.5	66.6	73.0	55.8	4.1	35.0	19.2	29.7	79.0
SynCo (ours)	200	68.1	54.9	63.7	69.8	51.6	3.2	42.8	16.5	26.7	82.5
SynCo (ours)	800	70.7	58.1	66.4	72.5	55.9	4.2	41.6	19.2	28.7	79.5

Table 13. **Top-1 accuracy (in %) across different ImageNet variants.** We use ResNet-50 as the backbone, except ImageNet-O (IN-O) where we evaluate using FPR95. **Abbreviations legend:** IN: ImageNet; MF/T07/TI: ImageNet-v2 variants; IN-C: ImageNet-C; IN-A: ImageNet-A; IN-S: ImageNet-Sketch; IN-R: ImageNet-R; IN-W: ImageNet-Watermark. Results for SimCLR are from [83]. We reproduce MoCo and MoCo-v2 linear probing since no checkpoints are available (thus results may differ from original implementation).

Method	Epochs	Noise			Blur				Weather				Digital			
		Gauss	Shot	Imp	Defoc	Glass	Mot	Zoom	Frost	Snow	Fog	Bright	Cont	Elas	Pix	JPEG
Supervised	90	32.9	30.5	28.6	35.3	25.3	36.2	36.2	34.9	30.1	42.9	65.0	35.7	42.9	45.6	53.0
SimCLR [14]	1000	29.1	26.3	17.3	22.1	14.7	20.0	18.6	27.2	33.3	46.2	59.7	53.9	31.0	24.2	43.9
MoCo [42]	200	29.9	26.5	10.2	26.1	24.3	33.0	20.7	32.4	25.2	28.1	52.2	47.0	43.3	35.8	40.3
MoCo-v2 [17]	200	51.8	50.2	36.3	48.2	44.1	50.4	36.1	50.2	40.4	44.8	63.7	58.1	58.1	58.1	52.9
MoCo-v2 [17]	800	56.2	54.8	39.9	52.6	48.7	58.1	40.1	53.9	45.6	51.4	67.1	62.1	61.9	61.7	56.7
SynCo (ours)	200	52.3	50.9	34.8	48.8	44.7	51.3	36.5	49.7	39.9	44.0	63.7	58.3	58.3	58.3	52.8
SynCo (ours)	800	57.5	56.3	40.9	53.1	49.7	57.3	41.6	53.6	44.9	49.8	66.8	62.0	61.9	61.0	55.8

Table 14. **Top-1 accuracy (%) for ImageNet-C corruption results.** We use ResNet-50 as the backbone. ImageNet-C: noise (gaussian, shot, impulse), blur (defocus, glass, motion, zoom), weather (frost, snow, fog, brightness), digital (contrast, elastic, pixelate, jpeg). We reproduce MoCo and MoCo-v2 linear probing since no checkpoints are available.

Method	Epochs	Clean	FGSM	PGD	C&W	Square	Auto	TIFGSM	OnePixel
Supervised	90	76.15	23.47	0.28	16.19	11.52	0.21	4.37	73.79
MoCo [42]	200	60.86	15.75	0.08	9.40	9.98	0.05	7.33	57.40
MoCo-v2 [17]	200	67.77	23.75	0.32	17.08	13.94	0.23	5.36	65.16
MoCo-v2 [17]	800	71.06	30.79	0.53	22.99	18.89	0.34	8.29	68.69
SynCo (ours)	200	68.13	24.70	0.33	17.87	14.73	0.24	6.24	65.71
SynCo (ours)	800	70.72	31.67	0.48	22.90	18.66	0.34	8.00	68.45

Table 15. **Top-1 accuracy (in %) under various adversarial attacks on ImageNet validation set.** We use ResNet-50 as the backbone. We reproduce MoCo and MoCo-v2 linear probing since no checkpoints are available.

Results and analysis. As shown in Table 16, our method achieves consistent improvements over the MoBY baseline across both DeiT-S [84] and Swin-T [62] architectures, while also outperforming other self-supervised approaches like DINO [11] and MoCo-v3 [18]. This demonstrates the effectiveness of synthetic hard negatives in enhancing representation learning for vision transformers, proving that the approach transfers successfully from convolutional to transformer-based architectures. While there remains a gap compared to supervised learning, our results show that the synthetic negative technique is a simple yet effective enhancement to existing self-supervised learning frameworks for vision transformers.

10.9. Class Average t-SNE Visualization

We examine the distribution of ImageNet concepts in SynCo's feature space. For each ImageNet class, we compute the average feature vector from its validation images. We apply t-SNE [86] with a perplexity of 30 and learning rate of 200 for 1000 iterations. Figure 18 and Figure 19 reveal that SynCo learns meaningful semantic structures: similar animal species naturally cluster together, *e.g.*, spider, barn

Method	Arch.	Epochs	Top-1
Supervised	DeiT-S	300	79.8
Supervised	Swin-T	300	81.3
DINO [11]	DeiT-S	300	72.5
MoCo-v3 [18]	DeiT-S	300	72.5
MoBY [96]	DeiT-S	300	72.8
MoBY [96]	Swin-T	300	75.0
SynCo (ours)	DeiT-S	300	73.0
SynCo (ours)	Swin-T	300	75.2

Table 16. **Linear evaluation on ImageNet ILSVRC-2012 using vision transformers.** Top-1 accuracy (in %) for methods using vision transformers as encoders, pretrained for 300 epochs. Results for SynCo are based on 1 run.

spider, garden spider, tarantula, wolf spider, and black widow cluster together (bottom right), while digital clock, digital watch, and dial telephone form another coherent group (right mid). The visualization at 800 epochs (Figure 19) shows coherent clusters as well, where *e.g.*, Yorkshire terrier, silky terrier, and Australian terrier cluster together (right mid). We also perform the same analysis for MoCo [42] with 200 epochs of pretraining (Figure 20) and MoCo-v2 [17] with both 200 epochs (Figure 21) and 800 epochs (Figure 22) of pretraining for comparison. Additionally, we include the results from a supervised model trained on ImageNet for comparison (Figure 23).

10.10. GradCAM Visualization

To gain deeper insights into the regions SynCo focuses on during feature extraction, we utilize GradCAM [78] to visualize the model's attention. Attention maps are generated from the final residual block of the ResNet-50 backbone. Figure 6 presents GradCAM visualizations for various ImageNet validation images, comparing SynCo pretrained for 200 epochs and 800 epochs alongside supervised models. The heatmaps reveal that SynCo effectively attends to discriminative object parts and regions, demonstrating its ability to learn meaningful semantic features without supervision.

10.11. UMAP Visualization

To better understand the feature representations learned by SynCo, we perform Uniform Manifold Approximation and Projection (UMAP) [66] on feature embeddings extracted from the validation set. UMAP reduces high-dimensional data to two dimensions, allowing for a qualitative evaluation of class separability. We considered three configurations based on the number of classes: the first 40, the first 100, and all 1000 classes from ImageNet. Figures 12 and 13 illustrate the results for models pretrained for 200 and 800 epochs, respectively. We also present UMAP visualizations for MoCo with 200 epochs of pretraining (Figure 14) and MoCo-v2 with both 200 epochs (Figure 15) and 800 epochs (Figure 16) of pretraining for comparison. For comparison, we also include UMAP visualizations of features from a supervised model trained on ImageNet (Figure 17).

10.12. Nearest Neighbor Retrieval

To analyze the semantic consistency of SynCo's learned representations, we perform nearest neighbor retrieval using the following process. We extract 2048-dimensional feature vectors from both ImageNet training and validation sets using the pretrained ResNet-50 backbone with the classification layer removed, applying average pooling to the final convolutional outputs. Using these embeddings, we randomly select query images from the validation set and find their nearest neighbors from the training set memory bank using cosine distance. Since the nearest neighbor is typically the same image in the memory bank, we analyze neighbors $\#2$ through $\#6$. Results shown in Figure 7 demonstrate how SynCo effectively clusters semantically similar images after 200 and 800 epochs of pretraining. We observe that the retrieved neighbors share similar semantic concepts, textures and object poses with the query image.

11. Ablation Studies

In this section, we perform ablation studies of SynCo on ImageNet-100 and CIFAR-100. For ImageNet we use a ResNet-50, while for CIFAR-100 we use a modified ResNet-18. We compare our method with the baseline of MoCo-v2, showing how SynCo improves performance through synthetic hard negatives. Our experiments analyze the impact of different negative types, hyperparameter sensitivity, and queue size variations. We did not search for the optimal combination of negative types, instead opting to ablate each type individually and all together, as even without considering hyperparameters, testing all possible combinations of the 6 negative types would require evaluating 63 different configurations ($2^6 - 1$), which would be computationally prohibitive.

11.1. Ablation Study on ImageNet-100

First, we perform ablations studies on ImageNet-100 for 100-way classification. Specifically, we ablate SynCo's hyperparameters σ, δ, η, types (1 to 6), and the effect of queue size K to pretraining. The results of our ablations are presented in Tables 17 to 19.

Ablation on hyperparameters. We conducted ablations on the parameters σ, δ, and η of SynCo's type 4, type 5, and type 6 negatives, respectively. The results, presented in Table 17, show that varying these parameters does not lead to significant differences in performance. This suggests that SynCo is robust across a wide range of values for σ, δ, η.

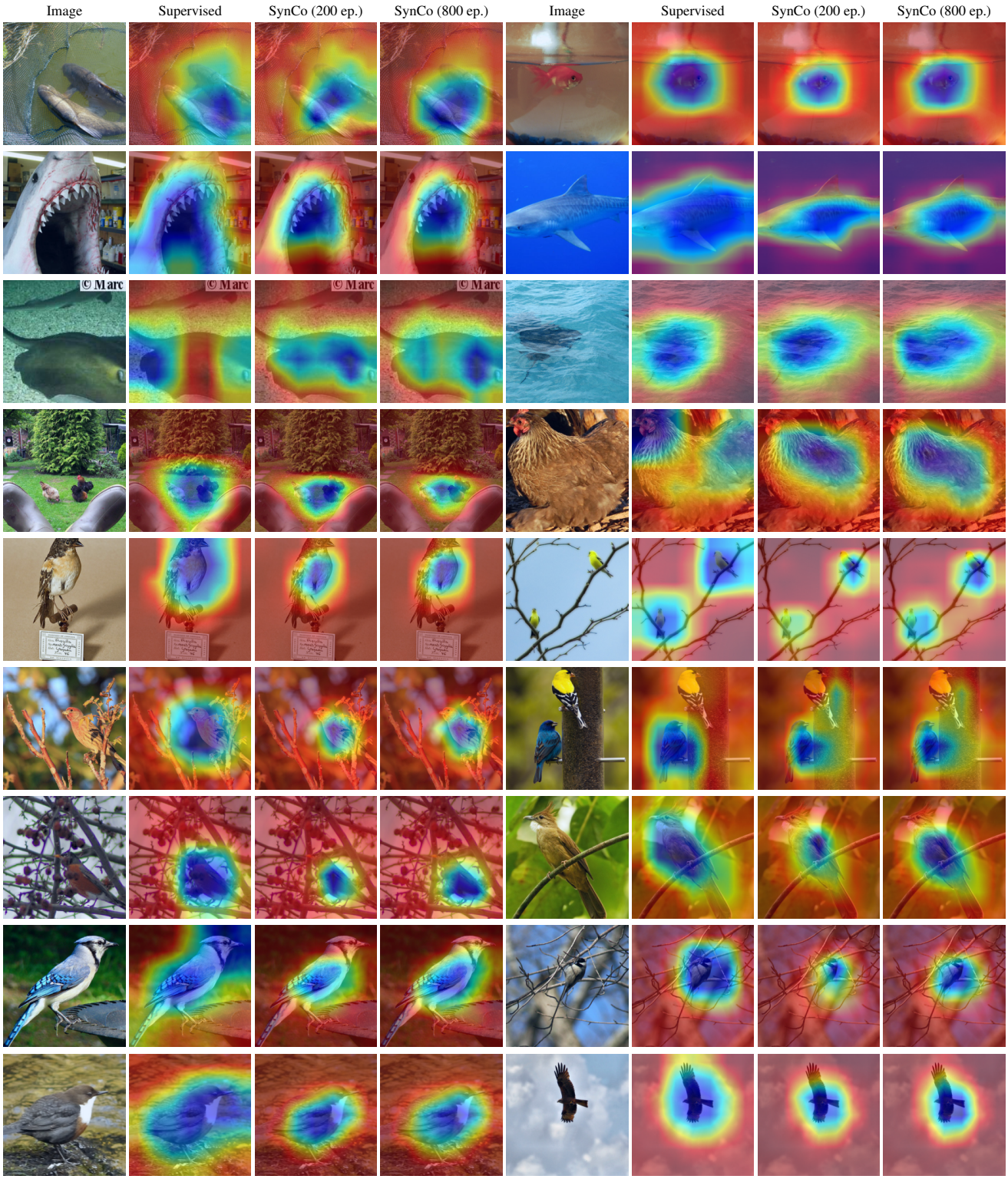


Figure 6. **GradCAM visualizations of ImageNet validation set.** We compare different models: supervised training and SynCo pretrained for 200 and 800 epochs.

Figure 7. **Visualization of nearest neighbors in the embedding space.** SynCo is pretrained at 200 epochs (**left**) and 800 epochs (**right**). Each row corresponds to a query image and its top-5 nearest neighbors in the respective embedding spaces.

	Value	Top-1	Top-5
σ	0.01	48.20	74.26
	0.05	48.36	73.84
	0.10	47.62	73.50
δ	0.01	48.12	74.46
	0.05	48.88	74.72
	0.10	48.04	73.72
η	0.01	48.06	74.00
	0.05	47.16	74.14
	0.10	47.76	74.06

Table 17. **Ablation study on ImageNet-100 for SynCo's hyperparameters σ, δ, and η.** Top-1 and top-5 accuracies (in %) under linear evaluation with 100 epochs of pretraining using ResNet-50. We highlight the default hyperparameter.

S^1	S^2	S^3	S^4	S^5	S^6	Top-1	Top-5
✗	✗	✗	✗	✗	✗	47.7	73.9
✓	✗	✗	✗	✗	✗	48.2	73.9
✗	✓	✗	✗	✗	✗	48.2	74.1
✗	✗	✓	✗	✗	✗	48.2	73.8
✗	✗	✗	✓	✗	✗	48.2	74.2
✗	✗	✗	✗	✓	✗	48.1	74.4
✗	✗	✗	✗	✗	✓	48.0	74.0
✓	✓	✗	✗	✗	✗	48.1	73.9
✓	✗	✓	✗	✗	✗	46.8	72.9
✗	✓	✓	✗	✗	✗	48.2	74.1
✗	✓	✗	✗	✓	✗	46.3	74.1
✓	✓	✓	✗	✗	✗	48.3	74.2
✗	✗	✗	✓	✓	✓	48.3	74.1
✓	✓	✓	✓	✓	✓	**48.4**	**74.5**

Table 18. **Ablation study on ImageNet-100 for the synthetic negative types.** We convert each configuration into a binary selection over six synthetic negative strategies (S^1–S^6). We highlight the default hyperparameter.

Ablation on types. We evaluate the impact of each synthetic hard negative type on pretraining by selecting the top $N = 1024$ hardest negatives and generating $N_i = 256$ synthetic negatives for each type $i = 1, 2, \ldots, 6$. We systematically evaluate individual types, pairwise combinations, subset groupings, and the complete combination to understand how different synthetic negative generation strategies contribute to representation learning. As we can see from Table 18, all individual synthetic negative types demonstrate improvements over the MoCo-v2 baseline (47.7% top-1 accuracy). The geometric transformation types (Types 1-3) achieve the strongest individual improvements, with interpolated, extrapolated, and mixup negatives all reaching 48.2% top-1 accuracy. Type 4 (noise-injected) matches this performance at 48.2%, while the gradient-based approaches show more modest individual gains—Type 5 (perturbed) at 48.1% and Type 6 (adversarial) at 48.0%. From the pairwise combinations in Table 18, we observe that some combinations like Types 1,2 (48.1%) and Types 2,3 (48.2%) maintain strong performance, while others show degradation—notably Types 1,3 (46.8%) and Types 2,5 (46.3%). This suggests that certain synthetic negative types may interfere with each other when combined inappropriately, potentially due to conflicting optimization signals or redundant hard example generation. As shown in Table 18, when grouped by methodology, geometric transformations (Types 1,2,3) achieve 48.3% top-1 accuracy, while gradient-based methods (Types 4,5,6) reach 48.3% as well, indicating that both geometric and gradient-based approaches provide substantial improvements when properly combined within their respective categories. Our complete SynCo approach, which combines all six synthetic negative types, achieves the best performance at 48.4% top-1 accuracy and 74.5% top-5 accuracy. This represents a +0.7% improvement in top-1 accuracy over MoCo-v2, demonstrating that despite some negative pairwise interactions, the different synthetic negative types provide complementary learning signals that collectively enhance representation learning beyond what individual approaches can achieve.

Method	Queue size K					
	4k	8k	16k	32k	65k	131k
MoCo-v2 [17]	50.10	50.50	49.32	48.02	47.74	47.60
SynCo (ours)	48.30	48.50	49.40	48.08	48.42	48.50

Table 19. **Ablation study on ImageNet-100 for the queue size K.** Top-1 accuracies (in %) under linear evaluation with 100 epochs of pretraining using ResNet-50, comparing MoCo-v2 and SynCo. We highlight the default hyperparameter.

Ablation on queue size. We investigate the effect of queue size $\mathcal{Q}$ on performance. We train SynCo and MoCo-v2 with reduced queue sizes. Our results, presented in Table 19, reveal that SynCo performs comparably to MoCo-v2 across various queue sizes. With smaller queues, SynCo underperforms compared to MoCo-v2. This can be attributed to the fact that the total generated synthetic negatives are too hard for the task and harm performance, a finding that is also observed in [49]. However, as the queue increases, SynCo performs on par with MoCo-v2. At the largest queue size tested, SynCo outperforms MoCo-v2.

11.2. Ablation Study on CIFAR-100

Secondly, we perform additional ablation studies on CIFAR-100 [55] for 32×32 images for 100-way classification, chosen for its computational efficiency while maintaining sufficient complexity for meaningful ablations. We ablate the

same paramters as in Section 11.1, with the addition of N_i, N, and batch size. We use the same settings as previously discussed with the following differences. We adopt a ResNet-18 (512 output units) [40] 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-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 8, indicate that training SynCo with different values of these parameters yields similar performance across all configurations.

Ablation on types. 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 9 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 10, 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 11. 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.

12. Extended Related Work

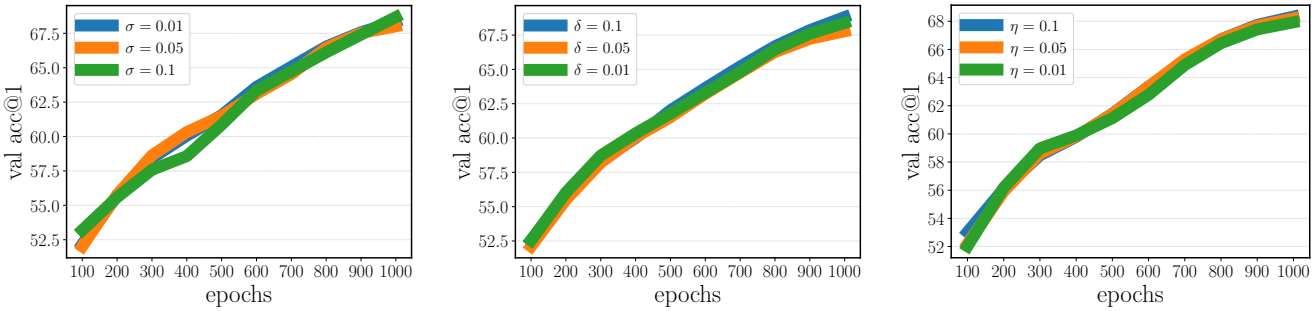
This section extends our related work discussion by examining two complementary approaches in self-supervised learning, *i.e.*, synthetic feature generation, which enhances model performance with limited labeled data, and generative self-supervised methods, particularly Masked Image Modeling (MIM), which learn by reconstructing or predicting parts of input data.

12.1. Synthetic Features

Synthetic feature generation is a widely used method to enhance deep learning models, especially with limited labeled data. Adding synthetic features to the representation space improves model generalization and performance. Some methods generate features for unseen classes using generative models [39, 77, 95], while others integrate these into self-supervised and contrastive learning frameworks [57, 104]. This approach has shown success in zero-shot learning [37]. In contrast, our approach directly generates synthetic hard negatives in contrastive learning, without requiring additional generative models.

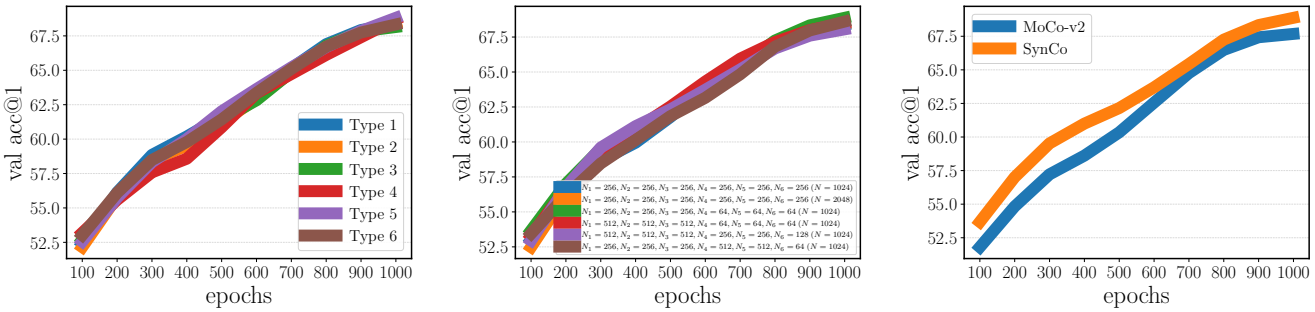
12.2. Generative Self-supervised Learning

While the previously discussed methods are discriminative approaches that learn by comparing and distinguishing between different views or instances, another major branch of self-supervised learning takes a generative approach. Generative methods learn by reconstructing or predicting parts of the input data, with MIM emerging as a particularly successful strategy. iGPT [13] demonstrated early success by treating images as sequences for autoregressive prediction, followed by BEiT [6] and BEiT-v2 [71] which adapted BERT-style [23] masked prediction to vision. MAE [43] showed that aggressive masking of image patches (up to 75%) creates an effective self-supervised task, while SimMIM [97] simplified the approach with a lightweight prediction head. Various improvements followed: MaskFeat [93] predicted HOG features instead of pixels, Context Autoencoder [19] leveraged contextual information, and MSN [3] combined masking with siamese networks. Recent work has focused on efficiency and effectiveness through approaches like SiamMAE [36], MixMAE [60], PixMIM [61], and TinyMIM [74]. The latest developments include CropMAE [29] with efficient siamese cropped autoencoders and ColorMAE [47] exploring data-independent masking strategies. These generative approaches differ fundamentally from discriminative methods by learning to predict or reconstruct missing information rather than comparing different views or instances, offering a complementary approach to self-supervised visual learning.



(a) Ablation on σ (b) Ablation on δ (c) Ablation on η

Figure 8. **Ablation study on CIFAR-100 for hyperparameters σ, δ, and η.** Top-1 accuracy 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.



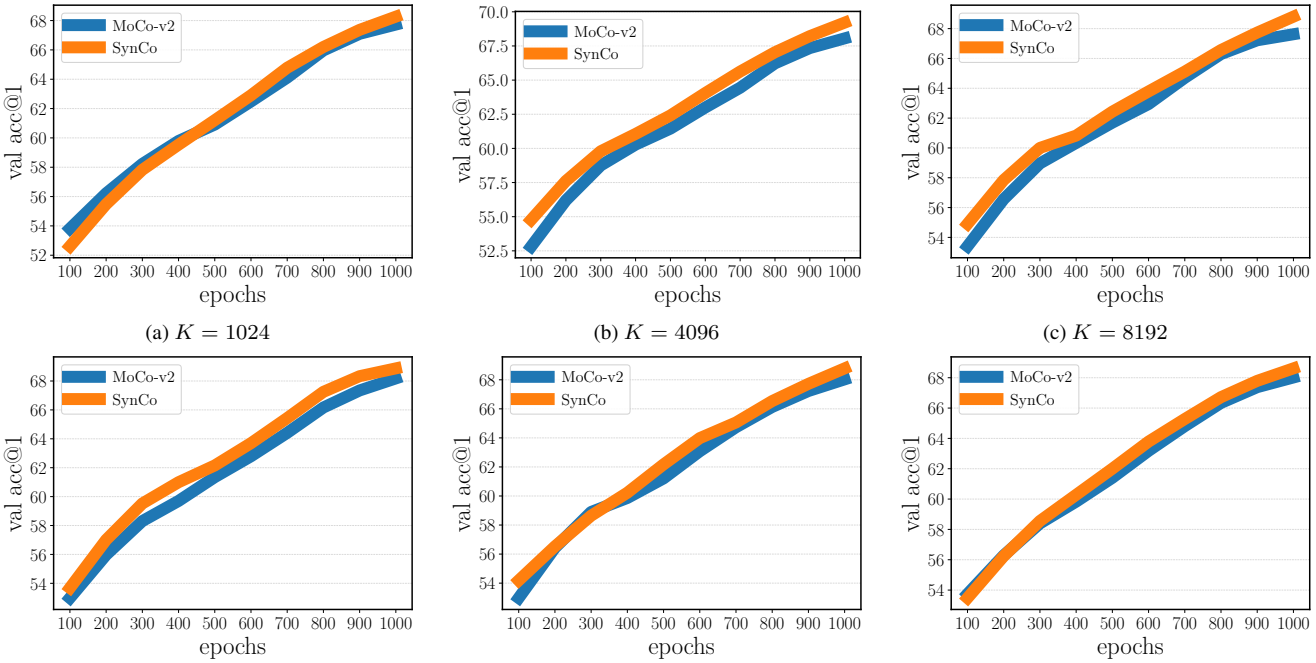
(a) Ablation on types (1-6) (b) Ablation on N_i (N) (c) MoCo-v2 vs SynCo

Figure 9. **Ablation study on CIFAR-100 for synthetic negative types and quantities.** Top-1 accuracy evaluated every 100 epochs over 1000 epochs of training. (a) Performance of SynCo with one type of hard negative at a time. (b) 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 $\hat{\mathcal{Q}}$. (c) Comparison of SynCo without hard negatives (equivalent to MoCo-v2) and with all hard negatives combined.

13. Discussion

In this section, we examine several critical aspects of our work: the rationale behind comparing with MoCo-based approaches rather than other self-supervised methods; the underlying mechanisms of each synthetic hard negative type and their contribution to model generalization; SynCo's role in model regularization and optimization strategies; detailed comparisons with closely related work (MoCHI and AdCo); the broader implications for other domains like text and audio; connections to classical self-supervised learning foundations; limitations in our hyperparameter analysis due to computational constraints; potential extensions to stronger frameworks like Vision Transformers and larger architectures; and possible adaptations to SimCLR's in-batch negative sampling approach. Through this comprehensive discussion, we aim to provide deeper insights into SynCo's effectiveness, limitations, and future research directions while contextualizing our contributions within the broader landscape of self-supervised learning.

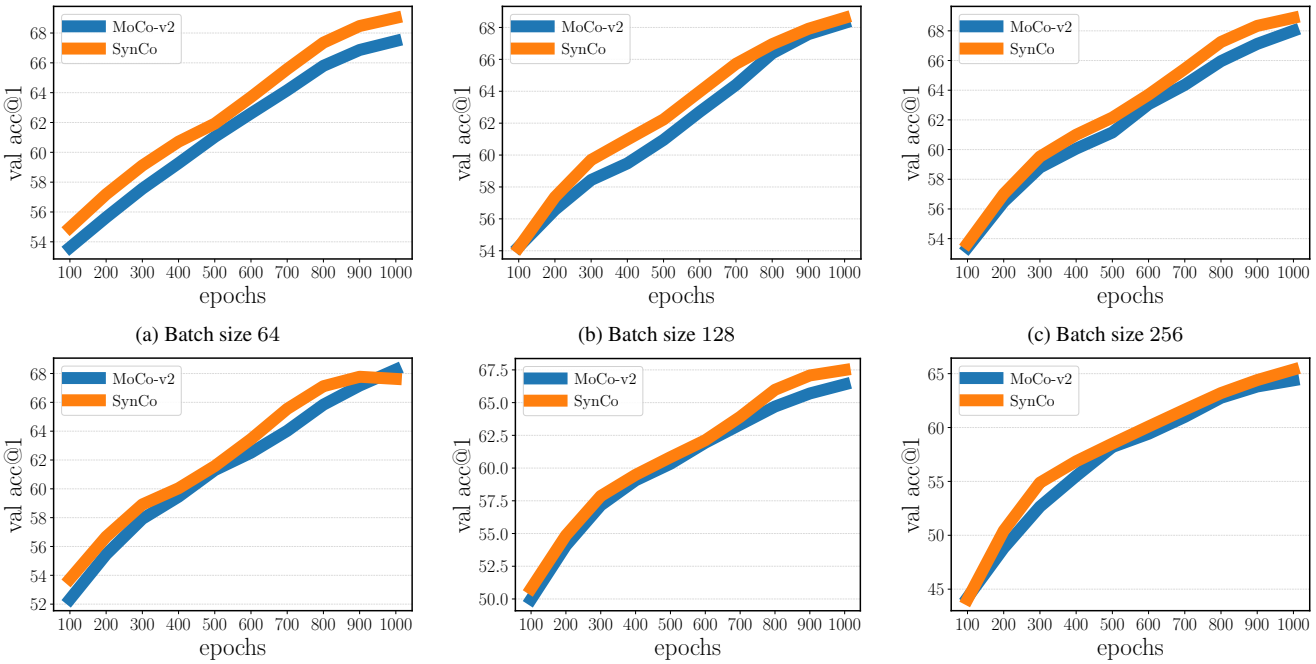
On the fairness of comparisons. Methods such as BYOL [35], Barlow Twins [100], SwAV [10], DINO [11], SimCLR-v2 [15], AdCo [48], and VICRegL [7] incorporate additional *tricks*, including larger projection heads (*e.g.*, DINO, SimCLR-v2), larger projection dimensions (*e.g.*, DINO 65k, Barlow Twins 8k), multi-crop augmentation strategies (SwAV, AdCo), and extended training schedules (BYOL, DINO, *etc.*), which significantly improve their performance. However, these improvements stem from architectural and training modifications rather than solely from their core learning mechanisms. In contrast, our approach focuses on demonstrating the effectiveness of synthetic negative generation within a simpler framework, without relying on such tricks. Therefore, a fair comparison should be made against MoCo-based approaches (MoCo-v2 [17], MoCHI [49], PCL [56], DCL [98]), which share similar architectural choices and training procedures, ensuring an equitable evaluation of our contributions. Additionally, the MoCo-v2 framework is practical for most laboratories for implementation, requiring only 4 GPUs compared to the 8/16+ GPUs that most other



(a) $K = 1024$ (b) $K = 4096$ (c) $K = 8192$

(d) $K = 16384$ (e) $K = 32768$ (f) $K = 65536$

Figure 10. **Ablation study on CIFAR-100 for queue size.** 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 queue size $K = 4096$. (c) With queue size $K = 8192$. (d) With queue size $K = 16384$. (e) With queue size $K = 32768$. (f) With queue size $K = 65536$.



(a) Batch size 64 (b) Batch size 128 (c) Batch size 256

(d) Batch size 512 (e) Batch size 1024 (f) Batch size 2048

Figure 11. **Ablation study on CIFAR-100 for batch size.** 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 batch size of 128. (c) With batch size of 256. (d) With batch size of 512. (e) With batch size of 1024. (f) With batch size of 2048.

methods need, making our contributions more accessible to the broader research community.

Relationship with MoCHI. MoCHI [49] represents the most directly related prior work to our approach, introducing feature-level mixing for hard negative samples in contrastive learning. However, MoCHI's scope is limited to only two synthetic negative strategies: interpolation between query and hard negatives (corresponding to our Type 1), and linear combination of pairs of hard negatives (corresponding to our Type 3). While these strategies provide foundational improvements, they primarily explore geometric transformations within the existing convex hull of hard negatives. SynCo significantly extends this foundation by introducing four additional strategies that explore previously unexplored regions of the representation space. Our extrapolation strategy (Type 2) pushes beyond the decision boundary, creating negatives that are more challenging than any existing samples. Our gradient-based strategies (Types 5 and 6) leverage optimization landscape information to create principled perturbations, while our noise injection strategy (Type 4) introduces controlled stochasticity to prevent overfitting to specific negative patterns. This comprehensive approach ensures that SynCo explores diverse aspects of the representation space, leading to more robust and generalizable features compared to MoCHI's limited geometric transformations, as evidenced by our ablations showing that novel strategies (Types 2, 4-6) alone achieve 48.2/48.3% accuracy compared to MoCHI-equivalent strategies reproduction (Types 1, 3) at 46.8%.

Relationship with AdCo. AdCo [48] takes a fundamentally different approach to hard negative generation by maintaining a set of trainable negative adversaries that are updated through adversarial optimization:

$$\mathbf{v}_k^{(t+1)} = \mathbf{v}_k^{(t)} + \alpha \frac{\partial \mathcal{L}_{\text{adv}}}{\partial \mathbf{v}_k^{(t)}}$$

where $\mathbf{v}_k^{(t)}$ represents the k-th learnable negative adversary at training step t, α is the learning rate for adversarial updates, and $\mathcal{L}_{\text{adv}}$ is the adversarial contrastive loss that is maximized to create more challenging negatives. While both AdCo and SynCo aim to create more challenging negatives, their methodologies differ significantly in computational requirements, implementation complexity, and integration with existing frameworks. AdCo requires a separate adversarial training loop, maintains persistent negative representations throughout training, and relies heavily on multi-crop augmentation that increases computational overhead by approximately 3.7×. Additionally, AdCo's trainable negatives require careful initialization and learning rate scheduling, adding complexity to the training process. In contrast, SynCo generates synthetic negatives on-the-fly without requiring additional training loops, persistent storage, or computationally expensive augmentation strategies. Our approach operates within the standard MoCo framework while providing diverse synthetic negatives through different geometric and gradient-based transformations.

Relationship to data augmentation strategies. While SynCo operates in representation space, it shares conceptual similarities with data augmentation methods like Mixup [102], CutMix [99], and AugMax [89] that operate in pixel space. The key distinction is that SynCo's transformations occur after the encoder, enabling more semantically meaningful negative generation that directly targets hard regions in the learned embedding space. The embedding-space approach also provides computational advantages, as transformations on low-dimensional vectors (*e.g.*, 128-d) are significantly cheaper than pixel-space operations on high-resolution images.

Intuition of SynCo. 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. To illustrate the distinction between gradient-based strategies, consider a concrete example in embedding space: given a horse (anchor) and zebra (hard negative), Type 6 applies fixed-magnitude perturbations ($\eta \cdot \text{sign}(\nabla)$) to push the zebra slightly closer to the horse, while Type 5 applies variable-magnitude perturbations ($\delta \cdot \nabla$) that can range from subtle to significant based on the gradient magnitude, enabling more diverse exploration of the challenging region between these semantically similar animals. Moreover, Type 5 provides significant advantages over interpolation methods by supporting arbitrary similarity functions (*e.g.*, cosine, Jaccard) and generating negatives of varying hardness. The complementary nature of these strategies ensures comprehensive coverage of challenging regions in the representation space, preventing overfitting to specific negative patterns while maintaining appropriate task difficulty.

Theoretical connections to contrastive learning theory. Recent theoretical work [2, 76] provides foundations for

understanding why SynCo works. By generating synthetic hard negatives, we effectively increase the "hardness" of the contrastive task, which theory suggests should improve the quality of learned representations. Our approach aligns with spectral contrastive loss analysis [38], where harder negatives help prevent dimensional collapse and encourage more uniform feature distributions (validated by our uniformity metrics in Figure 3). The gradient-based strategies (Types 5-6) connect to recent work on contrastive learning geometry [90], as they explicitly leverage similarity gradients to explore adversarial regions near decision boundaries. Future theoretical work could formalize the conditions under which synthetic hard negatives provably improve representation quality.

When does SynCo provide the most benefit? Our experiments reveal that SynCo provides the greatest improvements in scenarios where the model can benefit from increased proxy task difficulty. Specifically, SynCo excels in: **(i)** early to mid-training phases (epochs 10-400) where the model is actively learning discriminative features, **(ii)** tasks requiring strong transferability (*e.g.*, detection on COCO shows +1.0% AP^{bb} vs linear evaluation's +0.4%), and **(iii)** low-data regimes (semi-supervised learning with 1% labels shows +2.6% improvement). Conversely, in very late training stages (>400 epochs on ImageNet), excessive proxy task difficulty may hinder convergence, suggesting that adaptive strategies (*e.g.*, stopping generating synthetic hard negatives) are crucial for optimal performance.

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 [12]. This encourages learning robust features over memorization, addressing poor generalization common in empirical risk minimization [87, 102]. 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 [102].

Considerations for parameter tuning. Contrary to concerns about high-dimensional tuning spaces, while SynCo does incorporate multiple hyperparameters across its six synthetic negative generation strategies, the framework is designed to require minimal tuning. The key insight is that reasonable performance can be achieved by setting hyperparameters within logical bounds rather than through exhaustive search. For the perturbation parameters σ, δ, and η, our experiments demonstrate minimal performance variation when these parameters range from 0.01 to 0.1, with default values providing excellent performance (see Section 11.1). Additionally, the hyperparameters α_{max}, β_{max}, and γ_{max} are *empirically* chosen based on the intuition of each method and remain fixed throughout training, eliminating the need for dynamic tuning. For the core architectural hyperparameters, we provide practical guidelines based on extensive ablations. The number of hardest negatives $N = 1024$ follows the principle of maintaining a challenging but not overly restrictive negative pool—smaller values (*e.g.*, 256) would limit diversity while larger values show diminishing returns. The fixed ratios (256 vs 64) for different strategy types are based on empirical observation that geometric transformations benefit from more samples while perturbation-based methods achieve saturation with fewer samples (see Section 11.2).

Implications in broader contexts. Introducing synthetic hard negatives in contrastive learning not only improves performance in image classification and detection but also extends to other modalities such as text, audio, and multimodal tasks. In NLP, generating challenging negatives could benefit sentence similarity, text classification, and translation, while in audio, it may aid speaker recognition and event detection. Their flexibility also creates opportunities in domain adaptation and transfer learning: domain-specific hard negatives can help models generalize better across shifts. Overall, this adaptability suggests synthetic hard negatives may play a key role in building more robust, generalizable machine learning systems.

Bridging classical and modern self-supervised learning. While many foundational works in self-supervised learning date back years [32], their core principles and challenges remain relevant. Our work shows that these frameworks can be strengthened through synthetic hard negative generation, linking classical techniques with modern needs for more efficient and robust representation learning. The method's consistent performance gains across tasks indicate that self-supervised learning—especially when enhanced with synthetic samples in embedding space—continues to provide useful paths for advancing AI. As the field moves toward more generalizable, data-efficient approaches, methods that work well with limited labeled data become increasingly critical.

Limitations on hyperparameter analysis. While our experiments demonstrate SynCo's effectiveness across various configurations, our comprehensive hyperparameter analysis is primarily based on CIFAR-100 (see Section 11.2), with findings extended to ImageNet. Due to computational constraints, we cannot exhaustively ablate these parameters on larger datasets. Nevertheless, our results show that SynCo is remarkably robust to variations in hyperparameters (σ, δ, η, see Section 11.1) and the number of synthetic negatives

(N_i, N, see Section 11.2). This versatility suggests that even without dataset-specific optimization, SynCo can achieve strong performance with default parameters.

Potential extensions. While our current implementation is built on MoCo-v2 [17] for computational efficiency (requiring only 4 GPUs), SynCo's principles could be integrated with more advanced frameworks. Using larger projection and prediction heads [11, 35], incorporating multi-crop augmentation [10], or leveraging newer CNN architectures such as ConvNeXt [63] could further improve performance. However, these improvements typically require substantially more computational resources (> 8 GPUs), making them impractical for our current experimental setup. Future work may explore these extensions as additional resources become available.

Potential extension to SimCLR. While our method is built upon MoCo-v2 [17]'s memory bank, the concept of synthetic hard negatives could be adapted to SimCLR [14]'s in-batch negative sampling approach. Instead of generating synthetic negatives from memory bank features, one could generate them from in-batch features. However, SimCLR typically requires very large batch sizes (4096) and significant GPU resources (> 8 GPUs) to achieve competitive performance, making such an implementation computationally prohibitive for our current experimental validation. This remains an interesting direction for future research.

14. 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 70.7%).

15. 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.

16. Reproducibility Statement

We intend to make a public release of the code repository and pre-trained models to aid the research community in reproducing our experiments. Our implementation of SynCo is built upon the publicly available MoCo [42] codebase. We also provide the pseudocode for generating each type of synthetic negative in Section 8 to further assist in understanding and replication. With minimal additional hyperparameters introduced for synthetic negative generation, we have explicitly detailed these parameters and their values in Section 9 of our paper. We closely follow the experimental protocol of MoCo-v2 [17] to ensure fair comparison. To further support reproducibility, we have made our pretrained model checkpoints publicly available (links provided in Section 14). The code is available at `https://github.com/giakoumoglou/synco`. We believe these resources, combined with the detailed descriptions in our paper, will enable other researchers to replicate our results and build upon our work.

References

[1] Maksym Andriushchenko, Francesco Croce, Nicolas Flammarion, and Matthias Hein. Square attack: a query-efficient black-box adversarial attack via random search. *European Conference on Computer Vision*, pages 484–501, 2020. 7

[2] Sanjeev Arora, Hrishikesh Khandeparkar, Mikhail Khodak, Orestis Plevrakis, and Nikunj Saunshi. A theoretical analysis of contrastive unsupervised representation learning, 2019. 16

[3] Mahmoud Assran, Mathilde Caron, Ishan Misra, Piotr Bojanowski, Florian Bordes, Pascal Vincent, Armand Joulin, Michael Rabbat, and Nicolas Ballas. Masked siamese networks for label-efficient learning, 2022. 13

[4] Philip Bachman, R Devon Hjelm, and William Buchwalter. Learning representations by maximizing mutual information across views, 2019. 2

[5] Wele Gedara Chaminda Bandara, Celso M. De Melo, and Vishal M. Patel. Guarding barlow twins against overfitting with mixed samples, 2023. 4

[6] Hangbo Bao, Li Dong, Songhao Piao, and Furu Wei. Beit: Bert pre-training of image transformers, 2022. 13

[7] Adrien Bardes, Jean Ponce, and Yann LeCun. Vicreg: Variance-invariance-covariance regularization for self-supervised learning, 2022. 4, 14

[8] Adrien Bardes, Jean Ponce, and Yann LeCun. Vicregl: Self-supervised learning of local visual features, 2022. 4

[9] Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. In *2017 IEEE Symposium on Security and Privacy (SP)*, pages 39–57. IEEE, 2017. 7

[10] 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, 4, 14, 18

[11] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers, 2021. 4, 8, 9, 14, 18

[12] 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. 17

[13] Mark Chen, Alec Radford, Rewon Child, Jeffrey Wu, Heewoo Jun, David Luan, and Ilya Sutskever. Generative pretraining from pixels. In *Proceedings of the 37th International Conference on Machine Learning*, pages 1691–1703. PMLR, 2020. 13

[14] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations, 2020. 1, 2, 4, 7, 8, 18

[15] Ting Chen, Simon Kornblith, Kevin Swersky, Mohammad Norouzi, and Geoffrey Hinton. Big self-supervised models are strong semi-supervised learners, 2020. 4, 14

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

[17] Xinlei Chen, Haoqi Fan, Ross Girshick, and Kaiming He. Improved baselines with momentum contrastive learning, 2020. 1, 2, 3, 4, 5, 6, 8, 9, 12, 14, 18, 25, 29, 30

[18] Xinlei Chen, Saining Xie, and Kaiming He. An empirical study of training self-supervised vision transformers, 2021. 7, 8, 9

[19] Xiaokang Chen, Mingyu Ding, Xiaodi Wang, Ying Xin, Shentong Mo, Yunhao Wang, Shumin Han, Ping Luo, Gang Zeng, and Jingdong Wang. Context autoencoder for self-supervised representation learning, 2023. 13

[20] Francesco Croce and Matthias Hein. Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks. In *International Conference on Machine Learning*, pages 2206–2216. PMLR, 2020. 7

[21] Ekin D. Cubuk, Barret Zoph, Jonathon Shlens, and Quoc V. Le. Randaugment: Practical automated data augmentation with a reduced search space, 2019. 2

[22] 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

[23] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2018. 13

[24] Yinpeng Dong, Tianyu Pang, Hang Su, and Jun Zhu. Evading defenses to transferable adversarial examples by translation-invariant attacks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4312–4321, 2019. 7

[25] 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. 7

[26] Aleksandr Ermolov, Aliaksandr Siarohin, Enver Sangineto, and Nicu Sebe. Whitening for self-supervised representation learning, 2021. 4

[27] Imanol G. Estepa, Ignacio Sarasúa, Bhalaji Nagarajan, and Petia Radeva. All4one: Symbiotic neighbour contrastive learning via self-attention and redundancy reduction, 2023. 4

[28] Mark Everingham, Luc Van Gool, Christopher K. I. Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (VOC) challenge. *International Journal of Computer Vision*, 88(2):303–338, 2009. 2

[29] Alexandre Eymaël, Renaud Vandeghen, Anthony Cioppa, Silvio Giancola, Bernard Ghanem, and Marc Van Droogenbroeck. Efficient image pre-training with siamese cropped masked autoencoders, 2024. 13

[30] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer, New York, second edition, 2009. 5

[31] Songwei Ge, Shlok Mishra, Haohan Wang, Chun-Liang Li, and David Jacobs. Robust contrastive learning using negative samples with diminished semantics, 2022. 4

[32] Nikolaos Giakoumoglou, Tania Stathaki, and Athanasios Gkelias. A review on discriminative self-supervised learning methods in computer vision, 2025. 17

[33] Spyros Gidaris, Praveer Singh, and Nikos Komodakis. Unsupervised representation learning by predicting image rotations, 2018. 4

[34] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In *International Conference on Learning Representations*, 2014. 7

[35] 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, 4, 7, 14, 18

[36] Agrim Gupta, Jiajun Wu, Jia Deng, and Li Fei-Fei. Siamese masked autoencoders, 2023. 13

[37] Zongyan Han, Zhenyong Fu, Shuo Chen, and Jian Yang. Contrastive embedding for generalized zero-shot learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2371–2381, 2021. 13

[38] Jeff Z. HaoChen, Colin Wei, Adrien Gaidon, and Tengyu Ma. Provable guarantees for self-supervised deep learning with spectral contrastive loss, 2022. 17

[39] Bharath Hariharan and Ross Girshick. Low-shot visual recognition by shrinking and hallucinating features. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3018–3027, 2017. 13

[40] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015. 1, 13

[41] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn, 2018. 3

[42] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning, 2020. 2, 4, 5, 6, 7, 8, 9, 18, 24, 28

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

[44] Dan Hendrycks and Thomas Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. In *International Conference on Learning Representations*, 2019. 6

[45] Dan Hendrycks, Steven Basart, Norman Mu, Saurav Kadavath, Frank Wang, Evan Dorundo, Rahul Desai, Tyler Zhu, Samyak Parajuli, Mike Guo, et al. The many faces of robustness: A critical analysis of out-of-distribution generalization. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021. 6

[46] Dan Hendrycks, Kevin Zhao, Steven Basart, Jacob Steinhardt, and Dawn Song. Natural adversarial examples. In *CVPR*, 2021. 6

[47] Carlos Hinojosa, Shuming Liu, and Bernard Ghanem. Colormae: Exploring data-independent masking strategies in masked autoencoders, 2024. 13

[48] Qianjiang Hu, Xiao Wang, Wei Hu, and Guo-Jun Qi. Adco: Adversarial contrast for efficient learning of unsupervised representations from self-trained negative adversaries. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1074–1083, 2021. 4, 14, 16

[49] Yannis Kalantidis, Mert Bulent Sariyildiz, Noe Pion, Philippe Weinzaepfel, and Diane Larlus. Hard negative mixing for contrastive learning, 2020. 1, 3, 4, 6, 12, 14, 16

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

[51] Hoki Kim. Torchattacks: A pytorch repository for adversarial attacks. *arXiv preprint arXiv:2010.01950*, 2020. 7

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

[53] Soroush Abbasi Koohpayegani, Ajinkya Tejankar, and Hamed Pirsiavash. Mean shift for self-supervised learning, 2021. 4

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

[55] Alex Krizhevsky. Learning multiple layers of features from tiny images. pages 32–33, 2009. 12

[56] Junnan Li, Pan Zhou, Caiming Xiong, and Steven C. H. Hoi. Prototypical contrastive learning of unsupervised representations, 2021. 4, 6, 14

[57] Wenbin Li, Lei Wang, Jinglin Xu, Jing Huo Liu, Yang Gao, and Jiebo Luo. Generating representative samples for few-shot classification. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13783–13792, 2021. 13

[58] Zhiheng Li, Ivan Evtimov, Albert Gordo, Caner Hazirbas, Tal Hassner, Cristian Canton Ferrer, Chenliang Xu, and Mark Ibrahim. A whac-a-mole dilemma: Shortcuts come in multiples where mitigating one amplifies others, 2023. 6

[59] Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick, and Piotr Dollár. Microsoft coco: Common objects in context, 2015. 2

[60] Jihao Liu, Xin Huang, Jinliang Zheng, Yu Liu, and Hongsheng Li. Mixmae: Mixed and masked autoencoder for efficient pretraining of hierarchical vision transformers, 2023. 13

[61] Yuan Liu, Songyang Zhang, Jiacheng Chen, Kai Chen, and Dahua Lin. Pixmim: Rethinking pixel reconstruction in masked image modeling, 2023. 13

[62] 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. 7, 8

[63] Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A convnet for the 2020s, 2022. 18

[64] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization, 2019. 7

[65] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *International Conference on Learning Representations*, 2018. 7

[66] Leland McInnes, John Healy, and James Melville. Umap: Uniform manifold approximation and projection for dimension reduction, 2020. 9

[67] Ishan Misra and Laurens van der Maaten. Self-supervised learning of pretext-invariant representations, 2019. 4

[68] Jovana Mitrovic, Brian McWilliams, Jacob Walker, Lars Buesing, and Charles Blundell. Representation learning via invariant causal mechanisms, 2020. 4

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

[70] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library, 2019. 1, 25, 31

[71] Zhiliang Peng, Li Dong, Hangbo Bao, Qixiang Ye, and Furu Wei. Beit v2: Masked image modeling with vector-quantized visual tokenizers, 2022. 13

[72] Benjamin Recht, Rebecca Roelofs, Ludwig Schmidt, and Vaishaal Shankar. Do imagenet classifiers generalize to imagenet? In *International Conference on Machine Learning*, 2019. 6

[73] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks, 2016. 3

[74] Sucheng Ren, Fangyun Wei, Zheng Zhang, and Han Hu. Tinymim: An empirical study of distilling mim pre-trained models, 2023. 13

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

[76] Nikunj Saunshi, Jordan Ash, Surbhi Goel, Dipendra Misra, Cyril Zhang, Sanjeev Arora, Sham Kakade, and Akshay Krishnamurthy. Understanding contrastive learning requires incorporating inductive biases, 2022. 16

[77] Edgar Schonfeld, Sayna Ebrahimi, Samarth Sinha, Trevor Darrell, and Zeynep Akata. Generalized zero- and few-shot learning via aligned variational autoencoders. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8247–8255, 2019. 13

[78] Ramprasaath R. Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. *International Journal of Computer Vision*, 128(2):336–359, 2019. 9

[79] Jiawei Su, Danilo Vasconcellos Vargas, and Kouichi Sakurai. One pixel attack for fooling deep neural networks. pages 828–841. IEEE, 2019. 7

[80] Chenxin Tao, Honghui Wang, Xizhou Zhu, Jiahua Dong, Shiji Song, Gao Huang, and Jifeng Dai. Exploring the equivalence of siamese self-supervised learning via a unified gradient framework, 2022. 4

[81] Yonglong Tian, Dilip Krishnan, and Phillip Isola. Contrastive multiview coding, 2020. 3, 4

[82] 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. 4

[83] Nenad Tomasev, Ioana Bica, Brian McWilliams, Lars Buesing, Razvan Pascanu, Charles Blundell, and Jovana Mitrovic. Pushing the limits of self-supervised resnets: Can we outperform supervised learning without labels on imagenet?, 2022. 8

[84] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention, 2021. 7, 8

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

[86] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9 (86):2579–2605, 2008. 8

[87] Vladimir N. Vapnik. *Statistical Learning Theory*. Wiley, 1998. 17

[88] Haohan Wang, Songwei Ge, Zachary Lipton, and Eric P Xing. Learning robust global representations by penalizing local predictive power. In *Advances in Neural Information Processing Systems*, 2019. 6

[89] Haotao Wang, Chaowei Xiao, Jean Kossaifi, Zhiding Yu, Anima Anandkumar, and Zhangyang Wang. Augmax: Adversarial composition of random augmentations for robust training. In *NeurIPS*, 2021. 16

[90] 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. 3, 17

[91] Xiao Wang and Guo-Jun Qi. Contrastive learning with stronger augmentations, 2022. 4

[92] Xiao Wang, Yuhang Huang, Dan Zeng, and Guo-Jun Qi. Caco: Both positive and negative samples are directly learnable via cooperative-adversarial contrastive learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2023. 4

[93] Chen Wei, Haoqi Fan, Saining Xie, Chao-Yuan Wu, Alan Yuille, and Christoph Feichtenhofer. Masked feature prediction for self-supervised visual pre-training, 2023. 13

[94] Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick. Detectron2. https://github.com/facebookresearch/detectron2, 2019. 3

[95] Yongqin Xian, Thomas Lorenz, Bernt Schiele, and Zeynep Akata. Feature generating networks for zero-shot learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5542–5551, 2018. 13

[96] Zhenda Xie, Yutong Lin, Zhuliang Yao, Zheng Zhang, Qi Dai, Yue Cao, and Han Hu. Self-supervised learning with swin transformers, 2021. 7, 9

[97] Zhenda Xie, Zheng Zhang, Yue Cao, Yutong Lin, Jianmin Bao, Zhuliang Yao, Qi Dai, and Han Hu. Simmim: A simple framework for masked image modeling, 2022. 13

[98] Chun-Hsiao Yeh, Cheng-Yao Hong, Yen-Chi Hsu, Tyng-Luh Liu, Yubei Chen, and Yann LeCun. Decoupled contrastive learning, 2022. 4, 14

[99] Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Youngjoon Yoo. Cutmix: Regularization strategy to train strong classifiers with localizable features. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6023–6032, 2019. 16

[100] Jure Zbontar, Li Jing, Ishan Misra, Yann LeCun, and Stéphane Deny. Barlow twins: Self-supervised learning via redundancy reduction, 2021. 1, 4, 14

[101] 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

[102] Hongyi Zhang, Moustapha Cisse, Yann N. Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412*, 2017. 16, 17

[103] 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

[104] Yanzhao Zhang, Cho-Jui Hsieh, and Kai-Wei Wang. Unleashing the power of contrastive self-supervised visual models via contrast-regularized fine-tuning. In *Advances in Neural Information Processing Systems*, pages 2983–2995, 2021. 13

[105] Mingkai Zheng, Shan You, Fei Wang, Chen Qian, Changshui Zhang, Xiaogang Wang, and Chang Xu. Ressl: Relational self-supervised learning with weak augmentation, 2021. 4

[106] Chengxu Zhuang, Alex Lin Zhai, and Daniel Yamins. Local aggregation for unsupervised learning of visual embeddings, 2019. 4

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 coefficient
# t: temperature parameter
# n_hard: number of hard negatives to select
# hyperparameters for synthetic negative generation:
# n1, n2, n3, n4, n5, n6: number of synthetic hard negatives to generate
# sigma: noise standard deviation, epsilon: perturbation scale
# delta: step size for gradient-based, eta: step size for adversarial

f_k.params = f_q.params.copy() # 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

   # compute logits
   l_pos = bmm(q.view(N,1,C), k.view(N,C,1)) # positive logits: Nx1
   l_neg = mm(q.view(N,C), queue.view(C,K)) # negative logits: NxK

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

   # generate all six types of synthetic hard negatives
   s1 = hard_negatives_interpolation(q, idxs_hard) # type 1: interpolated hard negatives: Nxn1
   l_neg_1 = einsum("nc,nkc->nk", [q, s1])
   l_neg = cat([l_neg, l_neg_1], dim=1)

   s2 = hard_negatives_extrapolation(q, idxs_hard) # type 2: extrapolated hard negatives: Nxn2
   l_neg_2 = einsum("nc,nkc->nk", [q, s2])
   l_neg = cat([l_neg, l_neg_2], dim=1)

   s3 = hard_negatives_mixup(q, idxs_hard) # type 3: mixup hard negatives: Nxn3
   l_neg_3 = einsum("nc,nkc->nk", [q, s3])
   l_neg = cat([l_neg, l_neg_3], dim=1)

   s4 = hard_negatives_noise_inject(q, idxs_hard) # type 4: noise-injected hard negatives: Nxn4
   l_neg_4 = einsum("nc,nkc->nk", [q, s4])
   l_neg = cat([l_neg, l_neg_4], dim=1)

   s5 = hard_negatives_perturbed(q, idxs_hard) # type 5: gradient-based hard negatives: Nxn5
   l_neg_5 = einsum("nc,nkc->nk", [q, s5])
   l_neg = cat([l_neg, l_neg_5], dim=1)

   s6 = hard_negatives_adversarial(q, idxs_hard) # type 6: adversarial hard negatives: Nxn6
   l_neg_6 = einsum("nc,nkc->nk", [q, s6])
   l_neg = cat([l_neg, l_neg_6], dim=1)

   # logits: Nx(1+K+n1+n2+n3+n4+n5+n6)
   logits = cat([l_pos, l_neg], dim=1)

   # contrastive loss, positives are the 0-th
   labels = zeros(len(logits))
   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
```

Algorithm 2 Interpolated synthetic negatives generation (Type 1) in a PyTorch-like style.

```
def hard_negatives_interpolation(q, idxs_hard, alpha=0.5):
   idxs = randint(0, n_hard, size=(batch_size, n1))
   alpha = rand(size=(batch_size, n1, 1)) * 0.5
   hard_negatives = queue[gather(idxs_hard, dim=1, index=idxs)].clone().detach()
   hard_negatives = alpha * q.clone().detach()[:, None] + (1 - alpha) * hard_negatives
   return normalize(hard_negatives, dim=-1).detach()
```

Algorithm 3 Extrapolated synthetic negatives generation (Type 2) in a PyTorch-like style.

```
def hard_negatives_extrapolation(q, idxs_hard, beta=1.5):
   idxs = randint(0, n_hard, size=(batch_size, n2))
   beta = 1 + rand(size=(batch_size, n2, 1)) * 0.5
   hard_negatives = queue[gather(idxs_hard, dim=1, index=idxs)].clone().detach()
   hard_negatives = q.clone().detach()[:, None] + beta * (hard_negatives - q.clone().detach()[:, None])
   return normalize(hard_negatives, dim=-1).detach()
```

Algorithm 4 Mixup hard negatives generation (Type 3) in a PyTorch-like style.

```
def hard_negatives_mixup(q, idxs_hard, gamma=1.0):
   idxs1, idxs2 = randint(0, n_hard, size=(2, batch_size, n3))
   gamma = rand(size=(batch_size, n3, 1)) * 1.0
   hard_negatives1 = queue[gather(idxs_hard, dim=1, index=idxs1)].clone().detach()
   hard_negatives2 = queue[gather(idxs_hard, dim=1, index=idxs2)].clone().detach()
   neg_hard = gamma * hard_negatives1 + (1 - gamma) * hard_negatives2
   return normalize(neg_hard, dim=-1).detach()
```

Algorithm 5 Noise-injected synthetic negatives generation (Type 4) in a PyTorch-like style.

```
def hard_negatives_noise_inject(q, idxs_hard, sigma=0.01):
   idxs = randint(0, n_hard, size=(batch_size, n4))
   hard_negatives = queue[gather(idxs_hard, dim=1, index=idxs)].clone().detach()
   noise = randn_like(hard_negatives) * sigma
   return normalize(hard_negatives + noise, dim=-1).detach()
```

Algorithm 6 Perturbed synthetic negatives generation (Type 5) in a PyTorch-like style.

```
def hard_negatives_perturbed(q, idxs_hard, delta=0.01, epsilon=1e-5):
   idxs = randint(0, n_hard, size=(batch_size, n5))
   hard_negatives = queue[idxs_hard[arange(batch_size).unsqueeze(1), idxs]].clone().detach()
   hard_negatives_list = []
   for i in range(hard_negatives.size(1)):
      neighbor = hard_negatives[:, i, :].detach().clone().requires_grad_(True)
      similarity = einsum('nc,nc->n', [q, neighbor])
      grad = autograd.grad(similarity.sum(), neighbor, create_graph=False)[0]
      perturbed_neighbor = neighbor + delta * grad
      hard_negatives_list.append(perturbed_neighbor.detach())

   hard_negatives_final = stack(hard_negatives_list, dim=1)
   return normalize(hard_negatives_final, dim=-1).detach()
```

Algorithm 7 Adversarial synthetic negatives generation (Type 6) in a PyTorch-like style.

```
def hard_negatives_adversarial(q, idxs_hard, eta=0.01):
   batch_size = q.size(0)
   idxs = randint(0, n_hard, size=(batch_size, n6))
   hard_negatives = queue[gather(idxs_hard, dim=1, index=idxs)]
   hard_negatives_list = []
   for i in range(hard_negatives.size(1)):
      neighbor = hard_negatives[:, i, :].requires_grad_(True)
      similarity = einsum('nc,nc->n', [q, neighbor])
      gradient = autograd.grad(outputs=similarity.sum(),
            inputs=neighbor,
            retain_graph=True)[0]
      adversarial_negative = neighbor + eta * gradient.sign()
      hard_negatives_list.append(adversarial_negative.detach())
   hard_negatives_final = stack(hard_negatives_list, dim=1)
   return normalize(hard_negatives_final, dim=-1)
```

(a) 40 classes (b) 100 classes (c) 1000 classes

Figure 12. **UMAP visualizations of features extracted from SynCo pretrained for 200 epochs.** The visualizations correspond to 40, 100, and 1000 classes of ImageNet validation set.

(a) 40 classes (b) 100 classes (c) 1000 classes

Figure 13. **UMAP visualizations of features extracted from SynCo pretrained for 800 epochs.** The visualizations correspond to 40, 100, and 1000 classes of ImageNet validation set.

(a) 40 classes (b) 100 classes (c) 1000 classes

Figure 14. **UMAP visualizations of features extracted from MoCo [42] pretrained for 200 epochs.** The visualizations correspond to 40, 100, and 1000 classes of ImageNet validation set.

(a) 40 classes (b) 100 classes (c) 1000 classes

Figure 15. **UMAP visualizations of features extracted from MoCo-v2 [17] pretrained for 200 epochs.** The visualizations correspond to 40, 100, and 1000 classes of ImageNet validation set.

(a) 40 classes (b) 100 classes (c) 1000 classes

Figure 16. **UMAP visualizations of features extracted from MoCo-v2 [17] pretrained for 800 epochs.** The visualizations correspond to 40, 100, and 1000 classes of ImageNet validation set.

(a) 40 classes (b) 100 classes (c) 1000 classes

Figure 17. **UMAP visualizations of features extracted from the supervised model [70].** The plots show feature distributions for 40, 100, and 1000 classes from the ImageNet validation set.

Figure 18. **t-SNE visualization of ImageNet class embeddings in SynCo's feature space after 200 epochs of pretraining.** Each point represents the average feature vector of validation set images for one class. The visualization reveals semantic clustering, with similar concepts appearing close together.

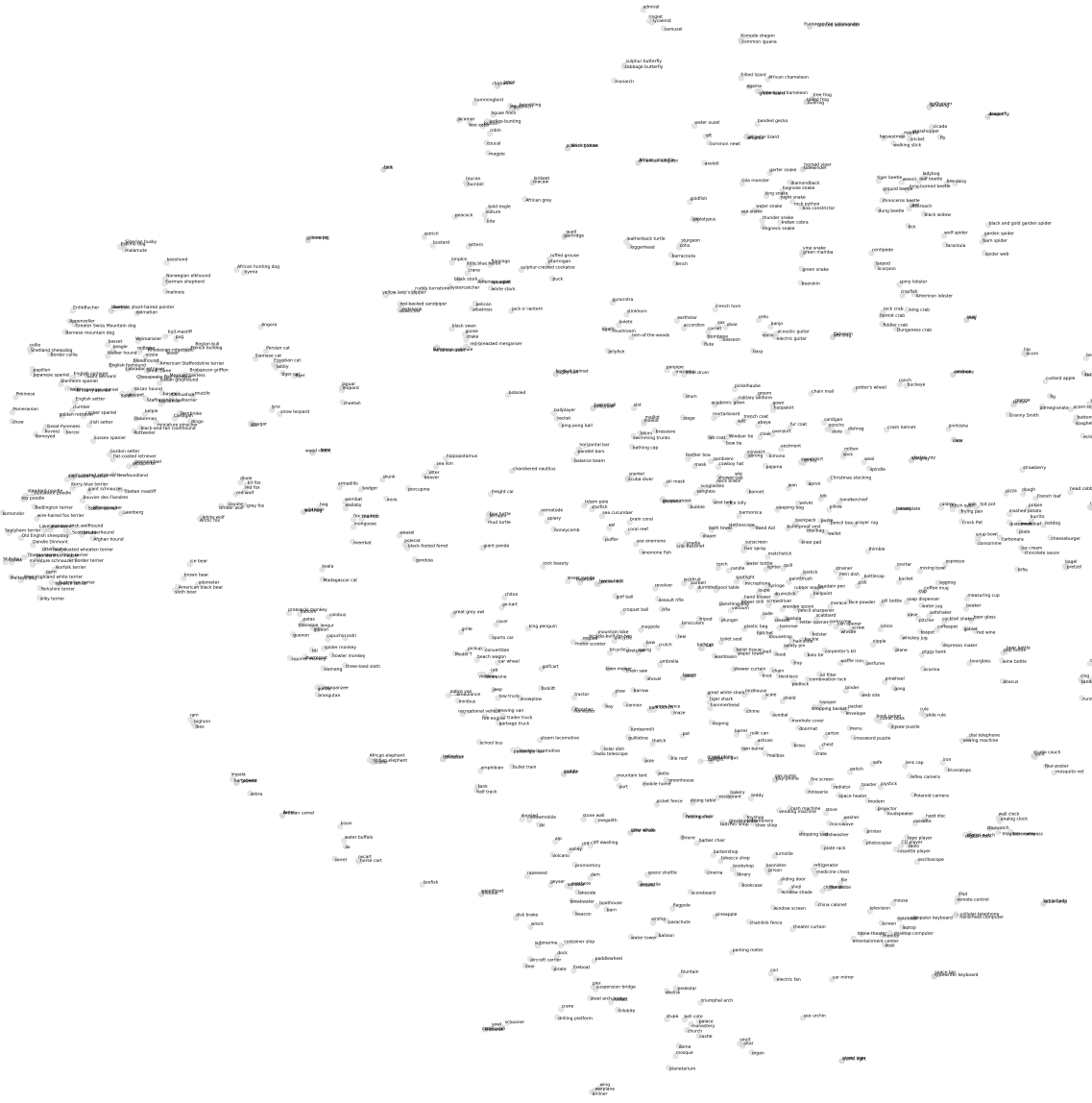


Figure 19. **t-SNE visualization of ImageNet class embeddings in SynCo's feature space after 800 epochs of pretraining.** Each point represents the average feature vector of validation set images for one class.

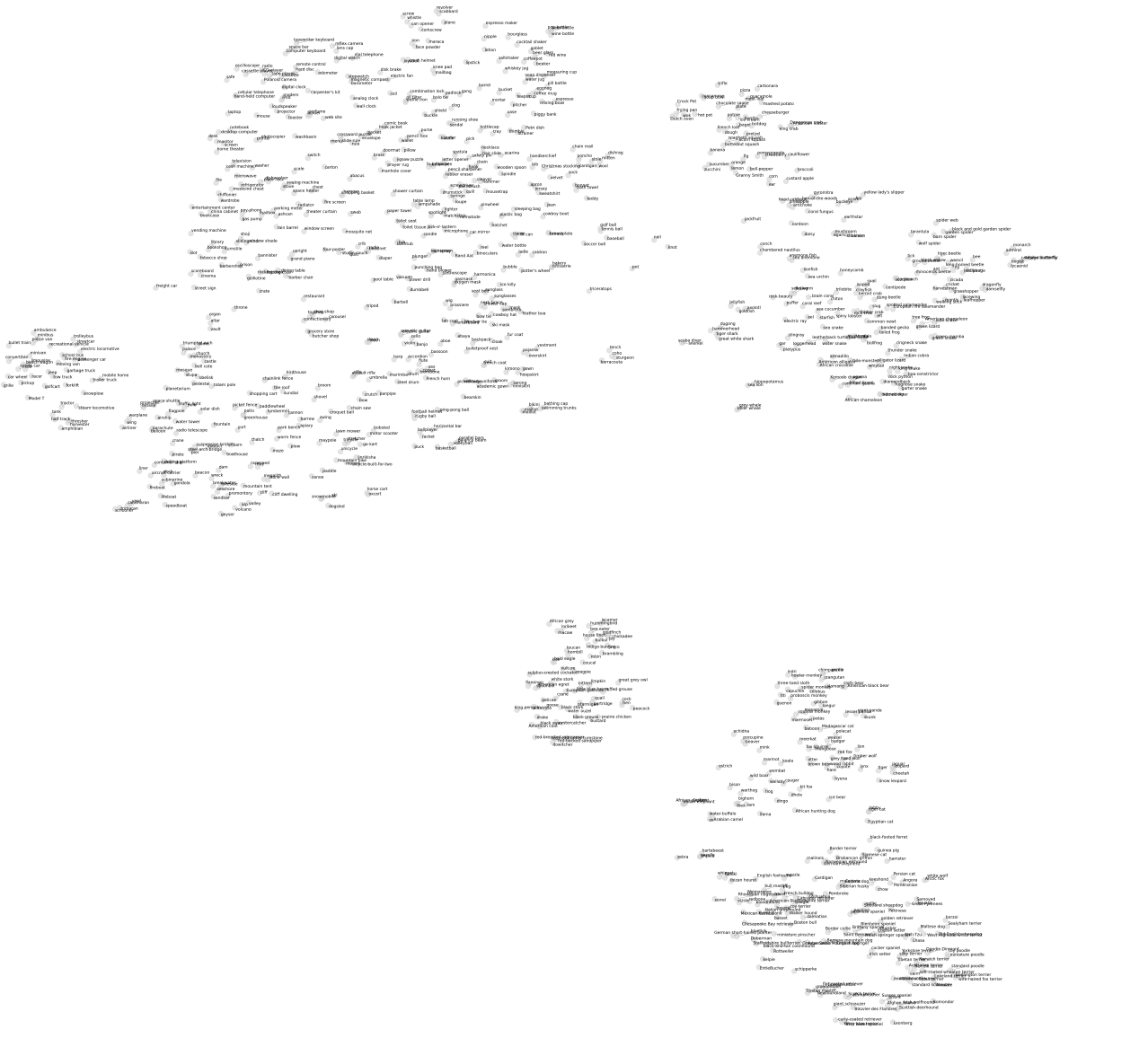


Figure 20. **t-SNE visualization of ImageNet class embeddings in MoCo's [42] feature space after 200 epochs of pretraining.** Each point represents the average feature vector of validation set images for one class.

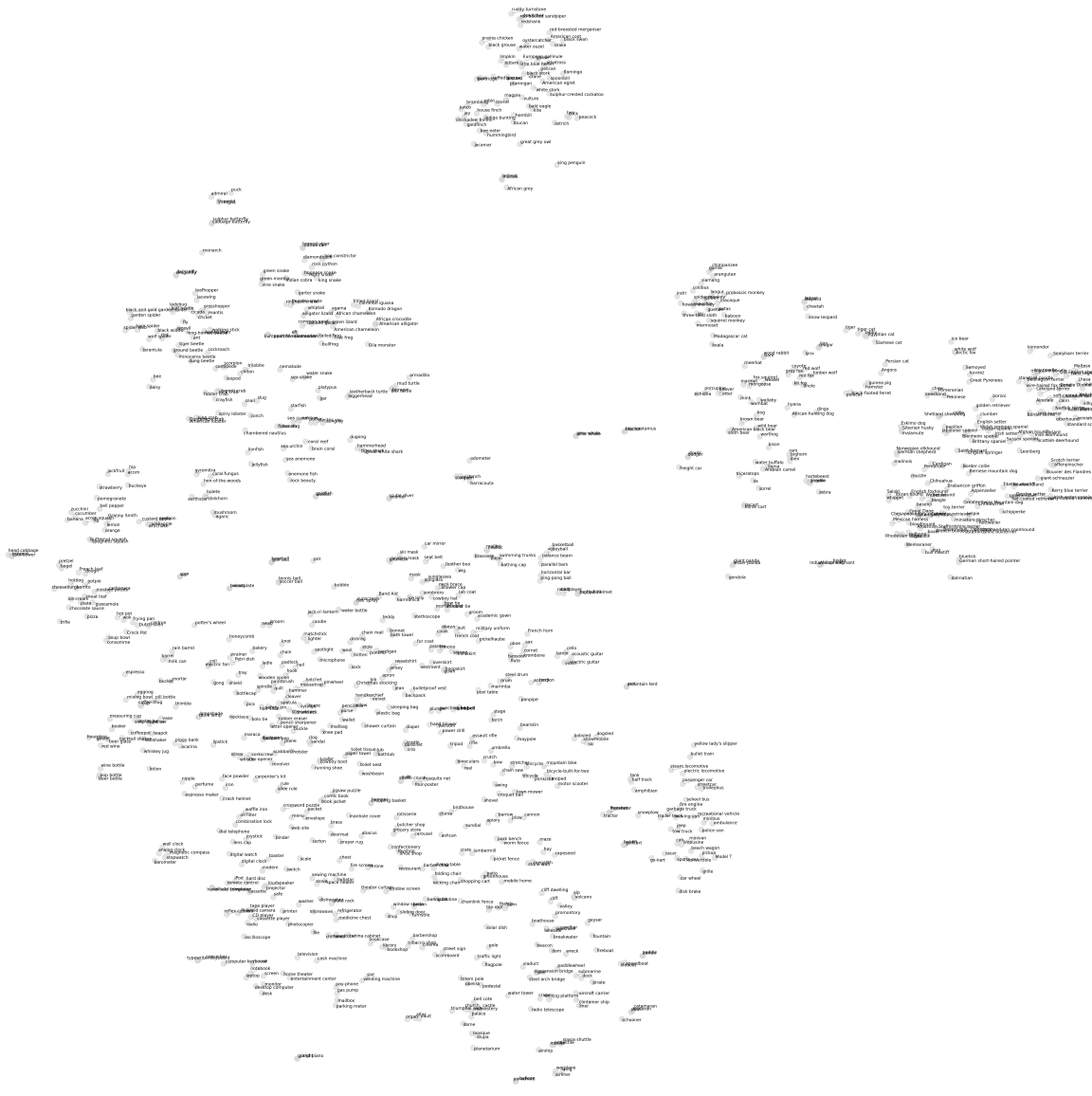


Figure 21. **t-SNE visualization of ImageNet class embeddings in MoCo-v2's [17] feature space after 200 epochs of pretraining.** Each point represents the average feature vector of validation set images for one class.

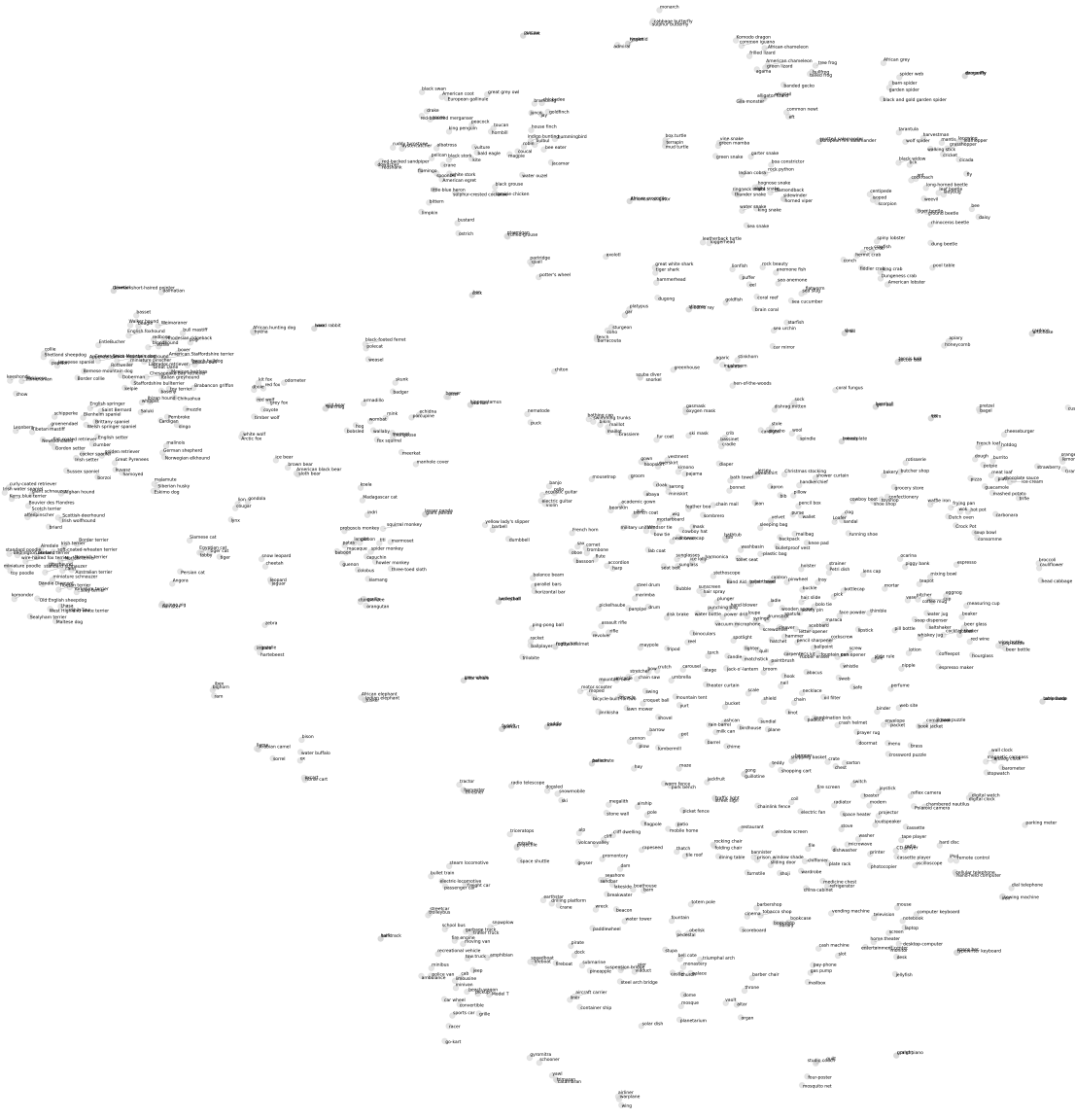


Figure 22. **t-SNE visualization of ImageNet class embeddings in MoCo-v2's [17] feature space after 800 epochs of pretraining.** Each point represents the average feature vector of validation set images for one class.

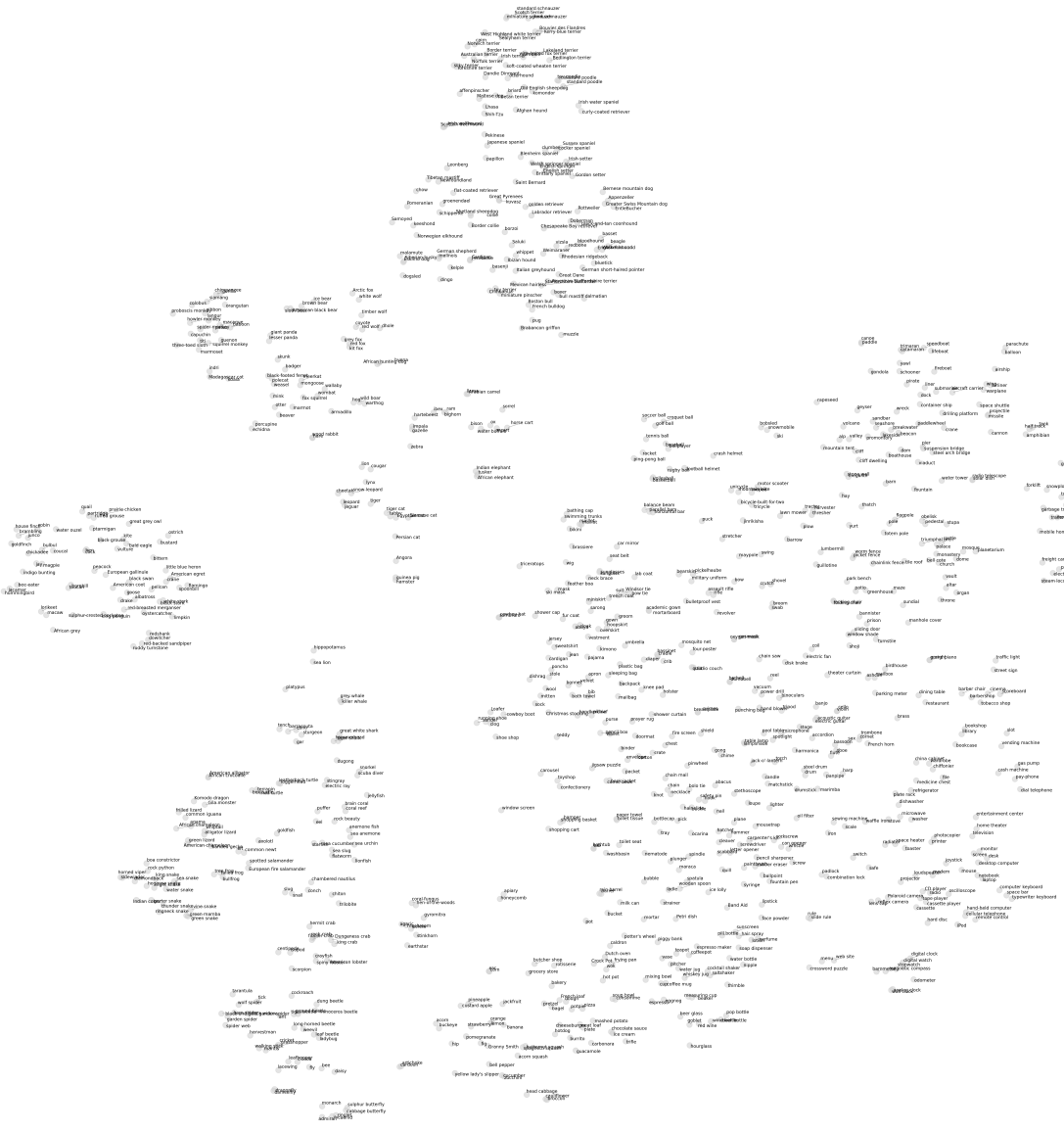


Figure 23. **t-SNE visualization of ImageNet class embeddings in the supervised feature space [70].** Each point corresponds to the mean feature vector of validation images belonging to a single class.