SynCo: Synthetic Hard Negatives for Contrastive Visual Representation Learning

Nikolaos Giakoumoglou[1], **Tania Stathaki**[1]

[1]Imperial College London

Contrastive learning has become a dominant approach in self-supervised visual representation learning, but efficiently leveraging hard negatives, which are samples closely resembling the anchor, remains challenging. We introduce SynCo (**Sy**nthetic **n**egatives in **Co**ntrastive learning), a novel approach that improves model performance by generating synthetic hard negatives on the representation space. Building on the MoCo framework, SynCo introduces six strategies for creating diverse synthetic hard negatives "on-the-fly" with minimal computational overhead. SynCo achieves faster training and strong representation learning, surpassing MoCo-v2 by +0.4% and MoCHI by +1.0% on ImageNet ILSVRC-2012 linear evaluation. It also transfers more effectively to detection tasks achieving strong results on PASCAL VOC detection (57.2% AP) and significantly improving over MoCo-v2 on COCO detection (+1.0% AP^{bb}) and instance segmentation (+0.8% AP^{msk}). Our synthetic hard negative generation approach significantly enhances visual representations learned through self-supervised contrastive learning.

Date: October 3, 2024
Correspondence: Nikolaos Giakoumoglou <nikos@imperial.ac.uk>
Code: https://github.com/giakoumoglou/synco

IMPERIAL

1 Introduction

Contrastive learning has emerged as a prominent approach in self-supervised learning, significantly advancing representation learning from unlabeled data through sophisticated feature space optimization techniques and novel architectural innovations. This technique, which distinguishes between similar and dissimilar data pairs, has shown remarkable promise in visual representation tasks across diverse domains and applications. Seminal works such as SimCLR (Chen et al., 2020a) and MoCo (He et al., 2020) established instance discrimination as a pretext task. These methods generate multiple views of the same data point through augmentation, training the model to minimize distance between positive pairs (augmented views of the same instance) while maximizing it for negative pairs (views of different instances).

Despite its effectiveness, instance discrimination faces significant computational and methodological challenges. A key limitation is the need for numerous negative samples, often leading to increased computational costs and memory requirements. For example, SimCLR requires large batch sizes for sufficient negatives (Chen et al., 2020a).

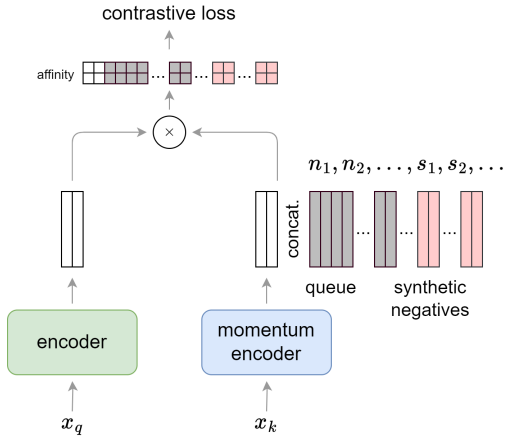


Figure 1 SynCo extends MoCo (He et al., 2020; Chen et al., 2020c) by introducing synthetic hard negatives generated "on-the-fly" from a memory queue. Two augmented views $\mathbf{x}_q$ and $\mathbf{x}_k$ are processed by an encoder and momentum encoder to produce features $\mathbf{q}$ and $\mathbf{k}$. Synthetic hard negatives $\mathbf{s}_1, \mathbf{s}_2, \ldots$ generated using SynCo strategies are concatenated with queue negatives $\mathbf{n}_1, \mathbf{n}_2, \ldots$ to compute the affinity matrix for InfoNCE loss with the positive pair ($\mathbf{q}$, $\mathbf{k}$).

While approaches like MoCo address some issues through dynamic queues and momentum encoders (He et al., 2020; Chen et al., 2020c), they still face challenges in selecting and maintaining high-quality hard negatives. Some variations, like SimCo (Zhang et al., 2022a), take a different approach by removing both the momentum encoder and queue in favor of a dual temperature mechanism that modulates positive and negative sample distances differently in the InfoNCE loss.

Recent studies have highlighted the importance of data augmentations in learning robust representations (Chen et al., 2020a; Dwibedi et al., 2021; Tian et al., 2020b; Wang and Qi, 2022; Reed et al., 2021; Balestriero et al., 2023; Rojas-Gomez et al., 2024; Giakoumoglou et al., 2025). These transformations provide diverse, challenging copies of images, increasing the difficulty of the self-supervised task. Moreover, techniques that combine data at the pixel level (Zhang et al., 2017b; Yun et al., 2019) or feature level (Verma et al., 2018) have proven effective in helping models learn resilient features. The concept of challenging negative samples has been explored to enhance contrastive learning, with MoCHI (Kalantidis et al., 2020) showing improvements by incorporating harder negatives.

The concept of challenging negative samples has been explored as a way to enhance contrastive learning models. These samples, which lie close to the decision boundary, are crucial for refining the model's discriminative abilities. Recent work like MoCHI (Kalantidis et al., 2020) has shown improvements by incorporating harder negatives. However, while the potential of hard negatives is clear, recent trends in AI have shifted focus toward large-scale foundation models (Bommasani et al., 2021; Awais et al., 2023), leaving this direction relatively unexplored. Yet, as Yann LeCun observed, "*if AI is a cake, self-supervised learning is the bulk of the cake*". We argue that revisiting self-supervised approaches, particularly through innovative hard negative strategies, remains crucial for advancing AI systems.

In this paper, we present SynCo (**Sy**nthetic **n**egatives in **Co**ntrastive learning), a novel approach to contrastive learning that leverages synthetic hard negatives to improve the learning process. Building on the foundations of MoCo, SynCo introduces six distinct strategies for generating synthetic hard negatives, each designed to provide diverse and challenging contrasts to the model. Our approach focuses on leveraging only the most challenging negatives (determined by their similarity scores with the query) to create new synthetic negatives that are both difficult and diverse. By incorporating these synthetic samples, SynCo aims to push the boundaries of contrastive learning, improving both the efficiency and effectiveness of the training process.

The main **contributions** of our work are as follows: **(i)** We introduce SynCo, a contrastive learning framework that improves representation learning by leveraging synthetic hard negatives. SynCo improves model discrimination by generating challenging negatives "on-the-fly" from a memory queue using *six* strategies targeting different aspects of the feature space. This process improves performance without significant computational overhead. **(ii)** SynCo's strategies extend MoCHI (Kalantidis et al., 2020) by exploring boundaries beyond existing negatives and combining gradient-guided perturbations with controlled stochastic noise, improving uniformity (Figure 3) and inter-/intra-class distances (Figure 4). **(iii)** We show improvements across multiple downstream tasks: linear evaluation (Tables 1 and 2), semi-supervised learning (Table 3), and object detection (Tables 4 and 5).

2 Related Work

2.1 Contrastive Learning

Contrastive learning methods focus on instance discrimination as a pretext task, treating each image as its own class (Chen et al., 2020a; He et al., 2020). The core principle involves bringing an anchor and a "positive" sample closer in the representation space while pushing the anchor away from "negative" samples (Khosla et al., 2021). Positive pairs are created through multiple views of each data point (Tian et al., 2020b; Caron et al., 2020), using techniques such as color decomposition (Tian et al., 2020a), random augmentation (Chen et al., 2020a; He et al., 2020), image patches (van den Oord et al., 2019), or student-teacher representations (Grill et al., 2020; Caron et al., 2021; Oquab et al., 2023). The common training objective, based on InfoNCE (van den Oord et al., 2019) or its variants (Chen et al., 2020a; Dwibedi et al., 2021; Tomasev et al., 2022; Yeh et al., 2022), aims to maximize mutual information (Hjelm et al., 2019; Bachman et al., 2019), necessitating numerous negative pairs. While some approaches like SimCLR use large batch sizes (Chen et al., 2020a),

others like MoCo (He et al., 2020; Chen et al., 2020c), PIRL (Misra and van der Maaten, 2019), and InstDis (Wu et al., 2018) employ memory structures. Recent advancements explore strategies such as regularizers (Mitrovic et al., 2020; Bardes et al., 2022a; Zhu et al., 2022; Bardes et al., 2022b) or prevent model collapse via redundancy reduction (Zbontar et al., 2021; Bandara et al., 2023). Some methods eliminate negative samples through asymmetric Siamese structures (Grill et al., 2020; Chen and He, 2020; Caron et al., 2021; Oquab et al., 2023). While other approaches address the false-negative pair issue (Zhuang et al., 2019; Li et al., 2021) and improve representation learning by separating the learning of features and metrics into distinct phases (Yeh et al., 2022).

2.2 Hard Negatives

Hard negatives are critical in contrastive learning as they improve visual representations by helping define the representation space. These challenging samples are harder to distinguish from the anchor, enabling the model to better differentiate between similar features. Hard negative mining involves selecting samples similar to positives but different enough to aid learning distinctive features. MoCo (He et al., 2020) maintains challenging negatives via dynamic queues and momentum updates, while SimCLR (Chen et al., 2020a) and InfoMin (Tian et al., 2020b) adjust negative difficulty through augmentation techniques. Recent work has examined optimal sampling strategies (Robinson et al., 2021; Jiang et al., 2025), semantic negative sampling (Ge et al., 2022), and broader negative sampling impacts (Yang et al., 2024). Building on these ideas, MoCHI (Kalantidis et al., 2020) proposes interpolation between query and hard negatives, and linear combination of hard negative pairs. While these geometric transformations provide improvements, they are limited to exploring the convex hull of existing negatives. Our work **extends** this foundation in *three* key directions: **(i)** Boundary exploration: Type 2 (extrapolation) pushes beyond the convex hull to explore regions outside existing negatives; **(ii)** Gradient-guided generation: Types 5-6 leverage optimization landscape information for principled perturbations rather than geometric mixing; **(iii)** Stochastic robustness: Type 4 introduces controlled noise to prevent overfitting to specific negative patterns.

3 Preliminaries

In this section, we establish the theoretical foundations of contrastive learning and analyze the critical role of hard negatives in representation learning.

3.1 Contrastive Learning

Contrastive learning seeks to differentiate between similar and dissimilar data pairs, often treated as a dictionary look-up where representations are optimized to align positively paired data through contrastive loss in the representation space (He et al., 2020). Given an image x, and a distribution of image augmentation $\mathcal{T}$, we create two augmented views of the same image using the transformation $t_q, t_k \sim \mathcal{T}$, *i.e.*, $x_q = t_q(x)$ and $x_k = t_k(x)$. Two encoders, f_q and f_k, namely the query and key encoders, generate the vectors $\mathbf{q} = f_q(x_q)$ and $\mathbf{k} = f_k(x_k)$, respectively. The learning objective minimizes a contrastive loss using the InfoNCE criterion (van den Oord et al., 2019):

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

Here, $\mathbf{k}$ is f_k's output from the same augmented image as $\mathbf{q}$, and $\mathcal{Q} = \{\mathbf{n}_i\}_{i=1}^K$ contains K negative samples from different images. The temperature parameter τ adjusts scaling for the ℓ_2-normalized vectors $\mathbf{q}$ and $\mathbf{k}$. The key encoder f_k can be updated in two ways. In the synchronized update approach, f_k is updated synchronously with f_q, maintaining identical weights throughout training (Chen et al., 2020a). Alternatively, a momentum update scheme can be employed, where f_k is updated using: $\theta_k \leftarrow m \cdot \theta_k + (1 - m) \cdot \theta_q$ (He et al., 2020). Here, θ_k and θ_q are the parameters of f_k and f_q, and $m \in [0, 1]$ is the momentum coefficient. This approach allows f_k to evolve slowly, providing consistent negative samples and stabilizing the learning process. The memory bank $\mathcal{Q}$ can be defined as an external memory of all dataset images (Misra and van der

Maaten, 2019; Tian et al., 2020a; Wu et al., 2018), a queue of recent batches (He et al., 2020), or the current minibatch (Chen et al., 2020a).

The gradient of the contrastive loss in Equation (1) with respect to the query $\mathbf{q}$ is given by:

$$\frac{\partial \mathcal{L}(\mathbf{q}, \mathbf{k}, \mathcal{Q})}{\partial \mathbf{q}} = -\frac{1}{\tau}\left((1 - p_k) \cdot \mathbf{k} - \sum_{\mathbf{n} \in \mathcal{Q}} p_n \cdot \mathbf{n}\right) \quad \text{where} \quad p_{z_i} = \frac{\exp(\mathbf{q}^\top \cdot \mathbf{z_i}/\tau)}{\sum_{j \in Z} \exp(\mathbf{q}^\top \cdot \mathbf{z_j}/\tau)} \tag{2}$$

with $\mathbf{z_i}$ being a member of the set $\mathcal{Q} \cup \{\mathbf{k}\}$. The positive and negative logits contribute to the loss similarly to a $(K+1)$-way cross-entropy classification, with the key logit representing the query's latent class (Arora et al., 2019).

3.2 Understanding Hard Negatives

The effectiveness of contrastive learning approaches hinges critically on the utilization of hard negatives (Arora et al., 2019; Hadsell et al., 2006; Iscen et al., 2018; Mishchuk et al., 2017; Wu et al., 2018; Kalantidis et al., 2020). Current approaches face significant challenges in efficiently leveraging these hard negatives. Sampling from within the same batch necessitates larger batch sizes (Chen et al., 2020a, 2021). Conversely, maintaining a memory bank containing representations of the entire dataset incurs substantial computational overhead in keeping the memory up-to-date (Misra and van der Maaten, 2019; Wu et al., 2018; He et al., 2020; Chen et al., 2020c). These limitations underscore the need for more efficient strategies to generate and utilize hard negatives in contrastive learning frameworks.

Hardness of negatives. The "hardness" of negative samples, defined by their similarity to positive samples in the representation space, determines how challenging they are for the model to differentiate, directly impacting the effectiveness of the contrastive learning process. Figure 2a illustrates the evolution of negative sample hardness during MoCo-v2 training. Initially, the distribution of these probabilities is relatively uniform. However, as training progresses, a clear trend emerges: fewer negatives contribute significantly to the loss function. This observation suggests that the model rapidly learns to distinguish most negatives, leaving only a small subset that remains challenging. Such a phenomenon underscores the importance of maintaining a diverse pool of hard negatives throughout the training process to sustain effective learning (Kalantidis et al., 2020).

Difficulty of the proxy task. The difficulty of the proxy task in contrastive learning, typically defined by the self-supervised objective, significantly influences the quality of learned representations. Figure 2b compares the proxy task performance of MoCo and MoCo-v2 on ImageNet-100, measured by the percentage of queries where the key ranks above all negatives. Notably, MoCo-v2, which employs more aggressive augmentations, exhibits lower proxy task performance compared to MoCo, indicating a more challenging learning objective. Paradoxically, this increased difficulty correlates with improved performance on downstream tasks such as linear classification (Kalantidis et al., 2020). Additionally, Figure 2c demonstrates how SynCo's performance varies under different configurations, providing insights into the optimal parameter settings for balancing proxy task difficulty and representation quality. The complete SynCo framework consistently outperforms each individual strategy (types 1-6) when applied in isolation. This counterintuitive relationship between proxy task difficulty and downstream performance suggests that more challenging self-supervised objectives can lead to the learning of more robust and transferable representations, motivating the development of strategies to dynamically modulate task difficulty during training.

4 Synthetic Hard Negatives in Contrastive Learning

In this section, we present an approach for generating synthetic hard negatives in the representation space using **six** distinct strategies. Building on MoCHI, we introduce four additional strategies that explore complementary aspects of the representation space for generating synthetic hard negatives (*cf.* Section E for differences from our method). We refer to our proposed approach as SynCo ("**Sy**nthetic **N**egatives in **Co**ntrastive Learning").

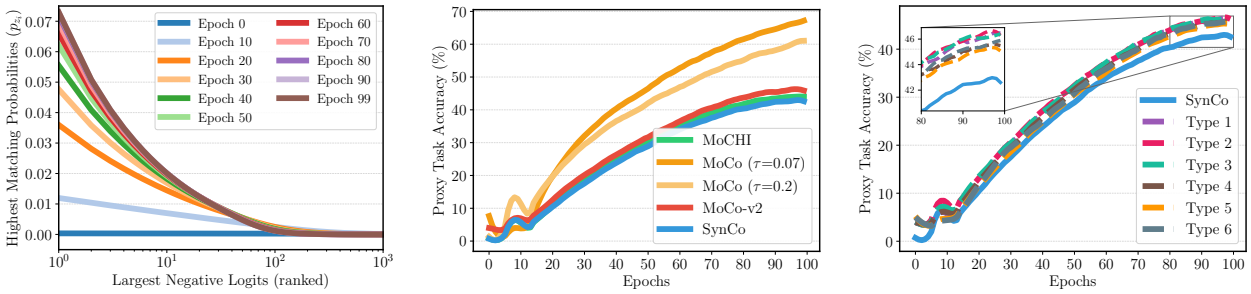


Figure 2 ImageNet-100 experiments. (a) Histogram of the top 1024 matching probabilities p_{z_i}, $z_i \in \mathcal{Q}$ for MoCo-v2 over various epochs. Logits are organized in descending order, and each line indicates the mean matching probability across all queries. (b) Performance comparison of MoCo, MoCo-v2, MoCHI, SynCo in terms of accuracy on the proxy task (percentage of queries where the key is ranked higher than all negatives). (c) Performance comparison of SynCo under various configurations in terms of accuracy on the proxy task.

4.1 Generating Synthetic Hard Negatives

Let $\mathbf{q}$ represent the query image, $\mathbf{k}$ its corresponding key, and $\mathbf{n} \in \mathcal{Q}$ denote the negative features from a memory structure of size K. The loss associated with the query is computed using the logits $\ell(\mathbf{z_i}) = \mathbf{q}^\top \cdot \mathbf{z}_i/\tau$, which are processed through a softmax function. We define $\hat{\mathcal{Q}} = \{\mathbf{n}_1, \mathbf{n}_2, \ldots, \mathbf{n}_K\}$ as the ordered set of all negative features, where $\ell(\mathbf{n}_i) > \ell(\mathbf{n}_j)$ for all $i < j$, implying that the negative features are sorted based on decreasing similarity to the query. The most challenging negatives are selected by truncating the ordered set $\hat{\mathcal{Q}}$, retaining only the first $N < K$ elements, denoted as $\hat{\mathcal{Q}}^N$. Let $S^{(i)} = \{\mathbf{s}_1^{(i)}, \mathbf{s}_2^{(i)}, \ldots, \mathbf{s}_{N_i}^{(i)}\}$ denote the set of synthetic negatives to be generated for type i, where $i \in \{1, 2, 3, 4, 5, 6\}$ represents the six different synthetic negative generation strategies and N_i denotes the cardinality of each set (*i.e.*, the number of synthetic negatives generated for the i-th strategy). Note that all synthetic hard negatives are ℓ_2-normalized before added to the set of negative logits for the query.

Interpolated synthetic negatives (type 1). Our first strategy creates synthetic negatives through controlled interpolation between samples, similar to MoCHI's type 2. This approach aims to generate features that lie in meaningful regions of the representation space between the query and existing hard negatives. For each query $\mathbf{q}$, we propose to generate N_1 synthetic hard negative features by mixing the query $\mathbf{q}$ with a randomly chosen feature from the N hardest negatives in $\hat{\mathcal{Q}}^N$. Then a synthetic negative feature $\mathbf{s}_k^1 \in S^1$ is given by:

$$\mathbf{s}_k^1 = \alpha_k \cdot \mathbf{q} + (1 - \alpha_k) \cdot \mathbf{n}_i, \quad \alpha_k \in (0, \alpha_{\max}) \tag{3}$$

where $\mathbf{n}_i \in \hat{\mathcal{Q}}^N$ and α_k is randomly sampled from a uniform distribution in the range $(0, \alpha_{\max})$. Interpolation creates a synthetic embedding that lies between the query and the negative in the representation space. We set $\alpha_{\max} = 0.5$ to guarantee that the contribution of the query is always less than than the negative.

Extrapolated synthetic negatives (type 2). As a natural extension of interpolation, we propose extrapolation to explore the "opposite" direction in feature space. While this approach operates further from the decision boundary, we carefully control the exploration through coefficients to balance the difficulty of synthetic negatives, ensuring they provide meaningful learning signals without making the contrastive task intractable. For each query $\mathbf{q}$, we propose to generate N_2 hard negative features by extrapolating beyond the query embedding in the direction of the hardest negative features. We use a randomly chosen feature from the N hardest negatives in $\hat{\mathcal{Q}}^N$. Then $\mathbf{s}_k^2 \in S^2$ is given by:

$$\mathbf{s}_k^2 = \mathbf{n}_i + \beta_k \cdot (\mathbf{n}_i - \mathbf{q}), \quad \beta_k \in (1, \beta_{\max}) \tag{4}$$

where $\mathbf{n}_i \in \hat{\mathcal{Q}}^N$ and β_k is randomly sampled from a uniform distribution in the range $(1, \beta_{\max})$. Extrapolation

generates a synthetic embedding that lies beyond the query embedding in the direction of the hardest negative. We choose $\beta_{\max} = 1.5$.

Mixup synthetic negatives (type 3). We propose to generate challenging synthetic negatives by combining pairs of hard negative examples, similar to MoCHI's type 1. For each query $\mathbf{q}$, we generate N_3 hard negative features by combining pairs of the N hardest existing negative features in $\hat{\mathcal{Q}}^N$. For $\mathbf{s}_k^3 \in S^3$ we have:

$$\mathbf{s}_k^3 = \gamma_k \cdot \mathbf{n}_i + (1 - \gamma_k) \cdot \mathbf{n}_j, \quad \gamma_k \in (0, 1) \tag{5}$$

where $\mathbf{n}_i, \mathbf{n}_j \in \hat{\mathcal{Q}}^N$ and γ_k is randomly sampled from a uniform distribution in the range $(0, 1)$. Mixup combines pairs of the hardest existing negative features to create a synthetic embedding that represents a blend of challenging cases.

Noise-injected synthetic negatives (type 4). To prevent overfitting to specific negative patterns while maintaining the characteristics of hard negatives, we introduce controlled stochasticity through noise injection. For each query $\mathbf{q}$, we propose to generate N_4 hard negative features by adding Gaussian noise to the hardest negative features. Each synthetic negative feature $\mathbf{s}_k^4 \in S^4$ is given by:

$$\mathbf{s}_k^4 = \mathbf{n}_i + \mathcal{N}(\mathbf{0}, \sigma^2 \cdot \mathbf{I}) \tag{6}$$

where $\mathbf{n}_i \in \hat{\mathcal{Q}}^N$ and $\mathcal{N}(\mathbf{0}, \sigma^2 \cdot \mathbf{I})$ represents Gaussian noise with standard deviation σ ($\mathbf{I}$ is the identity matrix). Noise injection adds Gaussian noise to the hardest negative features, producing a synthetic embedding with added randomness.

Perturbed synthetic negatives (type 5). Drawing inspiration from adversarial training (Mehrabi et al., 2021), we introduce perturbed synthetic negatives that use gradient-based perturbations with variable magnitudes. For each query $\mathbf{q}$, we propose to generate N_5 hard negative features by perturbing the embeddings of the hardest negative features. We formulate each synthetic negative feature $\mathbf{s}_k^5 \in S^5$ as:

$$\mathbf{s}_k^5 = \mathbf{n}_i + \delta \cdot \nabla_{\mathbf{n}_i} \text{sim}(\mathbf{q}, \mathbf{n}_i) \tag{7}$$

where $\mathbf{n}_i \in \hat{\mathcal{Q}}^N$ and $\text{sim}(\cdot, \cdot)$ is the similarity function and δ controls the perturbation magnitude. Perturbation modifies the embeddings of the hardest negative features based on the gradient of the similarity function, creating synthetic negatives that are slightly adjusted to be more challenging for the model. This approach offers greater flexibility than fixed interpolation, as it generalizes to arbitrary similarity functions and can generate negatives of varying hardness.

Adversarial synthetic negatives (type 6). While similar in concept to type 5, adversarial synthetic negatives differ fundamentally in their gradient scaling approach. For each query $\mathbf{q}$, we propose to generate N_6 hard negative features by applying adversarial perturbations to the hardest negative features to maximize their similarity to the query embeddings. Each synthetic negative $\mathbf{s}_k^6 \in S^6$ is defined as:

$$\mathbf{s}_k^6 = \mathbf{n}_i + \eta \cdot \text{sign}(\nabla_{\mathbf{n}_i} \text{sim}(\mathbf{q}, \mathbf{n}_i)) \tag{8}$$

where $\mathbf{n}_i \in \hat{\mathcal{Q}}^N$ and η controls the perturbation magnitude. Adversarial hard negatives apply adversarial perturbations to the hardest negative features, specifically altering them to maximize their similarity to the query embeddings, thereby producing the most challenging contrasts. Where type 5 allows variable perturbation sizes, type 6 enforces unit magnitude through the sign function.

4.2 Integrating Synthetic Hard Negatives into the Contrastive Loss

The synthetic hard negatives generated are integrated into the contrastive learning process by modifying the InfoNCE loss. Let $\mathcal{S} = \bigcup_{i=1}^{6} S^{(i)}$ represent the concatenation of all synthetic hard negatives, where $S^{(i)}$ is the set of synthetic negatives generated by the i-th strategy. This combined set of synthetic negatives augments the original negatives $\mathcal{Q}$, providing a more diverse and challenging set of contrasts for the query. The modified InfoNCE loss is given by:

$$\mathcal{L}(\mathbf{q}, \mathbf{k}, \mathcal{Q}, \mathcal{S}) = -\log \frac{\exp(\mathbf{q}^\top \cdot \mathbf{k}/\tau)}{\exp(\mathbf{q}^\top \cdot \mathbf{k}/\tau) + \underbrace{\sum_{\mathbf{n}\in\mathcal{Q}} \exp(\mathbf{q}^\top \cdot \mathbf{n}/\tau)}_{\text{memory-based negatives}} + \underbrace{\sum_{\mathbf{s}\in\mathcal{S}} \exp(\mathbf{q}^\top \cdot \mathbf{s}/\tau)}_{\text{synthetic negatives}}}. \tag{9}$$

Here, $\mathcal{Q}$ is the set of original memory-based negatives, and $\mathcal{S}$ is the set of all synthetic hard negatives. By incorporating both real and synthetic negatives, the model is exposed to a wider variety of challenging examples, which encourages learning more robust and generalizable representations. The overall computational overhead of SynCo is roughly equivalent to increasing the queue/memory by $\sum_{i=1}^{6} N_i \ll K$, along with the additional yet negligible cost of generating the synthetic negatives. Since synthetic negatives are generated "on-the-fly" during training and can be efficiently computed in parallel with the forward pass, the additional computational cost is marginal compared to the base contrastive learning framework. Moreover, the memory footprint remains manageable as synthetic negatives do not need to be stored persistently in the memory bank. Section A provides a detailed description of the SynCo algorithm (*cf.* Algorithms 1 to 7).

5 Experiments

In this section, we present comprehensive experiments demonstrating SynCo's effectiveness on ImageNet linear evaluation, semi-supervised learning, and transfer learning to object detection. The Appendix contains: **(i)** implementation details (Section B); **(ii)** Type 1–6 ablations (Section D.1); **(iii)** hyperparameter sensitivity (σ, δ, η, N, N_i, K, Sections D, D.1 and D.2); **(iv)** vision transformers (Dosovitskiy et al., 2021; Touvron et al., 2021) (Section C.8); **(v)** robustness (Sections C.6 and C.7); **(vi)** visualizations, *e.g.*, t-SNE (Section C.9), GradCAM (Section C.10), UMAP (Section C.11), retrieval (Section C.12).

5.1 Implementation Details

We pretrain SynCo on ImageNet ILSVRC-2012 (Deng et al., 2009) using a ResNet-50 encoder (He et al., 2015). Our method builds upon MoCo-v2 (Chen et al., 2020c); thus, it is only *fair* to compare against other MoCo-based methods (Chen et al., 2020c; Li et al., 2021; Kalantidis et al., 2020; Yeh et al., 2022), which share similar architectures and training setups (*cf.* **bold** entries in Tables 1 to 5, indicating the best performance among MoCo-based methods-we *underline* the second best). For training we use $K = 65$k. For SynCo, we also have a warm-up of 10 epochs, *i.e.* for the first epochs we do *not* synthesize hard negatives. We empirically set SynCo's hyperparameters σ, δ, η to 0.01. For hard negative generation, we use the top $N = 1024$ hardest negatives, with $N_1 = N_2 = N_3 = 256$ and $N_4 = N_5 = N_6 = 64$. For ImageNet linear evaluation, we train a linear classifier on frozen features for 100 epochs, using a batch size of 256 and a cosine learning rate schedule ($lr = 30.0$). To evaluate transfer learning, we apply SynCo to object detection tasks. For PASCAL VOC (Everingham et al., 2009), we fine-tune a Faster R-CNN (Ren et al., 2016) on `trainval07+12` and test on `test2007`. For COCO (Lin et al., 2015), we use a Mask R-CNN (He et al., 2018), fine-tuning on `train2017` and evaluating on `val2017`. We employ Detectron2 (Wu et al., 2019) and report standard AP metrics, following (He et al., 2020) *without* additional tuning.

5.2 Linear Evaluation on ImageNet

We evaluate SynCo by training a linear classifier on ImageNet-pretrained frozen features. With 200 epochs of pretraining, SynCo obtains 67.9% ± 0.16% top-1 accuracy and 88.0% ± 0.05% top-5, showing clear gains

Table 1 Linear evaluation on ImageNet ILSVRC-2012. Top-1 accuracy (in %) with 200 epochs of pretraining using ResNet-50. Results for SynCo are given as avg./max over 3 runs.

Method	Top-1
Supervised	76.5
PIRL (Misra and van der Maaten, 2019)	63.6
LA (Zhuang et al., 2019)	60.2
CMC (Tian et al., 2020a)	60.0
SimSiam (Chen and He, 2020)	68.1
ReSSL (Zheng et al., 2021)	62.9
AdCo (Hu et al., 2021)	68.6
SimCLR + DCL (Yeh et al., 2022)	65.8
MoCo-based	
MoCo (He et al., 2020)	60.7 ↓6.8
PCL-v1 (Li et al., 2021)	61.5 ↓6.0
MoCo-v2 (Chen et al., 2020c) (*baseline*)	67.5 ↑0.0
MoCHI (Kalantidis et al., 2020)	66.9 ↓0.6
PCL-v2 (Li et al., 2021)	67.6 ↑0.1
MoCo-v2 + DCL (Yeh et al., 2022)	67.6 ↑0.1
MoCo-v2 + NS (Ge et al., 2022)	67.9 ↑0.4
SynCo (ours)	**67.9**/**68.1** ↑0.6

Table 2 Linear evaluation on ImageNet ILSVRC-2012. Top-1 accuracy (in %) for models trained with *extended epochs* using ResNet-50. Results for SynCo are based on 1 run. **Symbols**: † We stop generating synthetic negatives at epoch 400 (*cf.* Section C, Table 11).

Method	Epochs	Top-1
InfoMin (Tian et al., 2020b)	800	73.0
SimSiam (Chen and He, 2020)	800	68.1
SimCLR (Chen et al., 2020a)	1000	69.3
BYOL (Grill et al., 2020)	1000	74.3
DINO (Caron et al., 2021)	800	75.3
Barlow Twins (Zbontar et al., 2021)	1000	73.2
AdCo (Hu et al., 2021)	800	72.8
VICReg (Bardes et al., 2022a)	1000	73.2
CaCo (Wang et al., 2023)	800	74.1
All4One (Estepa et al., 2023)	800	66.6
MoCo-based		
MoCo-v2 (Chen et al., 2020c)	800	71.1 ↑0.0
MoCHI (Kalantidis et al., 2020)	800	68.7 ↓2.4
SynCo (ours)	800	70.7 ↓0.4
SynCo† (ours)	800	**71.6** ↑0.5

over MoCo-based methods (**+0.4%** over MoCo-v2, **+1.0%** over MoCHI, **+0.3%** over PCL-v2 and DCL). While MoCHI's hard negative generation yields lower performance than MoCo-v2, our synthetic hard negatives provide consistent and stable improvements. With 800 epochs, SynCo reaches 70.7% top-1 (**+2.0%** over MoCHI). An extended comparison is provided in Section C.1. However, at 800 epochs, it still does not surpass MoCo-v2, similar to MoCHI, likely due to an overly hard proxy task. We observe that SynCo's performance plateaus around epoch 400 (*cf.* Table 11 in Section C), indicating that continued synthetic negative generation makes the proxy task increasingly challenging in later stages. Motivated by this observation, we stop generating synthetic negatives after epoch 400, allowing the model to consolidate learned features without overwhelming the contrastive objective; this yields 71.6% top-1, a **+0.5%** improvement over MoCo-v2.

5.3 Semi-supervised Evaluation on ImageNet

We evaluate SynCo in a semi-supervised setting using 1% and 10% of labeled ImageNet data (and 100% in Section C.4, Table 12). Results in Table 3 show that with 1% labels, SynCo achieves 50.8% ± 0.21% top-1 accuracy (**+25.4%** over supervised baseline, **+2.6%** over MoCo-v2, **+2.5%** over SimCLR) and 77.5% ± 0.12% top-5 accuracy. With 10% labels, it reaches 66.6% ± 0.19% top-1 (**+10.2%** over supervised, **+0.5%** over MoCo-v2, **+1.0%** over SimCLR) and 88.0% ± 0.10% top-5 accuracy. Interestingly, when we stop generating synthetic negatives after epoch 400, similar to our observation in linear evaluation, performance improves further to 51.2% ± 0.23% top-1 and 78.0% ± 0.14% top-5 with 1% labels, and 67.1% ± 0.20% top-1 and 88.7% ± 0.11% top-5 with 10% labels.

5.4 Transferring to Detection

We evaluate the SynCo representation, pretrained for 200 epochs, by applying it to detection tasks. Results in Table 4 and Table 10 (Section C.2) show that on PASCAL VOC, SynCo achieves strong results comparable to MoCHI, while significantly outperforming the supervised baseline (**+3.7** AP). On the more challenging COCO dataset (Table 5), with 1× schedule, SynCo shows consistent improvements over the supervised baseline (AP^{bb} **+1.7**, AP^{m} **+1.6**) and MoCo-v2 (AP^{bb} **+1.0**, AP^{m} **+0.8**). SynCo achieves competitive performance with detection-specific methods, *e.g.*, DetCo (Xie et al., 2021a) or InsLoc (Yang et al., 2021).

Table 3 Semi-supervised learning on ImageNet ILSVRC-2012. Top-1 and top-5 accuracies with 1% and 10% training examples using ResNet-50. Results for SynCo are averaged over 3 runs.

Method	Epochs	Top-1		Top-5	
		1%	10%	1%	10%
Supervised		25.4	56.4	48.4	80.4
SimCLR (Chen et al., 2020a)	1000	48.3	65.6	75.5	87.8
BYOL (Grill et al., 2020)	1000	53.2	68.8	78.4	89.0
SwAV (Caron et al., 2020)	800	53.9	70.2	78.5	89.9
Barlow Twins (Zbontar et al., 2021)	1000	55.0	69.7	79.2	89.3
VICReg (Bardes et al., 2022a)	1000	54.8	69.5	79.4	89.5
All4One (Estepa et al., 2023)	800	39.0	-	60.0	-
MoCo-based					
MoCo-v2 (repr.)	800	48.2	66.1	75.8	87.6
PCL-v1 (Li et al., 2021)	200	-	-	75.3	85.6
PCL-v2 (Li et al., 2021)	200	-	-	73.9	85.0
MoCHI (repr.)	800	50.4	65.7	76.2	87.2
SynCo (ours)	800	50.8	66.6	77.5	88.0
SynCo† (ours)	800	**51.2**	**67.1**	**78.0**	**88.7**

Table 4 Transfer learning on PASCAL VOC07+12 using R50-C4. We report standard AP metrics. Results for SynCo are averaged over 3 runs.

Method	Epochs	AP	AP_{50}	AP_{75}
Supervised	200	53.5	81.3	58.8
Random init	200	33.8	60.2	33.1
SimSiam (Chen and He, 2020)	200	57.0	82.4	63.7
BYOL (Grill et al., 2020)	300	51.9	81.0	56.5
SwAV (Caron et al., 2020)	800	56.1	82.6	62.7
SimCLR (Chen et al., 2020a)	1000	56.3	81.9	62.5
Barlow Twins (Zbontar et al., 2021)	1000	56.8	82.6	63.4
Detection-specific				
SoCo (Wei et al., 2021)	100	59.1	83.4	65.6
InsLoc (Yang et al., 2021)	200	57.9	82.9	64.9
DetCo (Xie et al., 2021a)	200	57.8	82.6	64.2
ReSim (Xiao et al., 2021)	200	58.7	83.1	66.3
MoCo-based				
MoCo (He et al., 2020)	200	55.9	81.5	62.6
MoCo-v2 (Chen et al., 2020c)	200	57.0	82.4	63.6
MoCHI (Kalantidis et al., 2020)	200	**57.5**	**82.7**	**64.4**
SynCo (ours)	200	57.2	82.6	63.9

6 Discussion

This section examines how synthetic negatives affect proxy task difficulty and shape representation space utilization.

6.1 Is the Proxy Task More Difficult?

We observe that incorporating synthetic negatives leads to faster learning and improved performance. Each synthetic negative type accelerates learning compared to MoCo-v2, with the full SynCo configuration showing the most significant improvement and lowest final proxy task performance, as shown in Figure 2b. This indicates SynCo presents the most challenging proxy task, evidenced by $\max \ell(\mathbf{s}_k^i) > \max \ell(\mathbf{n}_j)$, where $\mathbf{s}_k^i \in S^i$ are synthetic negatives and $\mathbf{n}_j \in \hat{\mathcal{Q}}_N$ are original negatives. Through SynCo, we modulate proxy task difficulty via synthetic negatives, pushing the model to learn more robust features.

Table 5 Transfer learning on COCO using R50-C4 with 1× and 2× training schedules. We report AP^{bb} (bounding box detection) and AP^{m} (instance segmentation). Results for SynCo are averaged over 3 runs.

Method	Epochs	COCO 1× schedule						COCO 2× schedule					
		AP^{bb}	AP^{bb}_{50}	AP^{bb}_{75}	AP^{m}	AP^{m}_{50}	AP^{m}_{75}	AP^{bb}	AP^{bb}_{50}	AP^{bb}_{75}	AP^{m}	AP^{m}_{50}	AP^{m}_{75}
Supervised	200	38.2	58.2	41.2	33.3	54.7	35.2	40.0	59.9	43.1	34.7	56.5	36.9
Random init	200	26.4	44.0	27.8	29.3	46.9	30.8	35.6	54.6	38.2	31.4	51.5	33.5
SimSiam (Chen and He, 2020)	200	39.2	59.3	42.1	34.4	56.0	36.7	-	-	-	-	-	-
BYOL (Grill et al., 2020)	300	-	-	-	-	-	-	40.3	60.5	43.9	35.1	56.8	37.3
SwAV (Caron et al., 2020)	800	38.4	58.6	41.3	33.8	55.2	35.9	-	-	-	-	-	-
SimCLR (Chen et al., 2020a)	1000	-	-	-	-	-	-	40.3	60.5	43.9	35.1	56.8	37.3
BT (Zbontar et al., 2021)	1000	39.2	59.0	42.5	34.3	56.0	36.5	-	-	-	-	-	-
Detection-specific													
SoCo (Wei et al., 2021)	100	40.4	60.4	43.7	34.9	56.8	37.0	41.1	61.0	44.4	35.6	57.5	38.0
InsLoc (Yang et al., 2021)	200	39.5	59.1	42.7	34.5	56.0	36.8	41.4	60.9	45.0	35.9	57.6	38.4
DetCo (Xie et al., 2021a)	200	39.8	59.7	43.0	34.7	56.3	36.7	41.3	61.2	45.0	35.8	57.9	38.2
ReSim (Xiao et al., 2021)	200	39.7	59.0	43.0	34.6	55.9	37.1	-	-	-	-	-	-
MoCo-based													
MoCo (He et al., 2020)	200	38.5	58.3	41.6	33.6	54.8	35.6	40.7	60.5	44.1	35.4	57.3	37.6
MoCo-v2 (Chen et al., 2020c)	200	38.9	58.4	42.0	34.2	55.2	36.5	40.7	60.5	44.1	35.6	**57.4**	37.1
MoCHI (Kalantidis et al., 2020)	200	39.2	58.9	42.4	34.3	55.5	36.6	-	-	-	-	-	-
SynCo (ours)	200	**39.9**	**59.6**	**43.3**	**34.9**	**56.5**	**36.9**	**41.0**	**60.6**	**44.8**	**35.7**	**57.4**	**38.1**

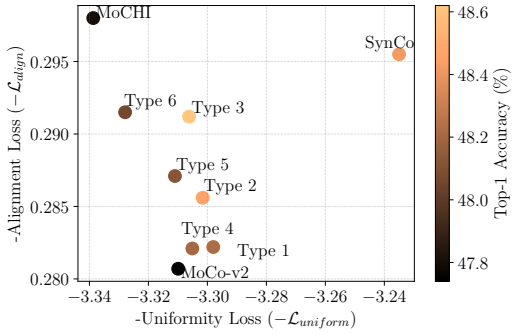


Figure 3 Alignment and uniformity on ImageNet-100. Comparison of MoCo-v2, MoCHI, and SynCo. The x- and y-axis represent $-\mathcal{L}_{\text{uniform}}$ and $-\mathcal{L}_{\text{align}}$, respectively. The model with the highest performance is located in the upper-right corner of the chart.

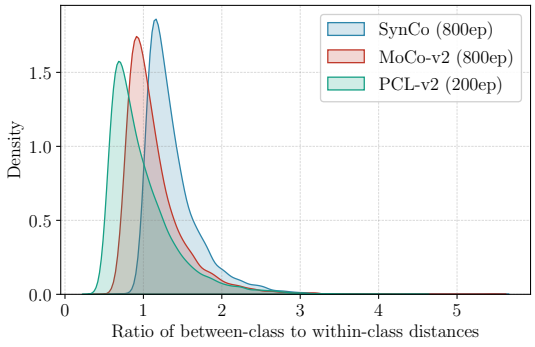


Figure 4 Distribution of the ratio between inter-class and intra-class distances for MoCo-based methods on ImageNet ILSVRC-2012. Higher values indicate better class separation. For clarity, we only show MoCo-v2 (800 ep.), PCL-v2 (200 ep.), and SynCo (800 ep.).

6.2 Evaluating the Usage of the Representation Space

To assess learned representations, we employ alignment and uniformity metrics (Wang and Isola, 2020). These metrics provide insights into representation space utilization, with alignment quantifying grouping of similar samples and uniformity measuring spread across the hypersphere. Figure 3 presents results for MoCo-based methods. Our findings demonstrate that SynCo significantly improves representation uniformity compared to MoCo-v2 and MoCHI, showing improved utilization of the representation space. Furthermore, incorporating synthetic negatives (types 1 to 6) leads to improved alignment. These results suggest that SynCo's approach yields stronger and more well-distributed feature representations.

6.3 Class Concentration Analysis

To quantify the learned latent space structure, we examine the relationship between within- and between-class distances. Figure 4 (and Section C.5, Figure 5) shows the distribution of ratios between inter- and intra-class

ℓ_2-distances for representations learned by MoCo-based contrastive methods on the ImageNet validation set. A higher mean ratio indicates better concentration within classes while maintaining greater separation between classes, reflecting improved linear separability (aligned with Fisher's linear discriminant analysis (Friedman et al., 2009)). After 800 training epochs, SynCo achieves a mean ratio of 1.384, surpassing MoCo-v2 (1.146) and PCL-v2 (0.988).

7 Conclusion

This paper introduces SynCo, leveraging synthetic hard negatives to improve contrastive visual representation learning. By generating diverse negatives "on-the-fly" using six complementary strategies, SynCo demonstrates consistent improvements without significant computational overhead. The proposed strategies are general and applicable to any contrastive method utilizing InfoNCE loss, *e.g.*, SimCLR (Chen et al., 2020a).

Scope and fair comparison. We deliberately focus on MoCo-based comparisons to ensure fair evaluation under identical training conditions. Recent self-distillation methods like DINO-v2/v3 (Oquab et al., 2023; Siméoni et al., 2025) or iBOT (Zhou et al., 2022) operate in fundamentally different paradigms, *i.e.*, they use teacher–student architectures, multiple crops, and train on billions of samples with larger computational budgets. Comparing SynCo to these methods would conflate the benefits of synthetic negatives with differences in architecture, data scale, and training. Our contribution specifically targets improving contrastive learning through diverse negative sampling. Within this scope, the consistent improvements across downstream tasks validate our approach (Tables 1 to 5), since small gains in self-supervised learning often translate to significant benefits when deployed at scale (*cf.* Section E).

Acknowledgments

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

References

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.

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

Muhammad Awais, Muzammal Naseer, Salman Khan, Rao Muhammad Anwer, Hisham Cholakkal, Mubarak Shah, Ming-Hsuan Yang, and Fahad Shahbaz Khan. Foundational models defining a new era in vision: A survey and outlook, 2023. https://arxiv.org/abs/2307.13721.

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

Randall Balestriero, Mark Ibrahim, Vlad Sobal, Ari Morcos, Shashank Shekhar, Tom Goldstein, Florian Bordes, Adrien Bardes, Gregoire Mialon, Yuandong Tian, Avi Schwarzschild, Andrew Gordon Wilson, Jonas Geiping, Quentin Garrido, Pierre Fernandez, Amir Bar, Hamed Pirsiavash, Yann LeCun, and Micah Goldblum. A cookbook of self-supervised learning, 2023.

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

Adrien Bardes, Jean Ponce, and Yann LeCun. Vicreg: Variance-invariance-covariance regularization for self-supervised learning, 2022a.

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

Rishi Bommasani, Drew A. Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S. Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, Erik Brynjolfsson, S. Buch, Dallas Card, Rodrigo Castellon,

Niladri S. Chatterji, Annie S. Chen, Kathleen A. Creel, Jared Davis, Dora Demszky, Chris Donahue, Moussa Doumbouya, Esin Durmus, Stefano Ermon, John Etchemendy, Kawin Ethayarajh, Li Fei-Fei, Chelsea Finn, Trevor Gale, Lauren E. Gillespie, Karan Goel, Noah D. Goodman, Shelby Grossman, Neel Guha, Tatsunori Hashimoto, Peter Henderson, John Hewitt, Daniel E. Ho, Jenny Hong, Kyle Hsu, Jing Huang, Thomas F. Icard, Saahil Jain, Dan Jurafsky, Pratyusha Kalluri, Siddharth Karamcheti, Geoff Keeling, Fereshte Khani, O. Khattab, Pang Wei Koh, Mark S. Krass, Ranjay Krishna, Rohith Kuditipudi, Ananya Kumar, Faisal Ladhak, Mina Lee, Tony Lee, Jure Leskovec, Isabelle Levent, Xiang Lisa Li, Xuechen Li, Tengyu Ma, Ali Malik, Christopher D. Manning, Suvir P. Mirchandani, Eric Mitchell, Zanele Munyikwa, Suraj Nair, Avanika Narayan, Deepak Narayanan, Benjamin Newman, Allen Nie, Juan Carlos Niebles, Hamed Nilforoshan, J. F. Nyarko, Giray Ogut, Laurel Orr, Isabel Papadimitriou, Joon Sung Park, Chris Piech, Eva Portelance, Christopher Potts, Aditi Raghunathan, Robert Reich, Hongyu Ren, Frieda Rong, Yusuf H. Roohani, Camilo Ruiz, Jack Ryan, Christopher R'e, Dorsa Sadigh, Shiori Sagawa, Keshav Santhanam, Andy Shih, Krishna Parasuram Srinivasan, Alex Tamkin, Rohan Taori, Armin W. Thomas, Florian Tramèr, Rose E. Wang, William Wang, Bohan Wu, Jiajun Wu, Yuhuai Wu, Sang Michael Xie, Michihiro Yasunaga, Jiaxuan You, Matei A. Zaharia, Michael Zhang, Tianyi Zhang, Xikun Zhang, Yuhui Zhang, Lucia Zheng, Kaitlyn Zhou, and Percy Liang. On the opportunities and risks of foundation models. *ArXiv*, 2021. https://crfm.stanford.edu/assets/report.pdf.

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.

Mathilde Caron, Ishan Misra, Julien Mairal, Priya Goyal, Piotr Bojanowski, and Armand Joulin. Unsupervised learning of visual features by contrasting cluster assignments. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 9912–9924. Curran Associates, Inc., 2020. https://proceedings.neurips.cc/paper_files/paper/2020/file/70feb62b69f16e0238f741fab228fec2-Paper.pdf.

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

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. https://dl.acm.org/doi/10.5555/3008751.3008802.

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

Ting Chen, Simon Kornblith, Kevin Swersky, Mohammad Norouzi, and Geoffrey Hinton. Big self-supervised models are strong semi-supervised learners, 2020b.

Xinlei Chen and Kaiming He. Exploring simple siamese representation learning, 2020.

Xinlei Chen, Haoqi Fan, Ross Girshick, and Kaiming He. Improved baselines with momentum contrastive learning, 2020c.

Xinlei Chen, Saining Xie, and Kaiming He. An empirical study of training self-supervised vision transformers, 2021.

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.

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

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. https://api.semanticscholar.org/CorpusID:57246310.

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.

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.

Debidatta Dwibedi, Yusuf Aytar, Jonathan Tompson, Pierre Sermanet, and Andrew Zisserman. With a little help from my friends: Nearest-neighbor contrastive learning of visual representations, 2021.

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

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

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, September 2009. doi: 10.1007/s11263-009-0275-4. https://doi.org/10.1007/s11263-009-0275-4.

Jerome Friedman, Trevor Hastie, and Robert Tibshirani. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer Series in Statistics. Springer, New York, second edition, 2009. ISBN 978-0-387-84857-0. doi: 10.1007/978-0-387-84858-7.

Songwei Ge, Shlok Mishra, Haohan Wang, Chun-Liang Li, and David Jacobs. Robust contrastive learning using negative samples with diminished semantics, 2022. https://arxiv.org/abs/2110.14189.

Nikolaos Giakoumoglou, Tania Stathaki, and Athanasios Gkelias. A review on discriminative self-supervised learning methods in computer vision, 2025. https://arxiv.org/abs/2405.04969.

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

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

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.

R. Hadsell, S. Chopra, and Y. LeCun. Dimensionality reduction by learning an invariant mapping. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 2, pages 1735–1742, 2006. doi: 10.1109/CVPR.2006.100.

Jeff Z. HaoChen, Colin Wei, Adrien Gaidon, and Tengyu Ma. Provable guarantees for self-supervised deep learning with spectral contrastive loss, 2022. https://arxiv.org/abs/2106.04156.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015.

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

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

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

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*, 2021a.

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

R Devon Hjelm, Alex Fedorov, Samuel Lavoie-Marchildon, Karan Grewal, Phil Bachman, Adam Trischler, and Yoshua Bengio. Learning deep representations by mutual information estimation and maximization, 2019.

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.

Ahmet Iscen, Giorgos Tolias, Yannis Avrithis, and Ondrej Chum. Mining on manifolds: Metric learning without labels. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7842–7851, 2018.

Ruijie Jiang, Thuan Nguyen, Shuchin Aeron, and Prakash Ishwar. Hard-negative sampling for contrastive learning: Optimal representation geometry and neural- vs dimensional-collapse. *Transactions on Machine Learning Research*, 2025. ISSN 2835-8856. https://openreview.net/forum?id=3cnpZ5SIjU.

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

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

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

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

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

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

Alex Krizhevsky. Learning multiple layers of features from tiny images. pages 32–33, 2009. https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf.

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

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. https://arxiv.org/abs/2212.04825.

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.

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.

Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A convnet for the 2020s, 2022. https://arxiv.org/abs/2201.03545.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization, 2019.

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.

Leland McInnes, John Healy, and James Melville. Umap: Uniform manifold approximation and projection for dimension reduction, 2020. https://arxiv.org/abs/1802.03426.

Mohammad Mehrabi, Adel Javanmard, Ryan A. Rossi, Anup Rao, and Tung Mai. Fundamental tradeoffs in distributionally adversarial training. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 7544–7554. PMLR, 18–24 Jul 2021. https://proceedings.mlr.press/v139/mehrabi21a.html.

Anastasiya Mishchuk, Dmytro Mishkin, Filip Radenovic, and Jiri Matas. Working hard to know your neighbor's margins: Local descriptor learning loss. In *Advances in Neural Information Processing Systems*, pages 4826–4837, 2017.

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

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

Vinod Nair and Geoffrey Hinton. Rectified linear units improve restricted boltzmann machines vinod nair. volume 27, pages 807–814, 06 2010.

Maxime Oquab, Timothée Darcet, Theo Moutakanni, Huy V. Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, Russell Howes, Po-Yao Huang, Hu Xu, Vasu Sharma, Shang-Wen Li, Wojciech Galuba, Mike Rabbat, Mido Assran, Nicolas Ballas, Gabriel Synnaeve, Ishan Misra, Herve Jegou, Julien Mairal, Patrick Labatut, Armand Joulin, and Piotr Bojanowski. Dinov2: Learning robust visual features without supervision, 2023.

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. https://arxiv.org/abs/1912.01703.

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

Colorado J Reed, Sean Metzger, Aravind Srinivas, Trevor Darrell, and Kurt Keutzer. Selfaugment: Automatic augmentation policies for self-supervised learning, 2021. https://arxiv.org/abs/2009.07724.

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

Joshua Robinson, Ching-Yao Chuang, Suvrit Sra, and Stefanie Jegelka. Contrastive learning with hard negative samples, 2021. https://arxiv.org/abs/2010.04592.

Renan A. Rojas-Gomez, Karan Singhal, Ali Etemad, Alex Bijamov, Warren R. Morningstar, and Philip Andrew Mansfield. Sassl: Enhancing self-supervised learning via neural style transfer, 2024. https://arxiv.org/abs/2312.01187.

Sebastian Ruder. An overview of gradient descent optimization algorithms, 2017.

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. https://arxiv.org/abs/2202.14037.

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, October 2019. ISSN 1573-1405. doi: 10.1007/s11263-019-01228-7. http://dx.doi.org/10.1007/s11263-019-01228-7.

Oriane Siméoni, Huy V. Vo, Maximilian Seitzer, Federico Baldassarre, Maxime Oquab, Cijo Jose, Vasil Khalidov, Marc Szafraniec, Seungeun Yi, Michaël Ramamonjisoa, Francisco Massa, Daniel Haziza, Luca Wehrstedt, Jianyuan Wang, Timothée Darcet, Théo Moutakanni, Leonel Sentana, Claire Roberts, Andrea Vedaldi, Jamie Tolan, John Brandt, Camille Couprie, Julien Mairal, Hervé Jégou, Patrick Labatut, and Piotr Bojanowski. Dinov3, 2025. https://arxiv.org/abs/2508.10104.

Jiawei Su, Danilo Vasconcellos Vargas, and Kouichi Sakurai. One pixel attack for fooling deep neural networks. volume 23, pages 828–841. IEEE, 2019.

Yonglong Tian, Dilip Krishnan, and Phillip Isola. Contrastive multiview coding, 2020a.

Yonglong Tian, Chen Sun, Ben Poole, Dilip Krishnan, Cordelia Schmid, and Phillip Isola. What makes for good views for contrastive learning? In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 6827–6839. Curran Associates, Inc., 2020b. https://proceedings.neurips.cc/paper_files/paper/2020/file/4c2e5eaae9152079b9e95845750bb9ab-Paper.pdf.

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.

Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention, 2021. https://arxiv.org/abs/2012.12877.

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

Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9 (86):2579–2605, 2008. http://jmlr.org/papers/v9/vandermaaten08a.html.

Vladimir N. Vapnik. *Statistical Learning Theory*. Wiley, 1998.

Vikas Verma, Alex Lamb, Christopher Beckham, Amir Najafi, Ioannis Mitliagkas, David Lopez-Paz, and Yoshua Bengio. Manifold mixup: Better representations by interpolating hidden states. *arXiv preprint arXiv:1806.05236*, 2018.

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.

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.

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.

Xiao Wang and Guo-Jun Qi. Contrastive learning with stronger augmentations, 2022.

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.

Fangyun Wei, Yue Gao, Zhirong Wu, Han Hu, and Stephen Lin. Aligning pretraining for detection via object-level contrastive learning, 2021. https://arxiv.org/abs/2106.02637.

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

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

Tete Xiao, Colorado J Reed, Xiaolong Wang, Kurt Keutzer, and Trevor Darrell. Region similarity representation learning, 2021.

Enze Xie, Jian Ding, Wenhai Wang, Xiaohang Zhan, Hang Xu, Peize Sun, Zhenguo Li, and Ping Luo. Detco: Unsupervised contrastive learning for object detection, 2021a. https://arxiv.org/abs/2102.04803.

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

Ceyuan Yang, Zhirong Wu, Bolei Zhou, and Stephen Lin. Instance localization for self-supervised detection pretraining, 2021. https://arxiv.org/abs/2102.08318.

Zhen Yang, Ming Ding, Tinglin Huang, Yukuo Cen, Junshuai Song, Bin Xu, Yuxiao Dong, and Jie Tang. Does negative sampling matter? a review with insights into its theory and applications, 2024. https://arxiv.org/abs/2402.17238.

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

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.

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

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.

Chaoning Zhang, Kang Zhang, Trung X. Pham, Axi Niu, Zhinan Qiao, Chang D. Yoo, and In So Kweon. Dual temperature helps contrastive learning without many negative samples: Towards understanding and simplifying moco, 2022a.

Hongyi Zhang, Moustapha Cisse, Yann N. Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412*, 2017a. https://arxiv.org/abs/1710.09412.

Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. Mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412*, 2017b.

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*, 2022b.

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

Jinghao Zhou, Chen Wei, Huiyu Wang, Wei Shen, Cihang Xie, Alan Yuille, and Tao Kong. ibot: Image bert pre-training with online tokenizer, 2022.

Jiachen Zhu, Rafael M. Moraes, Serkan Karakulak, Vlad Sobol, Alfredo Canziani, and Yann LeCun. Tico: Transformation invariance and covariance contrast for self-supervised visual representation learning, 2022.

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

Appendix

A 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).

B Implementation Details

We implement SynCo in PyTorch (Paszke et al., 2019) following the implementation of MoCo[1]. Specifically, we follow the same setting as MoCo-v2.

B.1 Pretraining

Datasets. We evaluate the proposed method on ImageNet ILSVRC-2012[2] (Deng et al., 2009), which includes 1000 classes and is commonly used in previous self-supervised methods (Chen et al., 2020a; Chen and He, 2020; Zbontar et al., 2021; Zhang et al., 2022b). The dataset consists of 1.28 million training images and 50,000 validation images. We also conduct ablation studies on ImageNet-100 (Khosla et al., 2021), 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 (He et al., 2015; Zbontar et al., 2021).

Augmentation. Each input image is transformed twice to generate two different views. For SynCo, we use the same augmentation as used in (Chen et al., 2020c) and (Kalantidis et al., 2020) 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 (Chen et al., 2020c): the hidden layers of the MLP are 2048-d and are with ReLU (Nair and Hinton, 2010); the output layer of the MLP is 128-d, without ReLU.

Optimization. We follow the same setting as (Chen et al., 2020c). We utilize the SGD optimizer (Ruder, 2017) with a base learning rate of 0.03 ($= 0.03 \times$ batch_size/256), where we scale the learning rate with the batch size as in (Chen et al., 2020a), and a weight decay of 10^{-4}. The training schedule begins with a warm-up period during the first 10 epochs in which the learning rate linearly increases from 0 to the base learning rate. Following this, the learning rate gradually decreases to zero following a cosine decay schedule without restarts. The batch size for ImageNet is set to 256 distributed over 4 NVIDIA 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 D), 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 D. 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

[1]Available at: https://github.com/facebookresearch/moco.
[2]Available at: https://www.image-net.org/.

Table 6 Architecture and optimization hyperparameters used for SynCo pretraining. This table lists all architectural, optimization, MoCo, and SynCo-specific settings used during contrastive pretraining.

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	∼43 hours/100 epochs
MoCo	
Queue size K	65536
Momentum m	0.999
Temperature τ	0.2
SynCo	
Top-N hardest negatives	1024
Synthetic N_i, $i = 1, 2, 3$	256
Synthetic N_i, $i = 4, 5, 6$	64
Hyperparameters σ, δ, η	0.01

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 6.

B.2 Linear Evaluation

We follow the linear evaluation protocol of (He et al., 2020) and as in (Kornblith et al., 2019; Kolesnikov et al., 2019; Chen et al., 2020a; Grill et al., 2020; van den Oord et al., 2019), 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 (Cubuk et al., 2019), tclip (Bachman et al., 2019), 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.

B.3 Semi-supervised Training

We follow the semi-supervised learning protocol of (Chen et al., 2020a; Kornblith et al., 2019; Zhai et al., 2020; Grill et al., 2020). 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× 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 (Cubuk et al., 2019), tclip (Bachman et al., 2019), or logits rescaling. Similar to (Caron et al., 2020), 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.

B.4 Object Detection

We follow the object detection protocol of (He et al., 2020; Chen et al., 2020c). We first initialize the network with the parameters of the pretrained representation, and fine-tune it on PASCAL VOC (Everingham et al., 2009)[3] and COCO (Lin et al., 2015)[4] datasets. During training, we apply spatial augmentations, specifically random resizing and random horizontal flipping. During testing, images are resized to a fixed size of 800 pixels along the shorter side. The R50-C4 backbones, similar to those used in Detectron2 (Wu et al., 2019), 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 (Ren et al., 2016) 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 (He et al., 2018) 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.

B.5 Alignment and Uniformity

We follow the protocol of (Kalantidis et al., 2020) but training the network 100 epochs on ImageNet-100. We calculate the alignment and uniformity based on (Wang and Isola, 2020). The alignment loss $\mathcal{L}_{\text{align}}$ and uniformity loss $\mathcal{L}_{\text{uniform}}$ are computed as follows:

$$\mathcal{L}_{\text{align}}(\mathbf{x}, \mathbf{y}) = \mathbb{E}_{(\mathbf{x},\mathbf{y})\sim p_{\text{data}}} \left[\|f_q(\mathbf{x}) - f_k(\mathbf{y})\|_2^{\alpha}\right] \tag{10}$$

$$\mathcal{L}_{\text{uniform}}(\mathbf{x}) = \log \mathbb{E}_{\mathbf{x},\mathbf{y}\sim p_{\text{data}}} \left[\exp(-t\|f_q(\mathbf{x}) - f_k(\mathbf{y})\|_2^2)\right] \tag{11}$$

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].

B.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 (Tian et al., 2020a). The list is shown in Table 7.

[3]Available at: https://host.robots.ox.ac.uk/pascal/VOC/.
[4]Available at: https://cocodataset.org/.
[5]Available at: https://github.com/Ssnl/align_uniform.
[6]Available at: https://github.com/HobbitLong/CMC/blob/master/imagenet100.txt.

Table 7 List of classes from ImageNet-100. These classes are randomly sampled from the original ImageNet ILSVRC-2012 dataset.

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

Table 8 Parameters used to generate image augmentations. This table lists the augmentation probabilities and intensity settings used for MoCo-v2 pretraining.

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

B.7 Image Augmentations

During self-supervised training, SynCo uses the same augmentation as (Chen et al., 2020c). The augmentation parameters are detailed in Table 8.

C Additional Results

In this section, we present extended results including comparison with state-of-the-art methods (Section C.1), transfer to object detection on PASCAL VOC (Section C.2), linear evaluation across training epochs (Section C.3), fine-tuning with varying label fractions (Section C.4), class concentration analysis (Section C.5), robustness and out-of-distribution evaluation (Section C.6), adversarial robustness (Section C.7), extension to vision transformers (Section C.8), and qualitative analyses via t-SNE (Section C.9), GradCAM (Section C.10), UMAP (Section C.11), and nearest neighbor retrieval (Section C.12).

C.1 Comparison with State-of-the-Art Methods

We present a comprehensive comparison with state-of-the-art self-supervised learning methods in Table 9, including instance discrimination methods (SimCLR (Chen et al., 2020a), PIRL (Misra and van der Maaten, 2019)), momentum-based (BYOL (Grill et al., 2020), MoCo variants), clustering-based (SwAV (Caron et al., 2020)), redundancy reduction (Barlow Twins (Zbontar et al., 2021), VICReg (Bardes et al., 2022a)), and various hard negative mining strategies (MoCHI (Kalantidis et al., 2020), DCL (Yeh et al., 2022), AdCo (Hu et al., 2021)). It is important to note that methods such as BYOL (Grill et al., 2020), Barlow Twins (Zbontar et al., 2021), SwAV (Caron et al., 2020), DINO (Caron et al., 2021), SimCLR-v2 (Chen et al., 2020b), AdCo (Hu et al., 2021), and VICReg (Bardes et al., 2022a) 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 (Caron et al., 2020), 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 (He et al., 2020), MoCo-v2 (Chen et al., 2020c), MoCHI (Kalantidis et al., 2020), PCL (Li et al., 2021), DCL (Yeh et al., 2022)), 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 E.

C.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 10. 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.

C.3 Linear Evaluation

Table 11 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

Table 9 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 underlined. **Symbols**: ‡ With multi-crop augmentation. † We stop generating synthetic negatives at epoch 400.

Method	Epochs	Top-1
Supervised	200	76.5
RotNet (Gidaris et al., 2018)	200	48.9
PIRL (Misra and van der Maaten, 2019)	200	63.6
LA (Zhuang et al., 2019)	200	60.2
CMC (Tian et al., 2020a)	200	60.0
InfoMin (Tian et al., 2020b)	200	70.1
InfoMin (Tian et al., 2020b)	800	73.0
SimSiam (Chen and He, 2020)	100	68.3
MSF (Koohpayegani et al., 2021)	200	72.4
ReSSL (Zheng et al., 2021)	200	69.9
ReSSL (Zheng et al., 2021)‡	200	74.7
AdCo (Hu et al., 2021)	200	68.6
AdCo (Hu et al., 2021)‡	800	**75.7**
SimCLR + DCL (Yeh et al., 2022)	200	65.8
SimCLR + DCLW (Yeh et al., 2022)	200	66.9
SimCLR (Chen et al., 2020a)	1000	69.3
BYOL (Grill et al., 2020)	1000	74.3
W-MSE (Ermolov et al., 2021)	400	72.5
DINO (Caron et al., 2021)‡	800	<u>75.3</u>
SwAV (Caron et al., 2020)	800	71.8
SwAV (Caron et al., 2020)‡	800	<u>75.3</u>
Barlow Twins (Zbontar et al., 2021)	1000	73.2
ReLIC (Mitrovic et al., 2020)	1000	70.3
VICReg (Bardes et al., 2022a)	1000	73.2
CLSA (Wang and Qi, 2022)	800	72.2
Mixed Barlow Twins (Bandara et al., 2023)	800	72.2
CaCo (Wang et al., 2023)	800	74.1
All4One (Estepa et al., 2023)	800	66.6
MoCo-based		
MoCo (He et al., 2020)	200	60.7
PCL-v1 (Li et al., 2021)	200	61.5
MoCo-v2 (Chen et al., 2020c)	200	67.5
MoCo-v2 (Chen et al., 2020c)	800	<u>71.1</u>
MoCHI (Kalantidis et al., 2020)	200	66.9
MoCHI (Kalantidis et al., 2020)	800	68.7
PCL-v2 (Li et al., 2021)	200	67.6
MoCo-v2 + DCL (Yeh et al., 2022)	200	67.6
MoCo-v2 + NS (Ge et al., 2022)	200	67.9
SynCo (ours)	200	68.1
SynCo (ours)	800	70.7
SynCo† (ours)	800	**71.6**

learning through controlled synthetic negative generation—rather than maintaining constant difficulty—is crucial for optimal representation learning.

Table 10 Results for object detection on PASCAL VOC. The values in bold indicate the maximum of each column.

Method	Epochs	AP	AP_{50}	AP_{75}
Supervised	90	53.5	81.3	58.8
MoCo (He et al., 2020)	200	55.9	81.5	62.6
MoCo-v2 (Chen et al., 2020c)	200	57.0	82.4	63.6
MoCo-v2 (Chen et al., 2020c)	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 11 Top-1 accuracy (%) during pretraining from 100 to 800 epochs. Comparison of MoCo-v2, SynCo, and SynCo† (synthetic negatives disabled after 400 epochs).

Method	100	200	300	400	500	600	700	800
MoCo-v2 (Chen et al., 2020c)	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 12 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	1%	10%	100%
MoCo-v2 (Chen et al., 2020c)	48.2	66.1	77.0
MoCHI (Kalantidis et al., 2020)	50.4	65.7	78.0
SynCo (ours)	50.8	66.6	79.0

C.4 Fine-tuning

We also evaluate SynCo's performance when fine-tuning with 100% of the labeled ImageNet data. As shown in Table 12, 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.

C.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 (Friedman et al., 2009)).

As shown in Table 13, SynCo trained for 800 epochs achieves the highest mean ratio (1.384) among all MoCo-based methods, approaching and slightly surpassing the supervised baseline (1.381). A higher mean ratio indicates better class separability, which is crucial for downstream classification tasks. This superior performance can be attributed to SynCo's synthetic hard negative generation strategies, which help create more discriminative feature representations.

The standard deviation of the ratio distribution provides insight into the consistency of learned features across

Table 13 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.

Method	Epochs	Mean ↑	Median ↑	Std ↓
Supervised	90	1.381	**1.369**	**0.110**
MoCo (He et al., 2020)	200	1.012	0.999	0.115
MoCo-v2 (Chen et al., 2020c)	200	1.061	0.971	0.358
MoCo-v2 (Chen et al., 2020c)	800	1.146	1.043	0.375
PCL-v1 (Li et al., 2021)	200	0.930	0.869	0.312
PCL-v2 (Li et al., 2021)	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

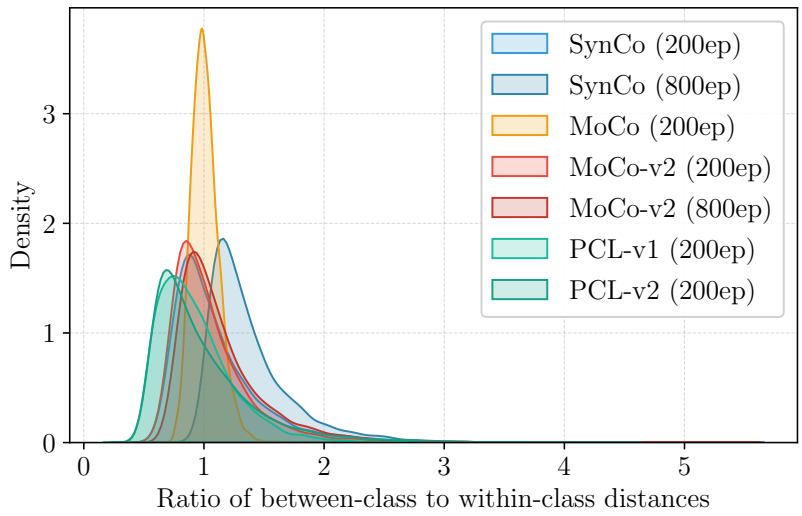


Figure 5 Distribution of the ratio between inter-class and intra-class distances for different MoCo-based methods. Higher values indicate better class separation. We show MoCo (He et al., 2020), MoCo-v2 (Chen et al., 2020c) (200 and 800 epochs), PCL-v1 and PCL-v2 (Li et al., 2021) (200 epochs), and SynCo (200 and 800 epochs).

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 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.

C.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 (Recht et al., 2019), which comprises three sets of 10,000 images (matched frequency, threshold 0.7, and top images); ImageNet-A (Hendrycks et al., 2021b), which contains naturally adversarial examples; ImageNet-C (Hendrycks and Dietterich, 2019), which consists of 15 synthetically generated corruptions (*e.g.*, blur, noise, weather); ImageNet-

Table 14 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 (Tomasev et al., 2022). We reproduce MoCo and MoCo-v2 linear probing since no checkpoints are available (thus results may differ from original implementation).

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 (Chen et al., 2020a)	1000	69.3	53.2	61.7	68.0	31.1	-	-	3.9	18.3	-
MoCo (He et al., 2020)	200	60.9	45.9	53.8	60.4	33.8	2.5	38.5	10.2	18.2	85.9
MoCo-v2 (Chen et al., 2020c)	200	67.8	54.8	63.0	69.0	51.4	2.8	44.2	17.5	27.8	81.9
MoCo-v2 (Chen et al., 2020c)	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 15 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	Noise			Blur				Weather				Digital			
		Ga	Sh	Im	De	Gl	Mo	Zo	Fr	Sn	Fo	Br	Co	El	Pi	JP
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 (Chen et al., 2020a)	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 (He et al., 2020)	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 (Chen et al., 2020c)	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 (Chen et al., 2020c)	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

Watermark (Li et al., 2023), testing robustness to watermark perturbations. For OOD generalization, we examine performance on five datasets: ImageNet-Sketch (Wang et al., 2019), containing 50,000 black-and-white sketches; ImageNet-R (Hendrycks et al., 2021a), consisting of 30,000 artistic renditions; ImageNet-O (Hendrycks et al., 2021b), 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 14, 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 15 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.

Table 16 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.

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 (He et al., 2020)	200	60.86	15.75	0.08	9.40	9.98	0.05	7.33	57.40
MoCo-v2 (Chen et al., 2020c)	200	67.77	23.75	0.32	17.08	13.94	0.23	5.36	65.16
MoCo-v2 (Chen et al., 2020c)	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

C.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 (Madry et al., 2018), 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 (Kim, 2020)[7], with evaluations conducted using a ResNet-50 backbone.

Evaluation protocol. Our evaluation includes gradient-based attacks: Fast Gradient Sign Method (FGSM) (Goodfellow et al., 2014) with $\varepsilon = 8/255$, and Projected Gradient Descent (PGD) (Madry et al., 2018) with $\varepsilon = 8/255$, $\alpha = 2/255$, and 10 steps. We also evaluate against optimization-based attacks: Carlini & Wagner (C&W) (Carlini and Wagner, 2017) 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 (Andriushchenko et al., 2020) with ℓ_∞ norm and 1,000 queries, and Auto Attack (Croce and Hein, 2020) using ℓ_∞ norm. Furthermore, we include advanced perturbation methods: Translation-Invariant FGSM (TIFGSM) (Dong et al., 2019) with $\varepsilon = 8/255$, $\alpha = 2/255$, and 10 steps, and One-Pixel Attack (Su et al., 2019) limited to single-pixel modifications with 10 steps.

Results and analysis. Results in Table 16 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.

C.8 Extending SynCo to Vision Transformers

In this section, we describe how we extend SynCo to vision transformers (Dosovitskiy et al., 2021). 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 (Xie et al., 2021b). We chose MoBY over MoCo-v3 (Chen et al., 2021) 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 (Touvron et al., 2021) or Swin-Tiny (Liu et al., 2021)) and a projection head (Chen et al., 2020a). The encoder f_q has an additional prediction head following (Grill et al., 2020). 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 (Grill et al., 2020), and a queue-based contrastive approach (He et al.,

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

Table 17 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.

Method	Arch.	Epochs	Top-1
Supervised	DeiT-S	300	79.8
Supervised	Swin-T	300	81.3
DINO (Caron et al., 2021)	DeiT-S	300	72.5
MoCo-v3 (Chen et al., 2021)	DeiT-S	300	72.5
MoBY (Xie et al., 2021b)	DeiT-S	300	72.8
MoBY (Xie et al., 2021b)	Swin-T	300	75.0
SynCo (ours)	DeiT-S	300	73.0
SynCo (ours)	Swin-T	300	75.2

2020). 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 (Loshchilov and Hutter, 2019) 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.

Results and analysis. As shown in Table 17, our method achieves consistent improvements over the MoBY baseline across both DeiT-S (Touvron et al., 2021) and Swin-T (Liu et al., 2021) architectures, while also outperforming other self-supervised approaches like DINO (Caron et al., 2021) and MoCo-v3 (Chen et al., 2021). 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.

C.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 (van der Maaten and Hinton, 2008) 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 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 (He et al., 2020) with 200 epochs of pretraining (Figure 20) and MoCo-v2 (Chen et al., 2020c) 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).

C.10 GradCAM Visualization

To gain deeper insights into the regions SynCo focuses on during feature extraction, we utilize GradCAM (Selvaraju et al., 2019) 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

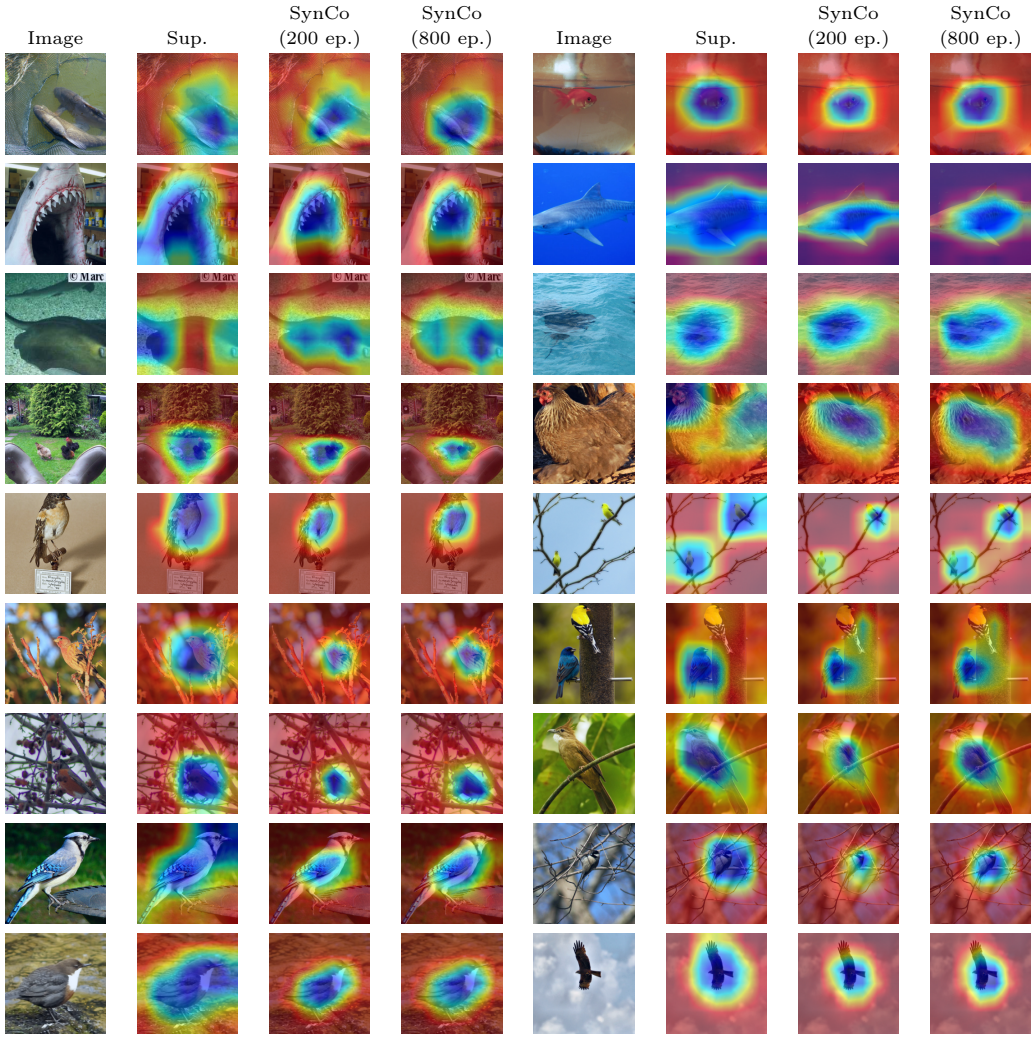


Figure 6 GradCAM visualizations of ImageNet validation set. We compare different models: supervised training and SynCo pretrained for 200 and 800 epochs.

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.

C.11 UMAP Visualization

To better understand the feature representations learned by SynCo, we perform Uniform Manifold Approximation and Projection (UMAP) (McInnes et al., 2020) 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).

Table 18 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.

	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

C.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.

D Ablation Studies

In this section, we perform ablation studies of SynCo on ImageNet-100 (Section D.1) and CIFAR-100 (Section D.2). For ImageNet-100 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.

D.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 18 to 20.

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 18, 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.

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

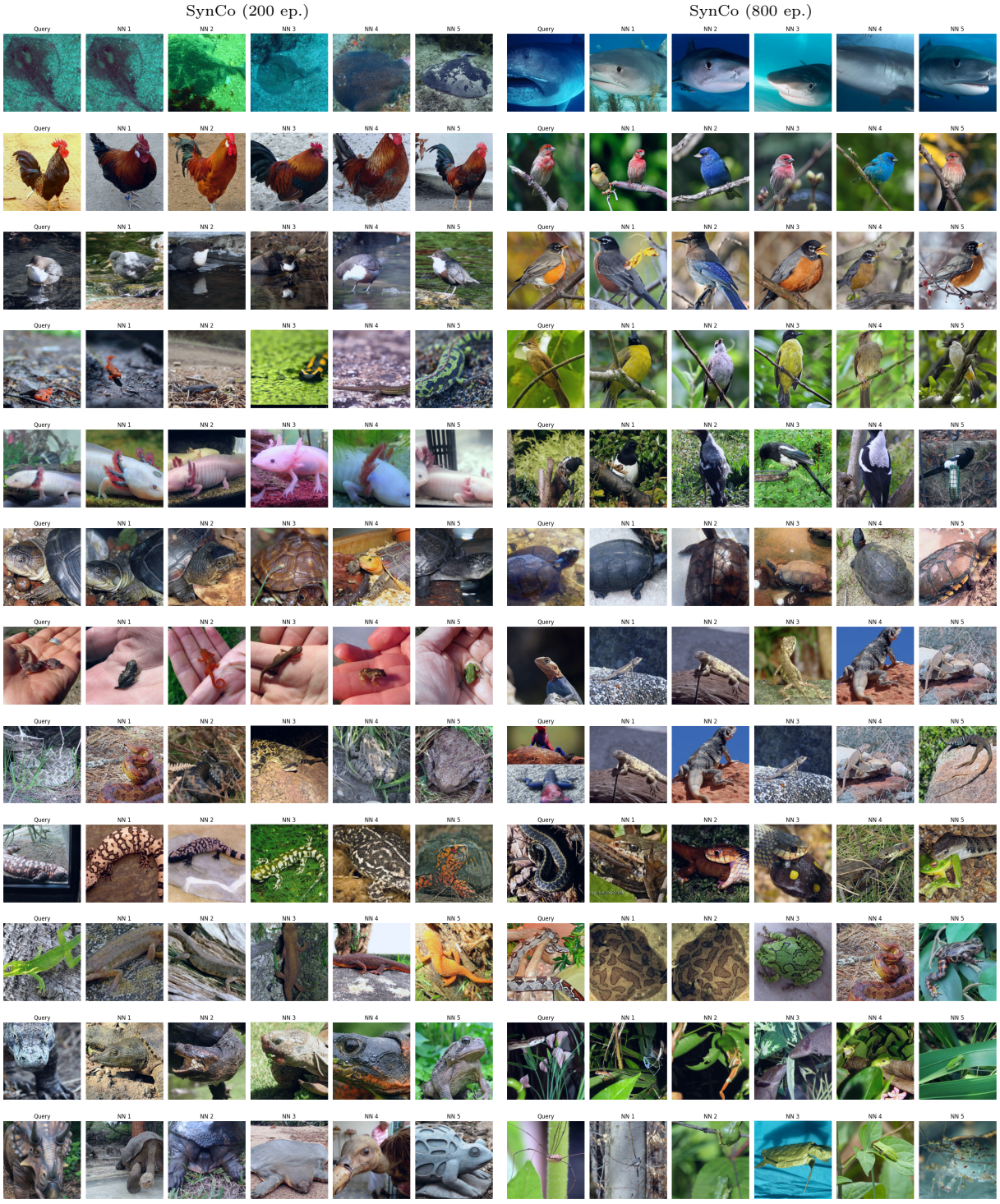


Figure 7 Visualization of nearest neighbors in the embedding space. SynCo is pretrained at 200 epochs (**left**) and 800 epochs (**right**). Each row corresponds to a query image and its top-5 nearest neighbors in the respective embedding spaces.

learning. As we can see from Table 19, 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

Table 19 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.

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 20 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.

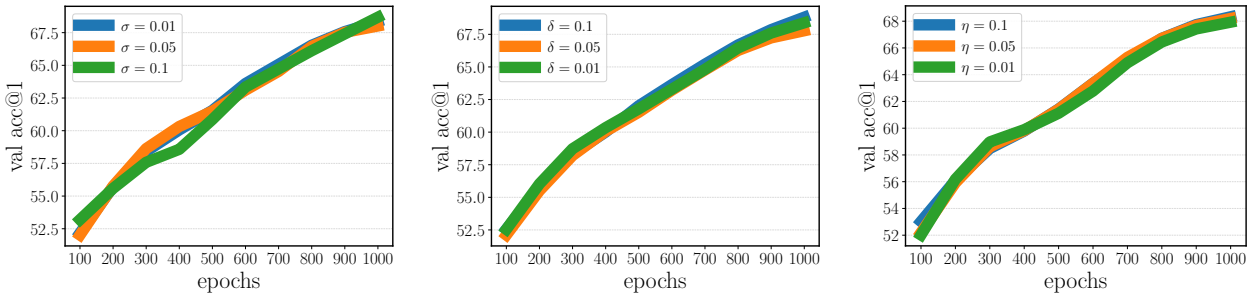
Method	Queue size K					
	4k	8k	16k	32k	65k	131k
MoCo-v2 (Chen et al., 2020c)	50.10	50.50	49.32	48.02	47.74	47.60
SynCo (ours)	48.30	48.50	49.40	48.08	48.42	48.50

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 19, 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 19, 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.

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 20, 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 (Kalantidis et al., 2020). However, as the queue increases, SynCo performs on par with MoCo-v2. At the largest queue size tested, SynCo outperforms MoCo-v2.

D.2 Ablation Study on CIFAR-100

Secondly, we perform additional ablation studies on CIFAR-100 (Krizhevsky, 2009) for 32×32 images for 100-way classification, chosen for its computational efficiency while maintaining sufficient complexity for



(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.

meaningful ablations. We ablate the same paramters as in Section D.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) (He et al., 2015) 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.

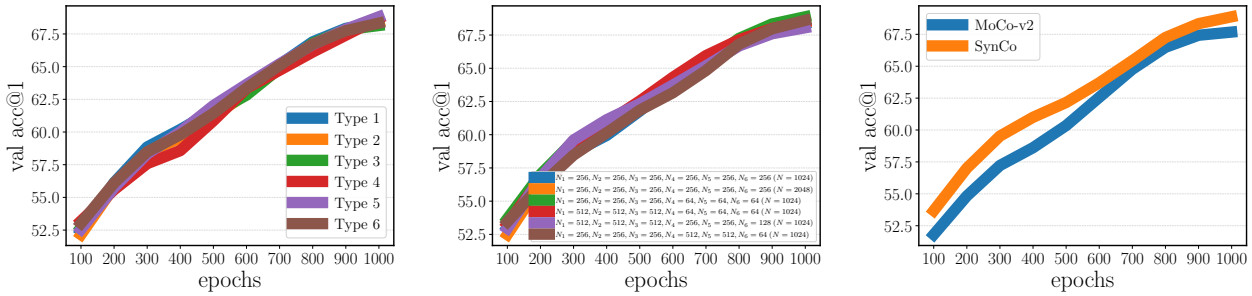
Ablation on types. We evaluate SynCo by first training without hard negatives (equivalent to MoCo-v2) and then by incorporating each type of hard negative individually, as well as in combination. Additionally, we test different configurations of the number of hard negatives (N_1 through N_6) to find 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.

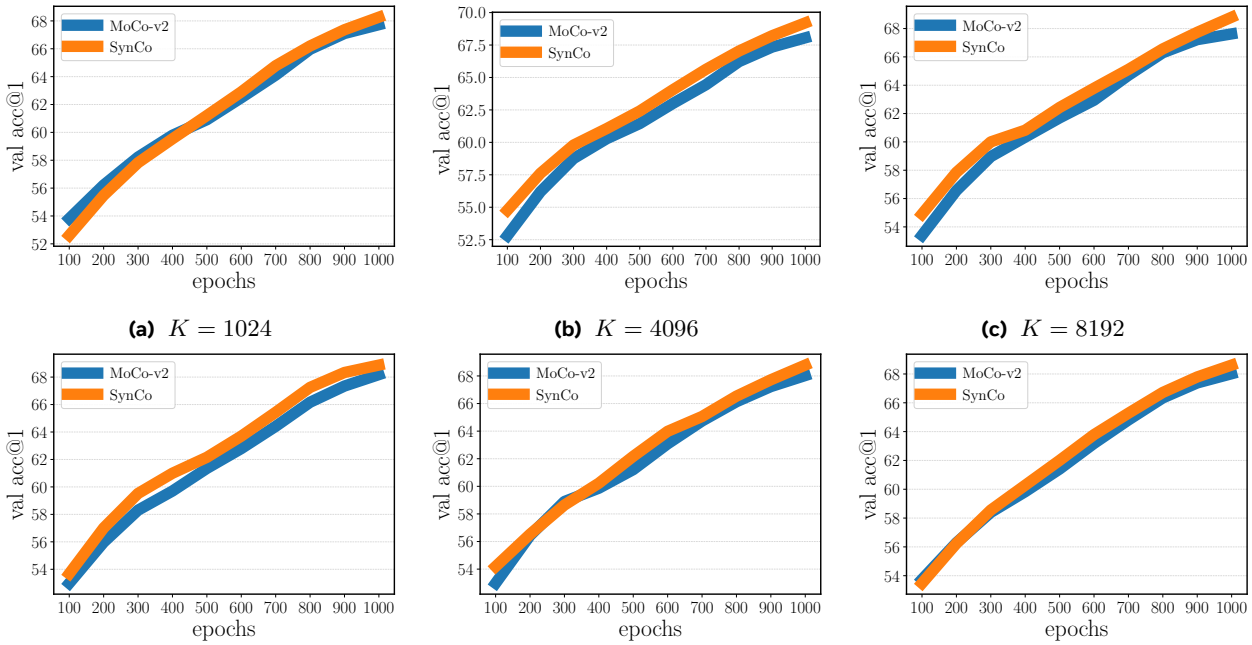
E Discussion

In this section, we discuss the scope of our framework and potential extensions to other self-supervised methods (Section E.1), the relationship with closely related approaches including MoCHI and AdCo (Section E.2), the



(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.



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

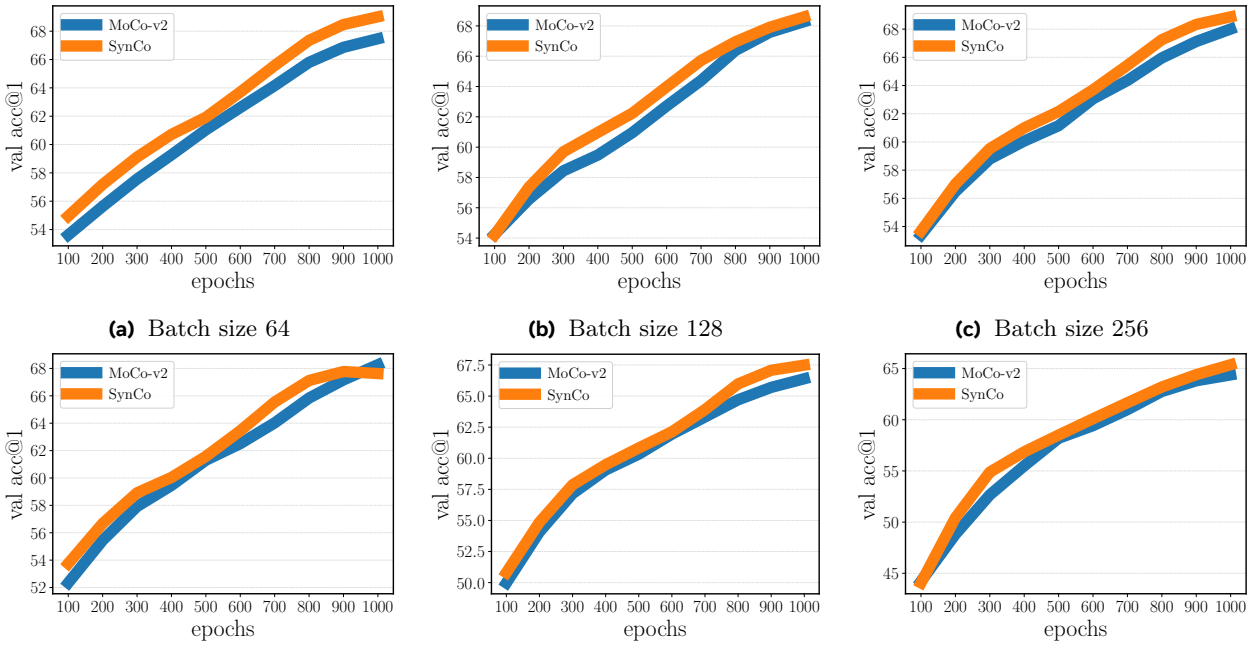
(d) $K = 16384$ **(e)** $K = 32768$ **(f)** $K = 65536$

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 $K = 65536$.

mechanisms underlying SynCo's effectiveness (Section E.3), practical considerations for regularization and hyperparameter tuning (Section E.4), and broader implications for other modalities and tasks (Section E.5).

E.1 Scope and Extensions

Methods such as BYOL (Grill et al., 2020), Barlow Twins (Zbontar et al., 2021), SwAV (Caron et al., 2020), DINO (Caron et al., 2021), SimCLR-v2 (Chen et al., 2020b), AdCo (Hu et al., 2021), and VICRegL (Bardes et al., 2022a) incorporate additional tricks including larger projection heads (*e.g.*, SimCLR-v2 uses a 3-layer MLP; DINO uses a 3-layer MLP with 2048 hidden dimensions), larger projection dimensions (*e.g.*, DINO 65k, Barlow Twins 8k), multi-crop augmentation strategies (*e.g.*, SwAV, DINO, and AdCo use 2 global views



(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.

at 224×224 and up to 8 local views at 96×96), and extended training schedules (*e.g.*, BYOL and DINO train for 800–1000 epochs), which significantly improve their performance but stem from architectural and training modifications rather than their core learning mechanisms. Our approach focuses on demonstrating the effectiveness of synthetic negative generation within a simpler framework, making comparisons against MoCo-based approaches (MoCo-v2 (Chen et al., 2020c), MoCHI (Kalantidis et al., 2020), PCL (Li et al., 2021), DCL (Yeh et al., 2022)) more equitable, as they share similar architectural choices and training procedures. Nevertheless, SynCo's principles are not limited to the MoCo framework. Integration with larger projection and prediction heads (Grill et al., 2020; Caron et al., 2021), multi-crop augmentation (Caron et al., 2020), or newer architectures such as ConvNeXt (Liu et al., 2022) could further improve performance, though at substantially higher computational cost ($>$ 8 GPUs). Similarly, synthetic negatives could be adapted to SimCLR (Chen et al., 2020a)'s in-batch negative sampling by generating negatives from in-batch features rather than memory bank features, though SimCLR's requirement for very large batch sizes (4096) makes this computationally prohibitive for our current setup. These extensions remain promising directions for future research.

E.2 Relationship with Related Approaches

MoCHI (Kalantidis et al., 2020) is the most directly related prior work, introducing feature-level mixing for hard negative samples. However, MoCHI is limited to interpolation between query and hard negatives (our Type 1) and linear combination of hard negative pairs (our Type 3), exploring only geometric transformations within the existing convex hull. SynCo extends this with four additional strategies: extrapolation beyond the decision boundary (Type 2), controlled noise injection (Type 4), and gradient-based perturbations (Types 5–6) that leverage optimization landscape information. Our ablations in Sections D.1 and D.2 confirm that the novel strategies (Types 2, 4–6) alone achieve 48.2/48.3% accuracy compared to the MoCHI-equivalent strategies (Types 1, 3) at 46.8%. AdCo (Hu et al., 2021) takes a fundamentally different approach by maintaining trainable negative adversaries updated through adversarial optimization. While both methods aim to create challenging negatives, AdCo requires a separate adversarial training loop, persistent negative representations, careful initialization and learning rate scheduling, and relies on multi-crop augmentation that increases

computational overhead by approximately 3.7×. In contrast, SynCo generates synthetic negatives on the representation space on-the-fly without additional training loops or persistent storage. Conceptually, SynCo also shares similarities with data augmentation methods like Mixup (Zhang et al., 2017a), CutMix (Yun et al., 2019), and AugMax (Wang et al., 2021), but operates in representation space rather than pixel space, enabling more semantically meaningful negative generation with significantly lower computational cost.

E.3 Understanding SynCo

Each strategy contributes to model generalization through complementary mechanisms. Types 1–3 explore geometric transformations in embedding space: interpolation increases sample diversity, extrapolation pushes representation boundaries, and mixup of hard negatives encourages robust feature learning. Types 4–6 introduce perturbation-based challenges: noise injection promotes invariance to minor fluctuations, while the gradient-based strategies refine discriminatory power near decision boundaries. The distinction between Types 5 and 6 is illustrative: given a horse (anchor) and zebra (hard negative), Type 6 applies fixed-magnitude perturbations ($\eta \cdot \text{sign}(\nabla)$) while Type 5 applies variable-magnitude perturbations ($\delta \cdot \nabla$), enabling more diverse exploration of challenging regions. These mechanisms align with theoretical foundations from contrastive learning theory (Arora et al., 2019; Saunshi et al., 2022) and spectral contrastive loss analysis (HaoChen et al., 2022), where harder negatives help prevent dimensional collapse and encourage uniform feature distributions (validated by our uniformity metrics in Figure 3). Empirically, SynCo provides the greatest benefit during early to mid-training phases where the model actively learns discriminative features, in tasks requiring strong transferability (*e.g.*, detection on COCO shows +1.0% AP^{bb} vs linear evaluation's +0.4%), and in low-data regimes (semi-supervised learning with 1% labels shows +2.6% improvement). In late training stages, adaptive strategies such as cooldown periods are important to prevent excessive task difficulty from hindering convergence.

E.4 Considerations

SynCo regularizes the network through synthetic hard negatives, aligning with vicinal risk minimization (Chapelle et al., 2000) and encouraging learning of robust features over memorization of specific patterns (Vapnik, 1998; Zhang et al., 2017a). Despite incorporating multiple hyperparameters across six strategies, the framework is designed to require minimal tuning. The perturbation parameters σ, δ, and η show minimal performance variation in the range 0.01 to 0.1 (see Section D.1), while $\alpha_{\max}$, $\beta_{\max}$, and $\gamma_{\max}$ are empirically set based on each strategy's intuition and remain fixed throughout training. For core architectural choices, the number of hardest negatives $N = 1024$ balances diversity against diminishing returns, and fixed ratios between strategy types (256 vs 64) reflect that geometric transformations benefit from more samples while perturbation-based methods saturate with fewer (see Section D.2). Our comprehensive hyperparameter analysis is primarily based on CIFAR-100 (Section D.2) with findings extended to ImageNet (Tables 1, 2 and 9); while computational constraints prevent exhaustive ablation on larger datasets, results demonstrate that SynCo achieves strong performance with default parameters across configurations.

E.5 Broader Implications

The principle of synthetic hard negative generation extends beyond image classification to other modalities and tasks. In NLP, generating challenging negatives could benefit sentence similarity and text classification; in audio, it may aid speaker recognition and event detection. Domain-specific hard negatives also create opportunities for domain adaptation, helping models generalize across distribution shifts. More broadly, our work demonstrates that classical self-supervised learning frameworks (Giakoumoglou et al., 2025) can be meaningfully strengthened through synthetic sample generation in embedding space, providing a practical path toward more data-efficient and generalizable representation learning.

F 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%).

G 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.

H 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 (He et al., 2020) codebase. We also provide the pseudocode for generating each type of synthetic negative in Section A 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 B of our paper. We closely follow the experimental protocol of MoCo-v2 (Chen et al., 2020c) to ensure fair comparison. To further support reproducibility, we have made our pretrained model checkpoints publicly available (links provided in Section F). 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.

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 (He et al., 2020) 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 (Chen et al., 2020c) 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 (Chen et al., 2020c) 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 (Paszke et al., 2019). 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.

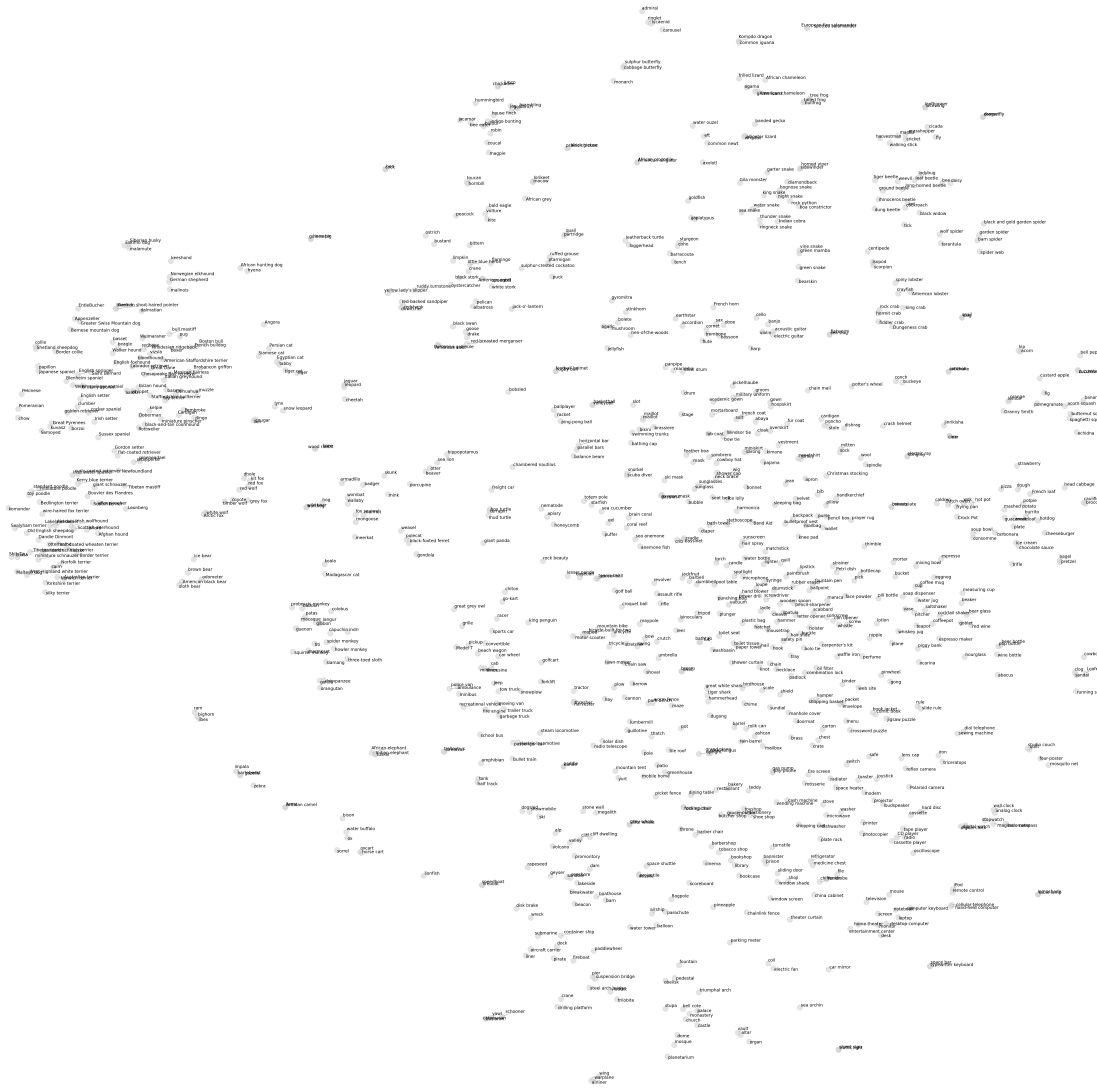


Figure 19 t-SNE visualization of ImageNet class embeddings in SynCo's feature space after 800 epochs of pretraining. Each point represents the average feature vector of validation set images for one class.

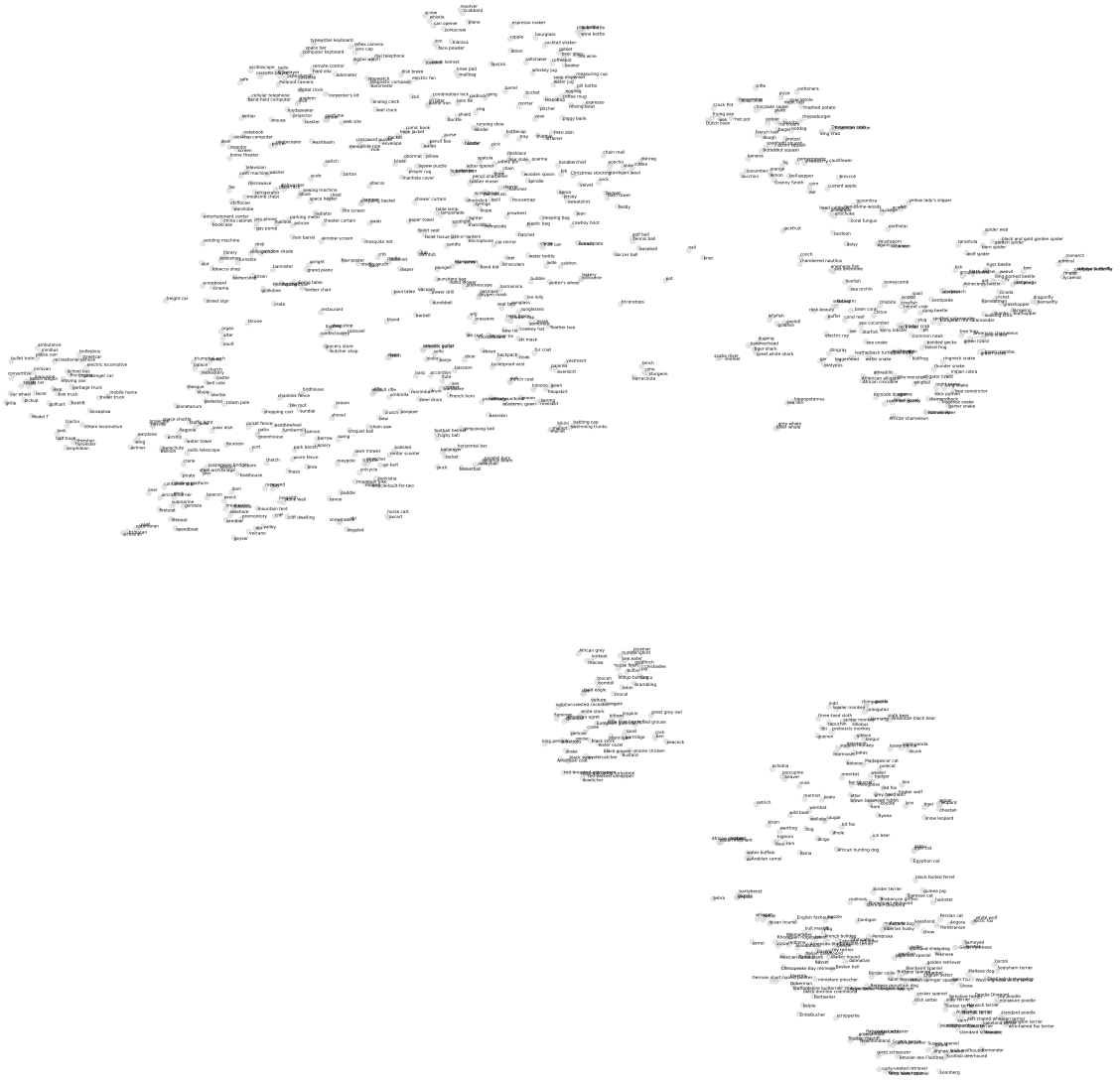


Figure 20 t-SNE visualization of ImageNet class embeddings in MoCo's (He et al., 2020) feature space after 200 epochs of pretraining. Each point represents the average feature vector of validation set images for one class.

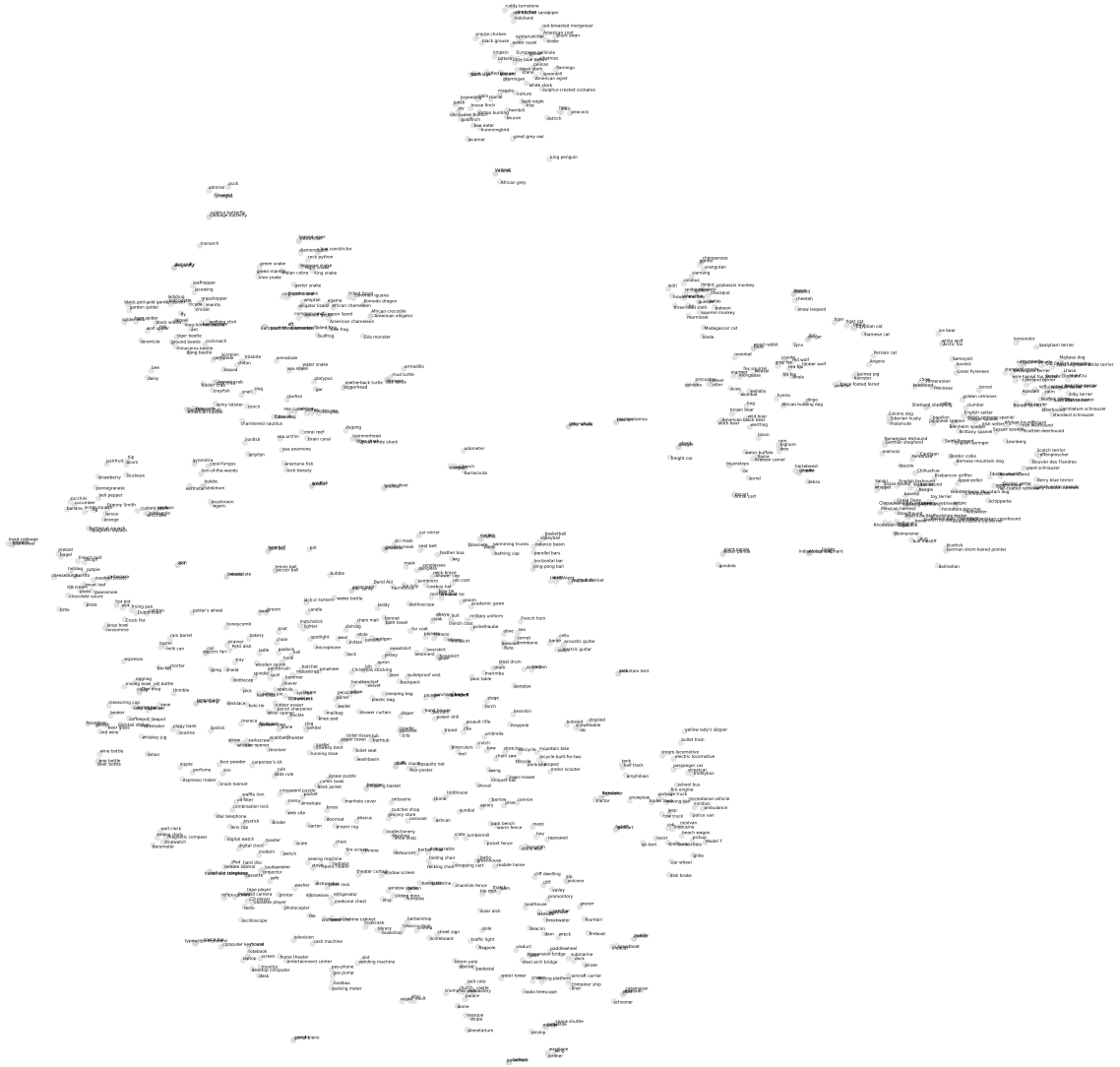


Figure 21 t-SNE visualization of ImageNet class embeddings in MoCo-v2's (Chen et al., 2020c) feature space after 200 epochs of pretraining. Each point represents the average feature vector of validation set images for one class.

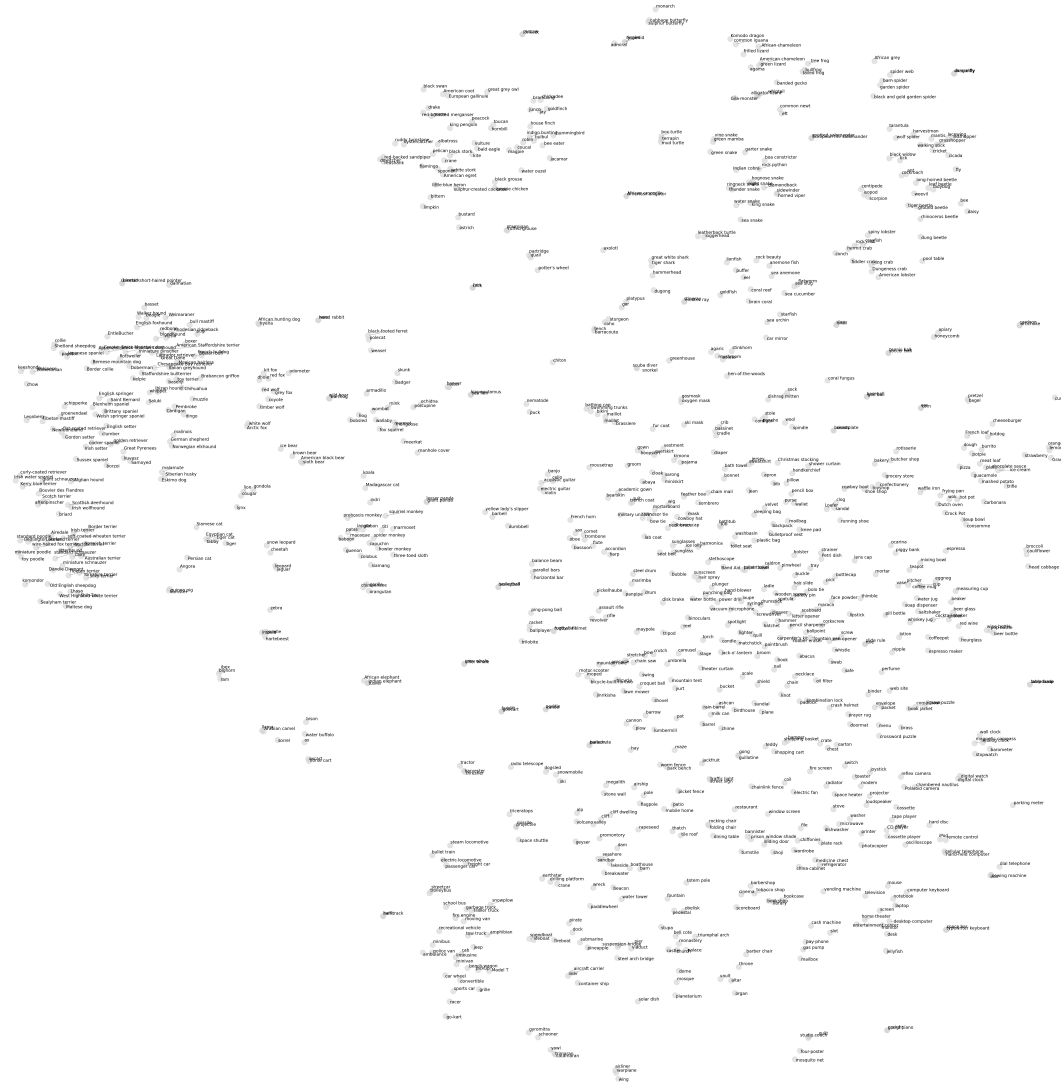


Figure 22 t-SNE visualization of ImageNet class embeddings in MoCo-v2's (Chen et al., 2020c) feature space after 800 epochs of pretraining. Each point represents the average feature vector of validation set images for one class.

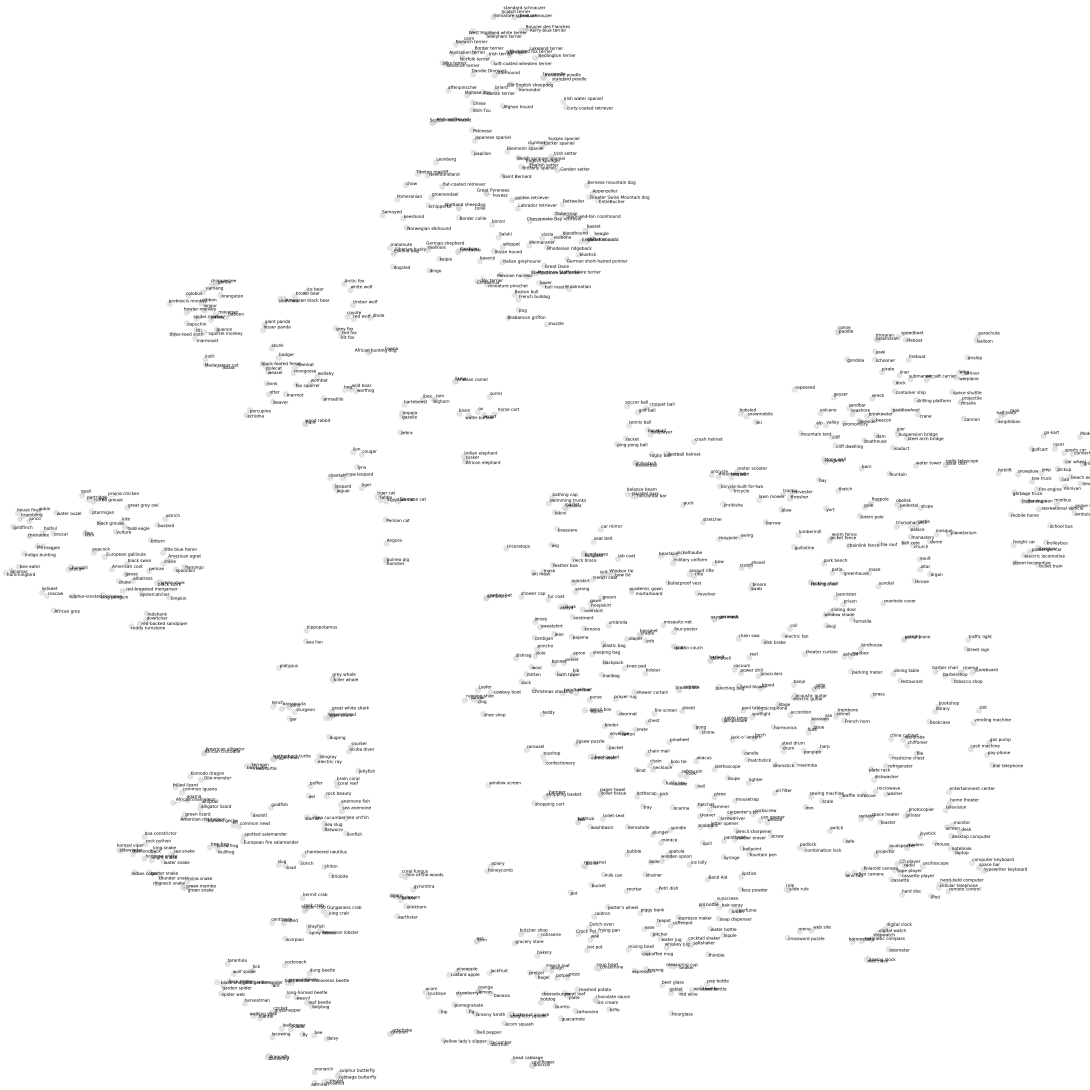


Figure 23 t-SNE visualization of ImageNet class embeddings in the supervised feature space (Paszke et al., 2019). Each point corresponds to the mean feature vector of validation images belonging to a single class.