Adaptive Similarity Bootstrapping for Self-Distillation based Representation Learning

Tim Lebailly[1*] Thomas Stegmüller[2*] Behzad Bozorgtabar[2,3]
Jean-Philippe Thiran[2,3] Tinne Tuytelaars[1]
[1]KU Leuven [2]EPFL [3]CHUV
[1]{firstname}.{lastname}@esat.kuleuven.be [2]{firstname}.{lastname}@epfl.ch

Abstract

Most self-supervised methods for representation learning leverage a cross-view consistency objective i.e. *they maximize the representation similarity of a given image's augmented views. Recent work NNCLR goes beyond the cross-view paradigm and uses positive pairs from different images obtained via nearest neighbor bootstrapping in a contrastive setting. We empirically show that as opposed to the contrastive learning setting which relies on negative samples, incorporating nearest neighbor bootstrapping in a self-distillation scheme can lead to a performance drop or even collapse. We scrutinize the reason for this unexpected behavior and provide a solution. We propose to adaptively bootstrap neighbors based on the estimated quality of the latent space. We report consistent improvements compared to the naive bootstrapping approach and the original baselines. Our approach leads to performance improvements for various self-distillation method/backbone combinations and standard downstream tasks. Our code is publicly available at https://github.com/tileb1/AdaSim.*

1. Introduction

Self-supervised learning (SSL) methods have seen a lot of breakthroughs over the past few years. Most recent self-supervised methods train features invariant to data augmentation by maximizing the similarity between two augmentations of a single input image. However, this task is ill-posed as this optimization procedure admits trivial solutions (resulting in a "collapsed" scenario). Similarity maximization (or cross-view consistency) SSL methods can be categorized based on how they avoid trivial solutions. The most famous subset are contrastive learning methods [10, 9, 25, 11, 13] in which the collapse is avoided by using negative pairs. On the one hand, the learning procedure is robust since collapse avoidance is explicitly modeled in the training objective, but on the other hand, it requires large batches to have a sufficient pool of negative samples. This makes them GPU memory inefficient and limits research to those who dispose of large distributed computing infrastructure.

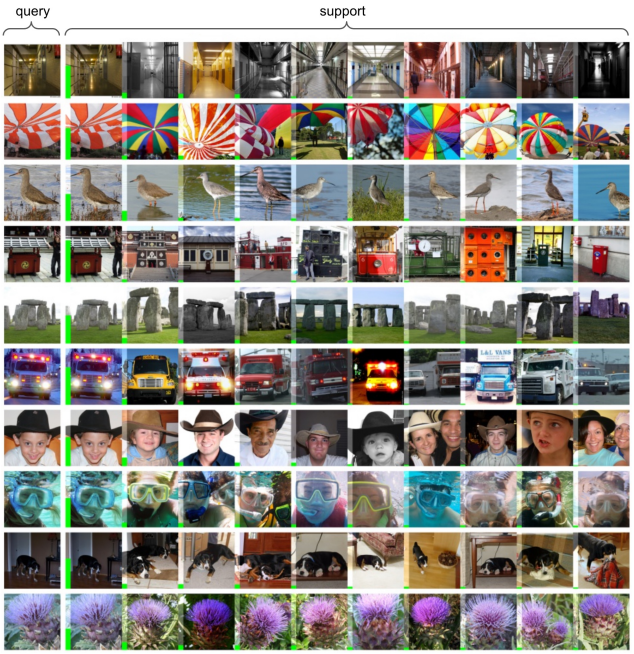


Figure 1: **The selection of positive image pairs used for cross-view consistency in self-supervised representation learning is key for good performance.** With our method, given the query (or anchor) image on the left, similar images are successfully ranked according to $p^{\text{win}}(\boldsymbol{x}_j|\boldsymbol{x}_i)$ (illustrated as a green bar on the bottom left of each image). Our algorithm enforces similarity between the query $\boldsymbol{x}_i$ and an image $\boldsymbol{x}_j$ sampled from $p^{\text{win}}(\boldsymbol{x}_j|\boldsymbol{x}_i)$. These results are non-cherry-picked and obtained at the final epoch (800) of the pretraining. Best viewed in color and zoomed-in.

* denotes equal contribution.

More recently, self-distillation methods have been gaining traction [12, 8, 24, 33]. These similarity maximization algorithms avoid trivial solutions by using asymmetry. This asymmetry can take the form of an additional predictor [12, 24] on one branch, using stop-gradients [12, 8, 33, 24], a momentum encoder [8, 24], *etc*. These methods are of particular interest as 1) they do not require large batch sizes, and 2) they currently show state-of-the-art performance [8, 33] on standard downstream tasks.

Orthogonal to the choice of the framework (contrastive vs self-distillation), one can wonder what is the best way to obtain positive pairs. Intuitively, similarity maximization SSL methods could be improved by using positive pairs from different images. Indeed if an oracle indicating valid positive pairs [28, 29] was available, instead of taking two augmentations from the same image, we could simply take pairs from the oracle. The features would, therefore, not be trained to be invariant to handcrafted data augmentations but invariant to intra-class variation, which would make them more aligned with most common downstream tasks, *e.g*. classification.

In the absence of labels, we can leverage the structure of the latent space to obtain a proxy for the oracle. Semantically related images are expected to lie in the vicinity of one another in the latent space. However, this is a chicken and egg problem, as this assumption only holds when the quality of the learned latent space is good enough. If the learned latent space is not of good quality, bootstrapping the proxy leads to unwanted gradient flows, e.g., an image of a cat is pulled closer to the image of a building.

Nevertheless, recent work NNCLR [19] has successfully incorporated nearest neighbor (NN) bootstrapping in a contrastive setting. Considering that self-distillation typically outperforms contrastive methods, in this work, we explore how the same can be achieved without explicit use of negatives.

Unfortunately, this combination does not work out of the box. **We empirically observe that it can be hurtful and even lead to collapse.** We scrutinize the reason for this unexpected behavior and provide a solution. We propose to estimate the quality of the latent space and adaptively use positive pairs sampled from a ranked set of neighbors (Fig. 1) if the estimated quality of the latent space is high enough. This leads to an **Ada**ptive learning algorithm based on **Sim**ilarity bootstrapping dubbed AdaSim. The overall framework is shown in Figure 2. We summarize our contributions as follows:

1. We provide empirical evidence that when combined with self-distillation, straightforward bootstrapping as in [19] can lead to a performance drop or even collapse. This is validated for multiple self-distillation methods and backbone combinations;
2. We propose an adaptive similarity bootstrapping learning method (AdaSim) in which the amount of bootstrapping is modulated via a single temperature parameter. Using a temperature parameter of 0, AdaSim defaults to self-distillation with standard positive image pairs generated from augmented views of the same image. We show that AdaSim performs best with a non-zero temperature parameter and outperforms the baselines on standard downstream tasks.

2. Related work

Cross-view consistency Early self-supervised methods make use of pretext tasks such as solving jigsaw puzzles [38], image rotation prediction [23] and more [17, 36, 41, 2]. Recently, there has been a shift towards learning features that are invariant to semantic preserving data augmentations [8, 12, 25, 10, 11, 13, 50]. These data augmentations include geometric transforms (*e.g*. `CROP`, `RESIZE` and `HORIZONTAL_FLIP`) and photometric transforms (*e.g*. `COLOR_JITTER`, `SOLARIZE`, `GAUSSIAN_BLUR` and `GRAYSCALE`). Stronger semantic preserving data augmentations lead to better downstream performance. However, the above-mentioned transforms lose their semantic preserving nature when they are too strong, *e.g*. a very small `CROP` does not capture the object or a strong `GAUSSIAN_BLUR` leads to a uniform image.

Dense Cross-view consistency Instead of applying coherence at the global-level, a more granular self-supervision can be obtained by enforcing cross-view consistency between matching local regions [27, 32, 39, 44, 47, 46].

Neighbor bootstrapping In order to generate strong semantic positive pairs less reliant on heuristics, NNCLR [19] proposes to use positive pairs of different images by bootstrapping nearest neighbors in the latent space. We describe their method in detail in Section 3.2 as well as the issues that arise when used in conjunction with a self-distillation objective, which we try to overcome using adaptivity in Section 3.3. Similarly, [30] proposes to bootstrap multiple neighbors for a single query.

Clustering methods Clustering methods [1, 6, 4, 5, 51] also process multiple different images but do not make use of positive/negative pairs. They enforce structure in the latent space by learning prototypes and enforcing clusters to be compact.

Queues/memory banks Memory banks have mostly been used in the context of contrastive learning for storing negatives [25, 11, 13] reducing the need for (very) large batch sizes. [19] uses memory banks for mining positives while [16] makes use of memory banks for mining both positives and negatives.

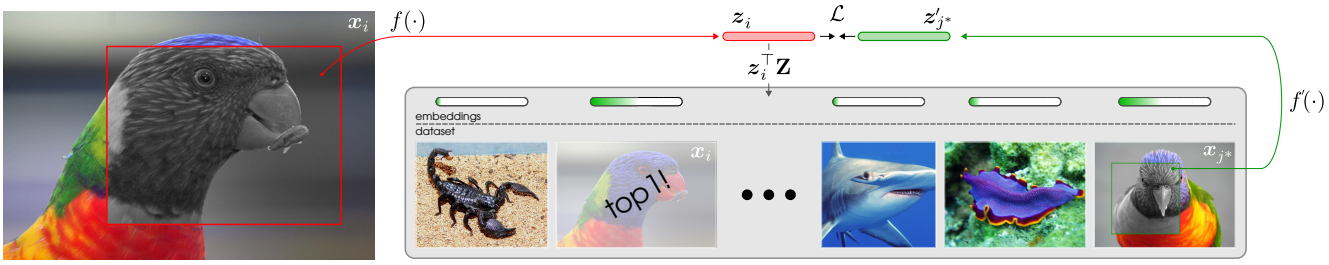


Figure 2: **Overview of AdaSim.** Given an input image $\boldsymbol{x}_i$, we obtain the latent representation $\boldsymbol{z}_i = f(t(\boldsymbol{x}_i))$. Additionally, we sample another image $\boldsymbol{x}_{j^\star}$ in the dataset from $p^{win}(\boldsymbol{x}_j|\boldsymbol{x}_i)$ (see Eq. (9) and Eq. (10)) and obtain its latent representation $\boldsymbol{z}'_{j^\star} = f'(t'(\boldsymbol{x}_{j^\star}))$. A self-distillation loss $\mathcal{L}$ is enforced between $\boldsymbol{z}_i$ and $\boldsymbol{z}'_{j^\star}$. For the sake of simplicity, only the scenario using bootstrapping is illustrated (see Algorithm 1). Data augmentations are represented with grayscale bounding boxes.

3. Method

3.1. Self-distillation vs contrastive learning

Self-distillation and contrastive learning are ubiquitous within self-supervised learning. Both schemes aim to learn discriminative features in the absence of labels. This is mainly done by enforcing similarity constraints between two augmentations of the same input image. The two methods are similar in essence but differ in the way they avoid trivial solutions. Assume we dispose of an encoder f from which we obtain a latent representation $\boldsymbol{z} \in \mathcal{Z}$ of an image $\boldsymbol{x} \in \mathcal{X}$, *i.e.* $\boldsymbol{z} = f(\boldsymbol{x})$ with $\mathcal{Z}$ and $\mathcal{X}$ being a latent- and image space, respectively. Moreover, assume we dispose of an oracle $\mathcal{N}^+$ indicating valid positive pairs of images $(\boldsymbol{x}, \boldsymbol{x}^+) \in \mathcal{N}^+$, an oracle $\mathcal{N}^-$ indicating valid negative pairs of images $(\boldsymbol{x}, \boldsymbol{x}^-) \in \mathcal{N}^-$ and a distance metric[1] $d(\cdot\ ,\ \cdot)$ defined in the latent space $\mathcal{Z}$. Valid positive pairs are images with the same semantic content and valid negative pairs are images with no shared semantic content.

Contrastive objective A contrastive learning loss relies on attraction and repelling mechanisms: the former enforces similarity between positive pairs and the latter enforces dissimilarity between the negative pairs. Formally, the attraction term is of the form $d(f(\boldsymbol{x}), f(\boldsymbol{x}^+))$ and the repelling terms are of the form $d(f(\boldsymbol{x}), f(\boldsymbol{x}^-))$. Here, we refer to "term" in its broad sense and therefore do not necessarily refer to an additive term. Usually, there are many negative terms for a single positive term. One famous example of such contrastive loss is the InfoNCE loss [43, 45, 48] defined as:

$$\mathcal{L}_{\text{contra}} = -\log\left(\frac{\exp(s^+/\tau)}{\exp(s^+/\tau) + \sum_{s^-}\exp(s^-/\tau)}\right) \tag{1}$$

where $s^+ = f(\boldsymbol{x})^\top f(\boldsymbol{x}^+)$ and $s^- = f(\boldsymbol{x})^\top f(\boldsymbol{x}^-)$. $(\boldsymbol{x}, \boldsymbol{x}^+)$ is sampled from $\mathcal{N}^+$ and $(\boldsymbol{x}, \boldsymbol{x}^-)$ are sampled from $\mathcal{N}^-$. The distance metric $d(\cdot\ ,\ \cdot)$ is defined as the scalar product $\langle \cdot\ ,\ \cdot \rangle$. The total contrastive objective is Equation (1) summed over all training images $\boldsymbol{x}$.

Self-distillation objective As opposed to the contrastive scenario, the self-distillation objective does not use negative image pairs to avoid the collapse to trivial solutions but uses asymmetry between the two branches. The form the asymmetry takes (momentum encoder, additional predictor on one branch, using stop-gradients in one branch *etc.* [8, 12, 24]) can be abstracted out. Given two encoders f and f', a self-distillation loss only has positive terms of the form

$$\mathcal{L}_{distil} = d(f(\boldsymbol{x}), f'(\boldsymbol{x}^+)) \tag{2}$$

for a given positive pair $(\boldsymbol{x}, \boldsymbol{x}^+)$. The total self-distillation objective is Equation (2) summed over all positive pairs $(\boldsymbol{x}, \boldsymbol{x}^+) \in \mathcal{N}^+$.

3.2. Bootstrapping neighbors in the latent space

In the absence of oracle $\mathcal{N}^+$ and $\mathcal{N}^-$, most (if not all) previous work approximate $\mathcal{N}^-$ with random image pairs. Given a distribution $\mathcal{T}$ of semantic preserving data augmentations, $\mathcal{N}^+$ is usually approximated with pairs of random augmentations from the same input image, *i.e.* $(t(\boldsymbol{x}), t'(\boldsymbol{x}))$ where t and t' are sampled from $\mathcal{T}$. The stronger the semantic preserving augmentations t and t' are, the better the learned features become. However, their semantic preserving nature will be lost if they are made too strong.

To obtain more complex and diverse pairs of positive images, NNCLR [19] proposes to approximate $\mathcal{N}^+$ with pairs of nearest neighbors. More precisely, given two latent representations ($\boldsymbol{z}$ and $\boldsymbol{z}'$) of the same image $\boldsymbol{x}$ and a FIFO queue Q of previously computed representations (with $|Q| < |\mathcal{D}|$), positive pairs are defined as $(\boldsymbol{z}', \text{NN}(\boldsymbol{z}, Q))$, where the nearest neighbor operator is defined as:

$$\text{NN}(\boldsymbol{z}, Q) = \underset{q \in Q}{\arg\min} \|z - q\|_2 \tag{3}$$

[1] This is an abuse of terminology as d does not necessarily have to satisfy all properties of a mathematical distance.

Note here that the positive pairs are defined in the latent space $\mathcal{Z}$ and not in the image space. Under the assumption that the latent space properly captures the semantics of images, these pairs of neighbors are expected to share the same semantic content but their representation may still be slightly different. Enforcing similarity constraints between the two representations would help to learn features that are invariant to everything but the semantics of the image (*e.g.* class label information). However, two issues arise when relying exclusively on nearest neighbors as positive pairs:

1. Using only neighbors as positive pairs, *i.e.* not relying on augmented views as positive pairs, leaves out valuable self-supervisory signal. Using standard positive pairs of augmented views from the same image is desirable to explicitly learn data-augmentation invariant features, but that is not enforced.

2. The latent space might not capture the semantics of the image well, *i.e.* the positive pair is wrong. This would lead to undesirable gradient flows, *e.g.* pulling an image of a cat closer to an image of a building.

Throughout the paper, we refer to the above as issue 1 and issue 2. Using a contrastive objective, the impact of these issues is limited since informative gradient signal can still be obtained from the negative pairs which in practice are almost always correct (random). Using a self-distillation objective, we empirically observe that the above issues are problematic to the point that the downstream performance can be worse than using standard positive pairs using data-augmentations (Sec. 4, Tab. 2, Tab. 3).

3.3. Adaptive similarity bootstrapping

3.3.1 Need for standard positive pairs

To avoid issue 1, we adaptively use augmentations of the same image or of a neighbor to form a positive pair. To do this, we propose to work with a cache that has the same size as the dataset $\mathcal{D}$ as opposed to using the queue Q from NNCLR [19]. Using a small queue, it is very unlikely to encounter a representation originating from the same image. We denote the cache with $\mathbf{Z} \in \mathbb{R}^{N \times d}$, where N is the size of the dataset and d is the dimension of the latent space. At the end of the forward pass, the current latent representation $\boldsymbol{z}_i$ of an augmentation of the i-th image $\boldsymbol{x}_i \in \mathcal{D}$ (*i.e.* $f(t(\boldsymbol{x}_i))$ with $t \sim \mathcal{T}$) is updated in the cache. As such, $\mathbf{Z}$ holds a latent representation for every image in the dataset at all times. Given a latent representation $\boldsymbol{z}_i$ of the i-th image and the cache $\mathbf{Z}$, we can define a similarity metric $m_i(j)$ between image i and all images $\boldsymbol{x}_j$:

$$m_i(j) = \boldsymbol{z}_i^\top \mathbf{Z}_j \tag{4}$$

where $\mathbf{Z}_j$ refers to the latent representation of image j in the cache. $m_i(j)$ can in turn be mapped into a similarity distribution using a softmax normalization:

$$s_i(j) = \frac{\exp\left(m_i(j)/\tau\right)}{\sum_{k \in [|\mathcal{D}|]} \exp\left(m_i(k)/\tau\right)}, \qquad i, j \in [|\mathcal{D}|] \tag{5}$$

where τ is a temperature parameter modulating the sharpness of the distribution (ablation in Tab. 4). We can now define an isomorphic probability distribution over the images in the dataset:

$$p(\boldsymbol{x}_j|\boldsymbol{x}_i) = s_i(j), \qquad \boldsymbol{x}_i, \boldsymbol{x}_j \in \mathcal{D} \tag{6}$$

To approximate the oracle $\mathcal{N}^+$ of positive pairs, we propose to use image i and an image sampled from the similarity distribution. That is, we form positive pairs of the form $(t(\boldsymbol{x}_i), t'(\boldsymbol{x}_{j^\star}))$ with $\boldsymbol{x}_{j^\star}$ sampled from $p(\boldsymbol{x}_j|\boldsymbol{x}_i)$ and with t and t' sampled from $\mathcal{T}$. Note that $\mathbf{Z}$ contains features for all images, not excluding image $\boldsymbol{x}_i$. Therefore, we always have a non-zero probability of having a positive pair generated from the same input image which mitigates issue 1 from Section 3.2. Sampling positive pairs of the form $(t(\boldsymbol{x}_i), t'(\boldsymbol{x}_{j^\star}))$ also allows for the possibility to sample more diverse and complex pairs of positives compared to the case when we only consider top-1 neighbors, as can be seen in Figure 1. This diversity can be increased by increasing the temperature τ.

3.3.2 Need for adaptivity

Recall that issue 2 from Section 3.2 is that the latent space might not capture the semantics of images properly (especially at the beginning of the pretraining). That is, neighbors in the latent space might have completely unrelated semantic content. We propose to estimate the quality of the latent space by observing how close two different augmentations of the same input image $\boldsymbol{x}_i$ are mapped via the encoder f. If this distance is low compared to that of the latent representations of other images in the cache $\mathbf{Z}$, then it means that the encoder f is good at mapping images similar to $\boldsymbol{x}_i$ close together. In that case, we can expect the vicinity of the queried image $\boldsymbol{x}_i$ to also share semantic content with image $\boldsymbol{x}_i$ and can therefore use elements of the vicinity to form a positive pair with $\boldsymbol{x}_i$. If this distance is too high, we default to a standard positive pair composed of two augmentations of the same input image. Mathematically, if $\arg\max_{\boldsymbol{x}_j} p(\boldsymbol{x}_j|\boldsymbol{x}_i) == \boldsymbol{x}_i$, then we sample $\boldsymbol{x}_{j^\star}$ from $p(\boldsymbol{x}_j|\boldsymbol{x}_i)$ and use a positive pair $(t(\boldsymbol{x}_i), t'(\boldsymbol{x}_{j^\star}))$ with t and t' sampled from $\mathcal{T}$. Otherwise, we use a standard positive pair $(t(\boldsymbol{x}_i), t'(\boldsymbol{x}_i))$.

3.3.3 How to rank neighbors?

We propose to extend the adaptive framework to account for the similarity history over the past epochs. The rationale behind this is that the similarity between two images

$t(\boldsymbol{x}_i)$ and $t'(\boldsymbol{x}_j)$ can be strongly affected by t and t', especially at the beginning of the pretraining. For example, given a randomly initialized encoder f, the similarity between $f(t(\boldsymbol{x}_i))$ and $f(t'(\boldsymbol{x}_j))$ will be mostly determined by how similar t and t' are. Therefore, an image $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ should be considered as semantically close, not only if $f(t(\boldsymbol{x}_i))$ is close to $f(t'(\boldsymbol{x}_j))$, but if $\mathbb{E}_{t\sim\mathcal{T}}[f(t(\boldsymbol{x}_i))]$ is close to $\mathbb{E}_{t'\sim\mathcal{T}}[f(t'(\boldsymbol{x}_j))]$.

In practice, we do not have access to the true expectation and therefore take the empirical mean over the last w epochs. More precisely, we define the similarity metric for a given epoch e which we denote with the superscript $^{(e)}$:

$$m_i^{(e)}(j) = \left(\boldsymbol{z}_i^\top \mathbf{Z}_j\right)^{(e)} \tag{7}$$

and average this similarity metric over the last w epochs to obtain a windowed similarity metric for the current epoch E:

$$m_i^{\text{win}}(j) = \frac{1}{w}\sum_{e\in\mathcal{W}_E^w} m_i^{(e)}(j) \tag{8}$$

where $\mathcal{W}_E^w = \{E-w+1, E-w+2, \cdots E\}$ denotes the set of the previous w epochs with epoch E being the current epoch. Similarly to Equation (5), we can define:

$$s_i^{\text{win}}(j) = \frac{\exp\left(m_i^{\text{win}}(j)/\tau\right)}{\sum_{k\in[|\mathcal{D}|]}\exp\left(m_i^{\text{win}}(k)/\tau\right)}, \qquad i,j\in[|\mathcal{D}|] \tag{9}$$

where τ is a temperature parameter as in Equation (5). And similarly to Equation (6), we can define:

$$p^{\text{win}}(\boldsymbol{x}_j|\boldsymbol{x}_i) = s_i^{\text{win}}(j), \qquad \boldsymbol{x}_i,\boldsymbol{x}_j\in\mathcal{D} \tag{10}$$

From here on, we use $p^{\text{win}}(\boldsymbol{x}_j|\boldsymbol{x}_i)$ instead of $p(\boldsymbol{x}_j|\boldsymbol{x}_i)$ as the sampling distribution. At the beginning of the pretraining, *i.e.* as long as no w similarity metrics have been computed yet, we default to using standard positive pairs generated from augmented views of the same image. Note that for a window of size 1 ($w=1$), we fall back to Equation (5) and Equation (6) from Section 3.3, *i.e.* $s_i(j) = s_i^{\text{win}}(j)$ and $p(\boldsymbol{x}_j|\boldsymbol{x}_i) = p^{\text{win}}(\boldsymbol{x}_j|\boldsymbol{x}_i)$.

3.4. Memory and compute overhead

Memory overhead In practice storing w versions of $m_i^{(e)}(j)$ with $e\in\mathcal{W}_E^w$ is not feasible when the dataset is large as it would require storing w entries for each pair of images. In the case of ImageNet-1k [15], that would require about $(1.3\text{M})^2\times 4\times w \sim 300\text{TB}$ which is infeasible[2]. However, since we sample from the similarity distribution to form positive pairs, we are only interested in the most similar images. Therefore, we can restrict the support of $m_i^{(e)}(j)$ to the K highest elements. We denote this new support as $\mathcal{S}^{(e)}$ with $|\mathcal{S}^{(e)}| = K$. Note that for every epoch e, the support of $m_i^{(e)}(j)$ is different. The similarity metric $m_i^{\text{win}}(j)$ from Equation (8) with restricted domain is obtained as follows:

$$m_i^{\text{win}}(j) = \frac{1}{w}\sum_{e\in\mathcal{W}_E^w} \mathbb{1}_{\{j\in\mathcal{S}^{(e)}\}} m_i^{(e)}(j) \tag{11}$$

where $\mathbb{1}$ denotes the indicator function and with $j\in\mathcal{S}_{\text{union}}$ and $\mathcal{S}_{\text{union}} = \bigcup_{e\in\mathcal{W}_E^w}\mathcal{S}^{(e)}$. The only difference for the similarity distribution from Equation (9) is that its support is limited to $\mathcal{S}_{\text{union}}$. Similarly for Equation (10), the only difference is that its support is limited to $\mathcal{S}_{\text{win}} = \{\boldsymbol{x}_j : j\in\mathcal{S}_{\text{union}}\}$. Taking all the above into consideration, the final algorithm AdaSim is illustrated in Algorithm 1.

Compute overhead The compute overhead is limited to the

Algorithm 1 `AdaSim`: Adaptive Similarity Bootstrapping framework

Input: $\mathcal{D}$: an unlabeled dataset, $\mathcal{T}$: a distribution over the possible augmentations, f: an encoder parametrized with weights $\boldsymbol{\theta}$, `OPTIMIZER`: an optimizer, $\mathbf{Z}\in\mathbb{R}^{N\times d}$: a zero-initialized cache ($N=|\mathcal{D}|$ and d is the dimension of the latent space), w: window size, L: self-distillation loss

Output: Trained weights

for $e\in\{1,2,\cdots \text{NB\_EPOCHS}\}$ **do**
  for $i\in[|\mathcal{D}|]$ **do**
    Sample t and t' from $\mathcal{T}$
    $\boldsymbol{z}_i = f(t(\boldsymbol{x}_i))$
    $m_i^{(e)}(j) = \left(\boldsymbol{z}_i^\top \mathbf{Z}_j\right)^{(e)}$ ▷ Eq. (7)
    `update`$(\mathbf{Z}, \boldsymbol{z}_i)$ ▷ Update cache with $\boldsymbol{z}_i$
    if $e\le w$ **then**
      $\mathcal{L} = L(t(\boldsymbol{x}_i), t'(\boldsymbol{x}_i))$
    else
      $m_i^{\text{win}}(j) = \frac{1}{w}\sum_{e'\in\mathcal{W}_e^w} m_i^{(e')}(j)$ ▷ Eq. (8)
      $s_i^{\text{win}}(j) = \frac{\exp\left(m_i^{\text{win}}(j)/\tau\right)}{\sum_k \exp\left(m_i^{\text{win}}(k)/\tau\right)}$ ▷ Eq. (9)
      $p^{\text{win}}(\boldsymbol{x}_j|\boldsymbol{x}_i) = s_i^{\text{win}}(j)$ ▷ Eq. (10)
      if $\boldsymbol{x}_i == \arg\max_{\boldsymbol{x}_j} p^{\text{win}}(\boldsymbol{x}_j|\boldsymbol{x}_i)$ **then**
        Sample $\boldsymbol{x}_{j^\star}$ from p^{win}
        $\mathcal{L} = L(t(\boldsymbol{x}_i), t'(\boldsymbol{x}_{j^\star}))$
      else
        $\mathcal{L} = L(t(\boldsymbol{x}_i), t'(\boldsymbol{x}_i))$
      end if } Adaptive sampling
    end if
    $\boldsymbol{\theta} \leftarrow$ `OPTIMIZER`$(\boldsymbol{\theta}, \nabla_{\boldsymbol{\theta}}\mathcal{L})$
  end for
end for
return $\boldsymbol{\theta}$

[2] 1.3M refers to the size of the dataset and 4 bytes are required to store a single float entry of 32 bits.

Table 1: **Supervised oracle.** p indicates the probability to sample a standard positive pair ($1-p$ is the probability to sample a supervised positive pair, see Sec. 4.4).

p	k-NN (top-1)	k-NN (top-5)	linear (top-1)	linear (top-5)
0	74.3	90.5	75.8	92.7
0.5	74.9	90.9	76.3	93.0

projection of a representation $\boldsymbol{z}$ onto the cache $\mathbf{Z}$ which is embarrassingly parallelizable on GPU. This requires about $d|\mathcal{D}| = 0.5$B operations (for ViT-S/16) which is much less than the 4.6B FLOPs in the backbone (see Appendix C).

4. Results

4.1. Rationale of the experiment design

The goal of the paper is **1)** to show that bootstrapping neighbors using a self-distillation objective can hinder the performance or **2)** even lead to collapse and **3)** ultimately propose an adaptive bootstrapping scheme which not only solves the above-mentioned issues but also improves on the baselines using standard positive pairs. To achieve this goal, we compare two self-distillation methods (SimSiam [12] and DINO [8]) with different backbones (ViT-S/16 [18] and ResNet-50 [26]) in a simple controlled setup (pretraining on ImageNet-1k [15], same hyperparameters, using only 2 global crops). For every evaluation, we compare 1) the baseline with 2) the baseline + straightforward nearest neighbor bootstrapping [19] and 3) the baseline + AdaSim.

We report results on the linear and k-NN benchmarks of ImageNet-1k which are industry standard evaluation protocols for self-supervised methods (Sec. 4.5). To evaluate how generalizable the learned features are, we further compare all methods on few-shot transfer downstream tasks (Sec. 4.6). Then, we run an ablation study on AdaSim-specific hyperparameters (Sec. 4.7) and finish with some interesting training metrics that are helpful to understand AdaSim intuitively (Sec. 4.8). The main takeaway from this section is that AdaSim avoids issues 1 and 2 incurred by straightforward nearest neighbor bootstrapping and shows performance improvements on all downstream tasks.

4.2. Evaluation benchmarks

Linear evaluation A linear layer is stacked on top of the frozen features and trained on the training set of the downstream task. We report the top-1 accuracy on the test set. For each setting, we use the evaluation protocol (*e.g.* choice of optimizer, number of training epochs *etc.*) from the corresponding baseline (SimSiam [12] or DINO [8]). To evaluate the intrinsic quality of representations, the downstream evaluation should ideally not require many learnable parameters. In the case of ResNet-50, the number of parameters in the linear layer is $1000 * d$ where $d = 2048$ which is about 2 million parameters. The following evaluations do not have any learnable parameters and are thus better suited to evaluate the intrinsic quality of the pretraining.

***k*-NN evaluation** The representation $\boldsymbol{z}$ of each image in both the training and test set is computed. Then each image in the test set gets a label assigned based on votes from the nearest neighbors in the training set. We use $k = 20$ to stay consistent with previous work and report the top-1 accuracy.

Few-Shot transfer This evaluation uses a nearest-centroid classifier (Prototypical Networks [42]). We use the code and datasets (except CIFAR-10 and CIFAR-100 because the images are only 32x32) from [20]. We consider 5-way 5-shot transfer with a query set of 15 images and average results over 600 randomly sampled few-shot episodes.

Table 2: **Linear evaluation and k-NN benchmarks on ImageNet-1k [15].** We report the performance of the proposed bootstrapping scheme in conjunction with various self-distillation methods and backbones. AdaSim is systematically compared against the settings where no bootstrapping occurs and the one using straightforward bootstrapping (+NN). "-" denotes a failure to converge.

Method	Model	Epochs	k-NN	Linear
SimSiam [12]	ResNet-50	100	57.1	68.0
SimSiam + NN	ResNet-50	100	56.2 (- 0.9)	65.9 (- 2.1)
SimSiam + AdaSim	ResNet-50	100	**57.9** (+ 0.8)	**68.1** (+ 0.1)
DINO-2 [8]	ResNet-50	100	50.2	60.0
DINO-2 + NN	ResNet-50	100	-	-
DINO-2 + AdaSim	ResNet-50	100	**50.7** (+ 0.5)	**60.1** (+ 0.1)
DINO-2 [8]	ViT-S/16	800	68.4	71.9
DINO-2 + NN	ViT-S/16	800	-	-
DINO-2 + AdaSim	ViT-S/16	800	**70.1** (+ 1.7)	**73.3** (+ 1.4)
DINO-2	ViT-B/16	800	69.2	73.5
DINO-2 + NN	ViT-B/16	800	-	-
DINO-2 + AdaSim	ViT-B/16	800	**72.7** (+ 3.5)	**75.0** (+ 1.5)

4.3. Implementation details

For both DINO and SimSiam, the same hyperparameters are used as reported on their GitHub. To make sure the size of the queue/cache does not impact the results, we implement the "baseline + NN" entries in Table 2 and Table 3 with a cache that has the size of the whole dataset. To confirm the fact that standard positive pairs are needed (see issue 1), we implement the querying of the nearest neighbor such that it cannot originate from the same image $\boldsymbol{x}_i \in \mathcal{D}$ (as is the case with a queue of small size). More implementation details can be found in Section 4.3.

4.4. Supervised oracle

As a starter, to confirm our intuition that better positive pairs lead to better performance on downstream tasks, we approximate the oracle of positive pairs $\mathcal{N}^+$ using the labels from ImageNet-1k [15]. We sample a positive pair as

Table 3: **Few-shot transfer (5-way 5-shot) using prototypical networks [42] on multiple standard datasets.** The reported metrics are top-1 accuracy for Food, SUN397, Cars, DTD and mean per-class accuracy for the other datasets. "-" denotes that the training objective does not converge, *e.g.* due to collapse. Rows corresponding to AdaSim are highlighted. **Bold** text is used for the best performing row within each block.

Method	Model	Epochs	Aircraft [35]	Caltech101 [22]	Cars [31]	DTD [14]	Flowers [37]	Food [3]	Pets [40]	SUN397 [49]	Avg
SimSiam [12]	Resnet-50	100	44.13	94.88	51.46	**78.94**	94.19	68.12	88.27	**91.12**	76.39
SimSiam + NN	Resnet-50	100	43.44	94.4	50.52	76.69	93.85	67.04	88.66	90.41	75.63 (- 0.76)
SimSiam + AdaSim	Resnet-50	100	**45.71**	**94.91**	**51.71**	78.87	**94.54**	**68.38**	**88.91**	91.0	**76.75** (+ 0.36)
DINO-2 [8]	Resnet-50	100	**40.19**	92.65	45.84	**79.58**	**90.00**	63.22	80.35	**90.58**	72.80
DINO-2 + NN	Resnet-50	100	-	-	-	-	-	-	-	-	-
DINO-2 + AdaSim	Resnet-50	100	38.80	**92.69**	**46.58**	79.54	89.68	**64.52**	**81.52**	90.41	**72.97** (+ 0.17)
DINO-2 [8]	ViT-S/16	800	52.78	98.4	56.42	81.87	96.54	76.06	96.04	94.26	81.55
DINO-2 + NN	ViT-S/16	800	-	-	-	-	-	-	-	-	-
DINO-2 + AdaSim	ViT-S/16	800	**56.54**	**98.93**	**58.04**	**82.35**	**96.96**	**77.23**	**96.54**	**94.78**	**82.67** (+ 1.12)
Supervised	Resnet-50		58.35	97.61	73.68	80.83	94.19	76.23	97.45	93.78	84.02

two random images from the same class, on top of which we still apply augmentations. Empirically, we observe that the convergence (speed) is much worse than using standard positive pairs. To speed up the convergence, we sample standard positive pairs with a certain probability. Given that it makes sense for this probability to be high at the beginning of the pretraining, we simply try a linear schedule going from 1 to p. Results for $p = 0$ and $p = 0.5$ can be found in Table 1. It can be observed that $p = 0.5$ performs better which corroborates our reasoning related to issue 1.

4.5. ImageNet-1k benchmarks

The k-NN and linear evaluation results on ImageNet-1k [15] are reported in Table 2. The last block in blue shows the best performing setting ($\tau = 0.2$, $w = 50$, $K = 10$) from the ablation in Table 4 with a long pretraining schedule of 800 epochs. The first 2 blocks of rows are trained with a window size of $w = 1$ (and $\tau = 0.2$, $K = 10$) to confirm that AdaSim does not require a large window to improve the baseline and avoid collapse. DINO-2 denotes DINO with only 2 global crops. First, we can observe that with DINO-2 [8], straightforward nearest neighbor bootstrapping (NN) does not converge (illustrated with "-"). This is confirmed for different backbones and training schedules. DINO-2 + AdaSim does converge and improves the baselines. The training objective of SimSiam [12] + NN does converge but suffers from a performance impact.

4.6. Few-shot transfer

The results of the few-shot transfer are shown in Table 3. Conclusions analogous to Section 4.5 can be drawn: AdaSim improves the downstream performance on most datasets and on average (last column). For a point of comparison, Table 3 contains a row "Supervised" taken from [20] which is obtained with the weights from the supervised ResNet-50 in torchvision. Interestingly, DINO-2+AdaSim performs much better than the baseline DINO-2 on datasets where the supervised method also performs better *e.g.* Cars [31] (+5.06) or Aircraft [35] (+2.54). This is because AdaSim bootstraps neighbors in the latent space which acts as a sort of self-labeling and therefore shares some properties with the supervised method.

4.7. Ablations

An ablation study over AdaSim specific hyperparameters (τ, w, K) can be found in Table 4. The best hyperparameters are highlighted in bold. These bold parameters are used for all runs, except for the parameter that is being varied. Importantly, **for a temperature $\tau = 0$, AdaSim behaves like a standard self-distillation** method using positive pairs of the form $(t(\boldsymbol{x}), t'(\boldsymbol{x}))$. A performance improvement can be observed for increasing temperature values which shows the merits of AdaSim.

4.8. Under the hood analysis

Multiple training metrics are shown in Figure 3 with varying temperature values. Such plots are useful to build intuition on the internal mechanisms of AdaSim.

Neighbor bootstrapping ratio indicates the percentage of positive pairs $(t(\boldsymbol{x}_i), t'(\boldsymbol{x}_{j^\star}))$ where the augmentations are from different images. The higher the temperature, the higher the percentage is. In the limit when $\tau \to 0$, it can be observed that this percentage goes to 0, and AdaSim defaults to standard self-distillation. This is only possible thanks to the adaptive sampling of positive pairs in AdaSim (lines 13 to 18 in Algorithm 1). Without the adaptive sampling, a low temperature would lead to a positive pair $(t(\boldsymbol{x}_i), t'(\boldsymbol{x}_{j^\star}))$ where $\boldsymbol{x}_{j^\star} = \arg\max p^{\text{win}}(\boldsymbol{x}_j|\boldsymbol{x}_i)$ but there is no guarantee that $\boldsymbol{x}_i = \boldsymbol{x}_{j^\star}$. The adaptivity of the proposed method can be observed in Figure 3.a. Indeed, at epoch 50, the window is filled and nearest neighbor bootstrapping is allowed to occur. As the quality of the latent space is low, so is that of the resulting gradients, which temporarily hurts the learned representations. Thanks to the adaptivity criterion, the bootstrapping ratio is automatically reduced to avoid collapse.

Table 4: **Ablation study over AdaSim specific hyperparameters.** The ablation is run over 800 epochs. If not otherwise specified, the values of the hyperparameters are $(\tau, w, K) = (0.2, 10, 3)$.

	Temperature (τ)					Window size (w)			Support size (K)				
	0.0	0.05	0.1	0.2	0.4	1	10	50	2	3	5	10	20
k-NN	68.4	68.8	69.0	**70.1**	69.8	69.3	**70.1**	69.4	69.4	**70.1**	70.1	69.1	68.4
linear	71.9	72.4	72.6	**73.3**	73.2	72.6	**73.3**	72.7	72.8	**73.3**	73.0	72.5	72.2

NN top-1 training accuracy shows how often the query image $\boldsymbol{x}_i$ and its "nearest neighbor" $\boldsymbol{x}_{j^\star}$ are from the same class. Here we observe that a higher temperature leads to a lower accuracy which makes sense because the "nearest neighbor" can be the same image and, therefore, would trivially be in the same class. Note that before epoch 50, all temperature values use the same positive pairs as $w = 50$ similarity metrics are being computed.

2-NN top-1 training accuracy shows if the query image $\boldsymbol{x}_i$ and its second "nearest neighbor" $\arg\max_{\boldsymbol{x}_j \neq \boldsymbol{x}_{j^\star}} p^{\text{win}}(\boldsymbol{x}_j|\boldsymbol{x}_i)$ are from the same class. This metric is a better indicator of the downstream generalizability of the learned features. It can be observed that higher temperature values (more neighbor bootstrapping) are initially worse but start to become advantageous as the training progresses. This is intuitive because bootstrapping neighbors is only useful when they are semantically related, which only happens as the network learns.

Visualization of positive pairs To get an understanding of the positive pairs which are formed by AdaSim, we visualize multiple query images $\boldsymbol{x}_i$ along with the sampling distribution $p^{\text{win}}(\boldsymbol{x}_j|\boldsymbol{x}_i)$ (overlayed in green) and its associated support $\mathcal{S}_{\text{win}}$ in Figure 1. In this example, all "nearest neighbors" are the same as the query image, and all neighbors seem to share semantic content. In Appendix F, we explicitly search for query images where the neighbors are from different classes. These results show evidence of wrongly labeled or duplicate images in ImageNet-1k [15].

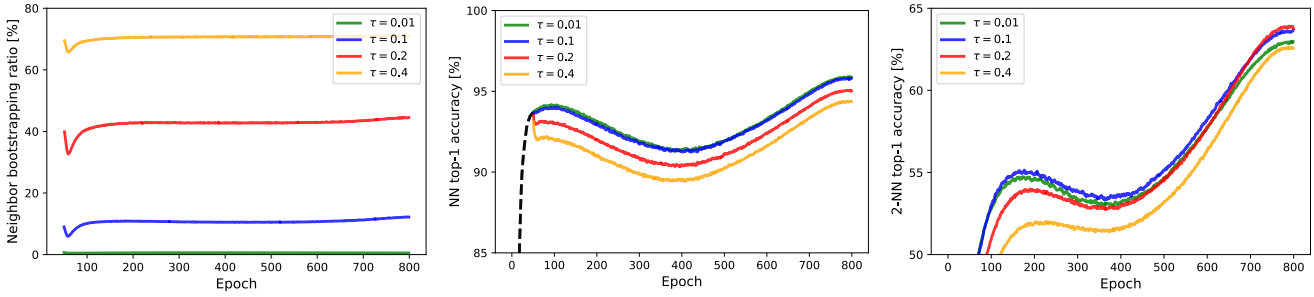
5. Conclusion

Self-distillation is becoming the go-to self-supervised learning paradigm due to its simplicity and state-of-the-art performance. However, non-explicit processing of negative pairs makes it less robust and more prone to collapse to trivial solutions than contrastive learning. Used in conjunction with bootstrapped positive pairs of neighbors, we empirically observe that self-distillation methods can perform worse than their vanilla baseline and in some cases even collapse. We propose an adaptive bootstrapping scheme that stabilizes the training and improves on the baselines. We also observe that long training schedules and larger backbones are particularly beneficial for AdaSim (better representations lead to better bootstrapping).

Limitations All results in the paper do not include multi-crop [7] for simplicity. In practice, not using multi-crop requires the use of more diverse random cropping (*e.g.* with scale sampled in $[0.1, 1]$) but we have not changed any hyperparameters from DINO and stuck with $[0.25, 1]$.

Acknowledgement

This project is funded by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation program (Grant Agreement No. 101021347). This work is also partially funded by the Personalized Health and Related Technologies (PHRT), grant number 2021/344. We acknowledge EuroCC Belgium for awarding this project access to the LUMI supercomputer, owned by the EuroHPC Joint Undertaking, hosted by CSC (Finland) and the LUMI consortium.



(a) Neighbor bootstrapping ratio (b) NN top-1 training accuracy (c) 2-NN top-1 training accuracy

Figure 3: **Visualization of multiple training metrics for different temperature τ values.**

References

[1] Yuki Markus Asano, Christian Rupprecht, and Andrea Vedaldi. Self-labelling via simultaneous clustering and representation learning. *arXiv preprint arXiv:1911.05371*, 2019. 2

[2] Piotr Bojanowski and Armand Joulin. Unsupervised learning by predicting noise. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70, pages 517–526. PMLR, 06–11 Aug 2017. 2

[3] Lukas Bossard, Matthieu Guillaumin, and Luc Van Gool. Food-101 – mining discriminative components with random forests. In David Fleet, Tomas Pajdla, Bernt Schiele, and Tinne Tuytelaars, editors, *Computer Vision – ECCV 2014*, pages 446–461, Cham, 2014. Springer International Publishing. 7

[4] Mathilde Caron, Piotr Bojanowski, Armand Joulin, and Matthijs Douze. Deep clustering for unsupervised learning of visual features. *CoRR*, abs/1807.05520, 2018. 2

[5] Mathilde Caron, Piotr Bojanowski, Julien Mairal, and Armand Joulin. Leveraging large-scale uncurated data for unsupervised pre-training of visual features. *CoRR*, abs/1905.01278, 2019. 2

[6] Mathilde Caron, Ishan Misra, Julien Mairal, Priya Goyal, Piotr Bojanowski, and Armand Joulin. Unsupervised learning of visual features by contrasting cluster assignments. In *Thirty-fourth Conference on Neural Information Processing Systems (NeurIPS)*, 2020. 2

[7] Mathilde Caron, Ishan Misra, Julien Mairal, Priya Goyal, Piotr Bojanowski, and Armand Joulin. Unsupervised learning of visual features by contrasting cluster assignments. *CoRR*, abs/2006.09882, 2020. 8

[8] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. *CoRR*, abs/2104.14294, 2021. 2, 3, 6, 7, 11, 12

[9] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 1597–1607. PMLR, 13–18 Jul 2020. 1

[10] Ting Chen, Simon Kornblith, Kevin Swersky, Mohammad Norouzi, and Geoffrey E. Hinton. Big self-supervised models are strong semi-supervised learners. *CoRR*, abs/2006.10029, 2020. 1, 2

[11] Xinlei Chen, Haoqi Fan, Ross Girshick, and Kaiming He. Improved baselines with momentum contrastive learning. *arXiv preprint arXiv:2003.04297*, 2020. 1, 2

[12] Xinlei Chen and Kaiming He. Exploring simple siamese representation learning. *CoRR*, abs/2011.10566, 2020. 2, 3, 6, 7, 11

[13] Xinlei Chen, Saining Xie, and Kaiming He. An empirical study of training self-supervised vision transformers. *CoRR*, abs/2104.02057, 2021. 1, 2

[14] Mircea Cimpoi, Subhransu Maji, Iasonas Kokkinos, Sammy Mohamed, and Andrea Vedaldi. Describing textures in the wild. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2014. 7

[15] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*, 2009. 5, 6, 7, 8, 13

[16] Zelu Deng, Yujie Zhong, Sheng Guo, and Weilin Huang. Insclr: Improving instance retrieval with self-supervision. *CoRR*, abs/2112.01390, 2021. 2

[17] C. Doersch, A. Gupta, and A. A. Efros. Unsupervised visual representation learning by context prediction. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 1422–1430, 2015. 2

[18] 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. *CoRR*, abs/2010.11929, 2020. 6

[19] 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. pages 9568–9577. IEEE Computer Society, Oct. 2021. 2, 3, 4, 6

[20] Linus Ericsson, Henry Gouk, and Timothy M. Hospedales. How Well Do Self-Supervised Models Transfer?, Mar. 2021. arXiv:2011.13377 [cs]. 6, 7

[21] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results. http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html. 12

[22] Li Fei-Fei, R. Fergus, and P. Perona. Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories. In *2004 Conference on Computer Vision and Pattern Recognition Workshop*, pages 178–178, 2004. 7

[23] Spyros Gidaris, Praveer Singh, and Nikos Komodakis. Unsupervised representation learning by predicting image rotations. *CoRR*, abs/1803.07728, 2018. 2

[24] Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre H. Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Daniel Guo, Mohammad Gheshlaghi Azar, Bilal Piot, Koray Kavukcuoglu, Rémi Munos, and Michal Valko. Bootstrap your own latent: A new approach to self-supervised learning, 2020. 2, 3

[25] 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, 2

[26] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016. 6

[27] Olivier J Hénaff, Skanda Koppula, Jean-Baptiste Alayrac, Aaron van den Oord, Oriol Vinyals, and João Carreira. Efficient visual pretraining with contrastive detection. *arXiv preprint arXiv:2103.10957*, 2021. 2

[28] Prannay Khosla, Piotr Teterwak, Chen Wang, Aaron Sarna, Yonglong Tian, Phillip Isola, Aaron Maschinot, Ce Liu, and Dilip Krishnan. Supervised contrastive learning, 2020. 2

[29] Jaewon Kim, Jooyoung Chang, and Sang Min Park. A generalized supervised contrastive learning framework, 2022. 2

[30] Soroush Abbasi Koohpayegani, Ajinkya Tejankar, and Hamed Pirsiavash. Mean shift for self-supervised learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 10326–10335, October 2021. 2

[31] Jonathan Krause, Jia Deng, Michael Stark, and Li Fei-Fei. Collecting a Large-Scale Dataset of Fine-Grained Cars. page 2. 7

[32] Tim Lebailly and Tinne Tuytelaars. Global-local self-distillation for visual representation learning. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 1441–1450, January 2023. 2

[33] Chunyuan Li, Jianwei Yang, Pengchuan Zhang, Mei Gao, Bin Xiao, Xiyang Dai, Lu Yuan, and Jianfeng Gao. Efficient self-supervised vision transformers for representation learning. *CoRR*, abs/2106.09785, 2021. 2

[34] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014. 12

[35] Subhransu Maji, Esa Rahtu, Juho Kannala, Matthew Blaschko, and Andrea Vedaldi. Fine-grained visual classification of aircraft, 2013. 7

[36] T. N. Mundhenk, D. Ho, and B. Y. Chen. Improvements to context based self-supervised learning. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9339–9348, 2018. 2

[37] Maria-Elena Nilsback and Andrew Zisserman. Automated flower classification over a large number of classes. In *2008 Sixth Indian Conference on Computer Vision, Graphics & Image Processing*, pages 722–729, 2008. 7

[38] Mehdi Noroozi and Paolo Favaro. Unsupervised learning of visual representations by solving jigsaw puzzles. In Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling, editors, *Computer Vision – ECCV 2016*, pages 69–84, Cham, 2016. Springer International Publishing. 2

[39] Pedro O O Pinheiro, Amjad Almahairi, Ryan Benmalek, Florian Golemo, and Aaron C Courville. Unsupervised learning of dense visual representations. *Advances in Neural Information Processing Systems*, 33:4489–4500, 2020. 2

[40] Omkar M Parkhi, Andrea Vedaldi, Andrew Zisserman, and C. V. Jawahar. Cats and dogs. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 3498–3505, 2012. 7

[41] Deepak Pathak, Philipp Krähenbühl, Jeff Donahue, Trevor Darrell, and Alexei A. Efros. Context encoders: Feature learning by inpainting. *CoRR*, abs/1604.07379, 2016. 2

[42] Jake Snell, Kevin Swersky, and Richard S. Zemel. Prototypical Networks for Few-shot Learning, June 2017. arXiv:1703.05175 [cs, stat]. 6, 7

[43] Kihyuk Sohn. Improved Deep Metric Learning with Multi-class N-pair Loss Objective. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016. 3

[44] Thomas Stegmüller, Tim Lebailly, Behzad Bozorgtabar, Tinne Tuytelaars, and Jean-Philippe Thiran. Croc: Cross-view online clustering for dense visual representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7000–7009, 2023. 2, 12

[45] Aäron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *CoRR*, abs/1807.03748, 2018. 3

[46] Xinlong Wang, Rufeng Zhang, Chunhua Shen, Tao Kong, and Lei Li. Dense contrastive learning for self-supervised visual pre-training. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3024–3033, 2021. 2

[47] Xin Wen, Bingchen Zhao, Anlin Zheng, Xiangyu Zhang, and Xiaojuan Qi. Self-supervised visual representation learning with semantic grouping. In *Advances in Neural Information Processing Systems*, 2022. 2

[48] Zhirong Wu, Yuanjun Xiong, Stella X. Yu, and Dahua Lin. Unsupervised feature learning via non-parametric instance-level discrimination. *CoRR*, abs/1805.01978, 2018. 3

[49] Jianxiong Xiao, James Hays, Krista A. Ehinger, Aude Oliva, and Antonio Torralba. Sun database: Large-scale scene recognition from abbey to zoo. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 3485–3492, 2010. 7

[50] Zhenda Xie, Yutong Lin, Zhuliang Yao, Zheng Zhang, Qi Dai, Yue Cao, and Han Hu. Self-supervised learning with swin transformers. *CoRR*, abs/2105.04553, 2021. 2

[51] Xueting Yan, Ishan Misra, Abhinav Gupta, Deepti Ghadiyaram, and Dhruv Mahajan. Clusterfit: Improving generalization of visual representations. *CoRR*, abs/1912.03330, 2019. 2