

SSL using Resnet101 and MOCO for Image Classification

Samarth Agarwal
sa9016@nyu.edu

Evan Beck
ecb9981@nyu.edu

Yash Jadhav
yj3076@nyu.edu

Abstract

We study self-supervised representation learning for image classification using a ResNet-101 encoder with a linear probing evaluation protocol. We implement and compare three widely used self-supervised learning frameworks: SimCLR [1], MoCo [3], and BYOL [2] and examine their performance under a consistent training and evaluation setup. Through extensive experimentation, we explore the impact of key hyperparameters, including learning rate, MLP projection head dimensionality, weight decay, and batch size. Across methods, we observe that representation quality improves rapidly during early training and saturates after approximately 50 epochs, with marginal gains beyond this point. Our results highlight the trade-offs between different self-supervised objectives and provide empirical insights into convergence behavior and hyperparameter sensitivity when scaling self-supervised learning to deeper encoders.

1. Introduction

Image classification is a fundamental and extensively studied problem in computer vision. Modern deep learning approaches have achieved strong performance on this task, but they typically rely on large-scale annotated datasets, which are expensive and time-consuming to curate. In many real-world scenarios, labeled data may be scarce, incomplete, or entirely unavailable. Self-supervised learning (SSL) addresses this limitation by learning meaningful visual representations from unlabeled data, significantly reducing annotation costs while enabling strong downstream performance.

Recent advances in SSL, such as DINO and JEPa, have demonstrated impressive results on image classification and representation learning. However, these methods often rely on very large backbone architectures and massive datasets, resulting in substantial computational and memory requirements. Training such models typically demands long training schedules and extensive hardware resources, which may be impractical in constrained research settings.

In this work, we study contrastive and bootstrap-based self-supervised learning methods under realistic computational constraints. Given a development timeline of approximately one month and access to limited GPU resources (L4 and 1–2 A100 GPUs), we focus on the ResNet-101 backbone as a strong yet computationally tractable architecture. ResNet-101 provides sufficient representational capacity to capture semantic structure while allowing for extensive experimentation with training objectives, data augmentation strategies, and evaluation protocols.

We investigate and compare several widely used SSL methods, including SimCLR, Momentum Contrast (MoCo), and Bootstrap Your Own Latent (BYOL). SimCLR and MoCo rely on contrastive objectives that leverage strong data augmentations and either large batch sizes or momentum-based negative queues to learn discriminative representations. BYOL, in contrast, removes explicit negative samples and instead employs an asymmetric architecture with a momentum-updated target network to avoid representation collapse. By studying these methods within a unified training and evaluation framework, we aim to understand their convergence behavior, robustness, and efficiency under fixed compute budgets.

Our experiments evaluate representation quality using both linear probing and k-nearest neighbors (k-NN) classification across multiple datasets. We analyze performance as a function of training duration and observe that representation quality improves rapidly during early training and saturates after approximately 50 epochs. These findings suggest that effective self-supervised representations can be learned efficiently without prolonged training schedules or excessively large models.

Overall, this work provides a systematic empirical study of self-supervised image classification using ResNet-101 under practical computational constraints. Our results highlight the trade-offs between contrastive and bootstrap-based methods and demonstrate that strong representations can be learned efficiently through careful architectural and augmentation choices.

2. Methodology

2.1. Overall Framework

We follow a two-stage self-supervised learning pipeline. First, an encoder is pretrained on unlabeled images using a self-supervised objective. Second, the encoder is frozen and evaluated on downstream image classification using linear probing or nearest-neighbor classification. All methods share a common backbone architecture, data augmentation pipeline, and evaluation protocol to ensure fair comparison.

2.2. Problem Setup and Notation

Let x denote an input image sampled from an unlabeled dataset. Two stochastic augmentations $t, t' \sim \mathcal{T}$ are applied to generate two correlated views $v = t(x)$ and $v' = t'(x)$. An encoder f_θ maps each view to a representation $h = f_\theta(v)$. A projection head g_θ maps h to an embedding $z = g_\theta(h)$ used for the self-supervised objective. During evaluation, the encoder f_θ is frozen and representations h are used for downstream classification.

2.3. Data Augmentation

Data augmentation plays a central role in self-supervised learning by defining the invariances that the learned representations should capture. For all methods, we employ a SimCLR-style augmentation pipeline to generate multiple correlated views from each input image. All images are resized to a fixed resolution of 96×96 pixels before being processed by the network.

Global Views. Each input image is transformed into two *global views* using random resized cropping, followed by random horizontal flipping, color jittering, random grayscale conversion, and Gaussian blur. For global crops, the random resized crop samples a large portion of the image, with the crop area covering a substantial fraction of the original image, and resizes it to 96×96 . This encourages invariance to scale and viewpoint. Color jittering perturbs brightness, contrast, saturation, and hue, while random grayscale conversion reduces reliance on color cues. Gaussian blur is applied with a randomly sampled standard deviation to suppress high-frequency texture information. All augmentations are applied independently to generate two distinct global views.

Local Crop Augmentation. In addition to global views, we optionally introduce a *local crop* augmentation. Local crops are generated by sampling a smaller spatial region of the image using random resized cropping, followed by resizing to 96×96 . Compared to global crops, local crops cover a significantly smaller area of the image and emphasize fine-grained local structures. This multi-scale

augmentation encourages the encoder to learn representations that are consistent across spatial scales and prevents over-reliance on global context. Local cropping is enabled conditionally and implemented using GPU-accelerated Kornia operators to avoid CPU bottlenecks during training.

Implementation Details. All augmentations are implemented using torchvision transforms by default, with an optional Kornia-based implementation for GPU acceleration. The same augmentation strategy and crop configurations are shared across SimCLR, MoCo, and BYOL to ensure a fair comparison between methods.

2.4. Architecture

The encoder backbone is based on a ResNet-101 architecture. The final fully connected classification layer is removed and replaced with an identity mapping, allowing the network to output a high-dimensional feature representation. On top of the encoder, we attach a projection head implemented as a multi-layer perceptron (MLP).

The projection head consists of two fully connected layers. The first layer maps the encoder output to a hidden projection dimension, followed by a ReLU nonlinearity. The second layer maps this hidden representation to the final embedding space used for the self-supervised objective. The dimensionality of the projection head output is treated as a tunable hyperparameter and is shared across all methods.

For MoCo and BYOL, a momentum-updated copy of the encoder and projection head is additionally maintained. The momentum encoder is updated using an exponential moving average of the online encoder parameters and is not updated via backpropagation.

2.5. SimCLR

SimCLR learns representations by contrasting positive pairs (two augmented views of the same image) against negative samples drawn from the same minibatch. Given a batch of N images and two views per image, SimCLR optimizes the NT-Xent loss:

$$\mathcal{L}_{\text{SimCLR}} = - \sum_{i=1}^{2N} \log \frac{\exp(\text{sim}(z_i, z_{j(i)})/\tau)}{\sum_{k \neq i} \exp(\text{sim}(z_i, z_k)/\tau)}, \quad (1)$$

where $\text{sim}(\cdot, \cdot)$ denotes cosine similarity, τ is a temperature parameter, and $j(i)$ indexes the positive pair for i . All remaining samples in the batch act as implicit negatives. As a result, SimCLR performance is sensitive to batch size, since the number of negative samples scales with the minibatch. For the SimCLR implementation, and our base code starts with [4]

2.6. Momentum Contrast (MoCo)

MoCo addresses the batch size limitation of SimCLR by maintaining a large, persistent dictionary of negative samples. The method uses two encoders: a query encoder f_θ updated by backpropagation, and a key encoder f_ξ updated via exponential moving average (EMA):

$$\xi \leftarrow m\xi + (1 - m)\theta, \quad (2)$$

where m is a momentum coefficient. Negative samples are stored in a fixed-size queue, which is updated by enqueueing new keys and dequeuing the oldest entries.

For a query embedding q and its corresponding positive key k^+ , MoCo optimizes an InfoNCE loss against the queue of negative keys $\{k^-\}$:

$$\mathcal{L}_{\text{MoCo}} = -\log \frac{\exp(\text{sim}(q, k^+)/\tau)}{\exp(\text{sim}(q, k^+)/\tau) + \sum_{k^-} \exp(\text{sim}(q, k^-)/\tau)} \quad (3)$$

In our implementation, the queue size is fixed and the momentum coefficient is constant throughout training. Batch normalization shuffling is applied in the single-GPU setting to prevent information leakage between the query and key encoders.

2.7. Bootstrap Your Own Latent (BYOL)

BYOL removes explicit negative samples entirely. It consists of an online network and a target network, each composed of an encoder and projection head. An additional prediction head is applied to the online network. The target network parameters are updated using an exponential moving average of the online network parameters.

Given two augmented views, the online network predicts the representation produced by the target network from the alternate view. The BYOL loss is defined as:

$$\mathcal{L}_{\text{BYOL}} = \|\hat{p}_\theta(g_\theta(f_\theta(v))) - \text{sg}(\hat{g}_\xi(f_\xi(v')))\|_2^2, \quad (4)$$

where $\text{sg}(\cdot)$ denotes the stop-gradient operator and $\hat{\cdot}$ indicates ℓ_2 normalization. Collapse is avoided through the asymmetric architecture and EMA updates. BYOL is trained using a separate training configuration with stochastic gradient descent.

2.8. Training Details

For SimCLR and MoCo, we use the Adam optimizer with cosine annealing learning rate scheduling. BYOL is trained using stochastic gradient descent with momentum. Mixed-precision training is enabled to improve computational efficiency. We explore the effects of learning rate, batch size, projection head dimensionality, and weight decay. Across all methods, we observe that representation quality improves rapidly during early training and saturates after approximately 50 epochs.

2.9. Evaluation Protocol

To evaluate representation quality, we freeze the pretrained encoder and train a linear classifier on top of the extracted representations. We additionally report results using a k-nearest neighbors (k-NN) classifier for non-parametric evaluation. These protocols measure the linear separability of the learned features without fine-tuning the backbone.

3. Experimental Setup and Progress

3.1. Experimental Progression

Our experimental design evolved iteratively through multiple stages, guided by empirical performance on downstream evaluation tasks. We began with ResNet-50 as the backbone architecture and focused on evaluating learned representations using k-nearest neighbors (k-NN) classification on the CUB-200 dataset, which provides a fine-grained image classification benchmark well-suited for representation analysis.

Initial experiments employed SimCLR and BYOL with ResNet-50. During this stage, we primarily evaluated performance using k-NN classification. To improve representation diversity, we augmented the training data by incorporating approximately 200k images from the iNaturalist dataset. This addition resulted in modest improvements in k-NN accuracy, suggesting that increased visual diversity positively impacts learned representations.

To further enhance performance, we introduced a local crop augmentation in addition to the standard global crops. This modification encouraged multi-scale consistency and led to a noticeable improvement in k-NN accuracy, increasing performance on CUB-200 to approximately 15%. These results motivated further exploration of architectural and objective-level changes.

3.2. Transition to Momentum-Based Methods

In later stages of experimentation, we transitioned from SimCLR and BYOL to Momentum Contrast (MoCo), motivated by improved training stability observed in preliminary experiments. We first trained MoCo using a ResNet-50 backbone and observed more stable optimization dynamics compared to SimCLR and BYOL. Based on these findings, we progressively scaled the backbone architecture, first to Wide ResNet-50 ($2 \times$ width) and subsequently to ResNet-101.

Among the evaluated architectures, ResNet-101 achieved the best performance on the CUB-200 dataset. We then evaluated the learned representations on additional datasets, including Mini-ImageNet and the SUN scene recognition dataset, to assess generalization across domains.

To further improve performance on these datasets, we expanded the pretraining corpus by incorporating approx-

imately 600k images from the CC12M dataset. The additional large-scale, diverse data improved generalization, particularly on Mini-ImageNet and SUN, highlighting the importance of data scale for representation learning.

3.3. Training Dynamics and Loss Behavior

We observed distinct optimization behaviors across self-supervised methods. For SimCLR, the contrastive loss decreased rapidly early in training and quickly saturated around a value of approximately 0.12, after which it exhibited oscillatory behavior without corresponding improvements in downstream accuracy. BYOL exhibited less stable training dynamics, with the loss failing to clearly stabilize across training runs.

In contrast, MoCo demonstrated more consistent optimization behavior. The MoCo loss decreased rapidly to approximately 4.0 and continued a gradual decline toward 3.0 over training. While the loss continued to decrease, downstream accuracy improvements eventually saturated, indicating diminishing returns from extended training. These observations suggest that while momentum-based contrastive objectives offer improved stability, loss convergence alone does not guarantee continued gains in representation quality.

3.4. Summary of Experimental Findings

Overall, our experiments highlight the importance of architectural scaling, dataset diversity, and augmentation strategies in self-supervised learning. Momentum-based methods exhibited more stable training dynamics, while multi-scale augmentations and increased data diversity consistently improved k-NN and linear probe performance. Our final configuration, based on MoCo with a ResNet-101 backbone and large-scale pretraining data, achieved the strongest performance across evaluated datasets.

4. Results

We evaluate the quality of learned representations using both linear probing and k-nearest neighbors (k-NN) classification across three datasets. Unless otherwise stated, results are reported as top-1 accuracy. We analyze performance trends as a function of pretraining epochs to study convergence behavior and stability across methods.

4.1. k-Nearest Neighbor Evaluation

We additionally report k-NN validation accuracy to assess representation quality without training a parametric classifier. We used $k = 20$ for evaluation every 5 epochs. As shown in Fig. 1, k-NN accuracy follows a similar trend to linear probing, with rapid early improvement and stabilization after approximately 50 epochs.

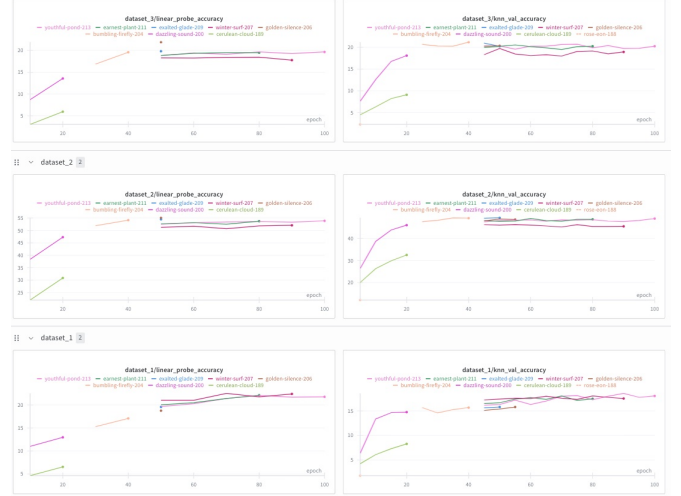


Figure 1. Validation accuracy across datasets for different runs with resume-from-checkpoints.

4.2. Linear Probing Performance

Linear probe accuracy improves rapidly during early stages of training across all datasets. As shown in Fig. 1, representation quality increases sharply within the first 20–40 epochs, after which performance gains diminish. We would evaluate the linear probe every 10 epochs and used a $1e-3$ learning rate and trained for 10 epochs. Across datasets, accuracy stabilizes around 50 epochs, with only marginal improvements observed up to 100 epochs.

4.3. Final Training and Results

For the leaderboard evaluation, we pretrained a ResNet-101 encoder for 100 epochs using our extended pretraining dataset as discussed in Section 3.2. The pretrained encoder was frozen and evaluated using linear probing, where a linear classifier was trained for 30 epochs. Hyperparameters for the linear probe were selected via grid search, with the best performance achieved using a learning rate of 1×10^{-2} and no weight decay. Compared to default probe settings, hyperparameter tuning and extended probe training yielded a consistent improvement of approximately 5–15% across all three datasets.

Table 1. Leaderboard results (ResNet-101, 100-epoch SSL pre-training; 30-epoch linear probe, best LR = 10^{-2} , WD = 0).

Part	Public	Private
Part 1 (Dataset 1)	34.46%	31.26%
Part 2 (Dataset 2)	59.93%	62.05%
Part 3 (Dataset 3)	33.58%	33.24%

References

- [1] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey E. Hinton. A simple framework for contrastive learning of visual representations. *CoRR*, abs/2002.05709, 2020. [1](#)
- [2] Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre H. Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Ávila 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. *CoRR*, abs/2006.07733, 2020. [1](#)
- [3] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross B. Girshick. Momentum contrast for unsupervised visual representation learning. *CoRR*, abs/1911.05722, 2019. [1](#)
- [4] Thalles Santos Silva. Exploring simclr: A simple framework for contrastive learning of visual representations. <https://sthalles.github.io>, 2020. [2](#)