Adaptive Similarity Bootstrapping for Self-Distillation based Representation Learning

Tim Lebailly\textsuperscript{1*} Thomas Stegmüller\textsuperscript{2*} Behzad Bozorgtabar\textsuperscript{2,3} Jean-Philippe Thiran\textsuperscript{2,3} Tinne Tuytelaars\textsuperscript{1}  
\textsuperscript{1}KU Leuven \textsuperscript{2}EPFL \textsuperscript{3}CHUV  
\texttt{{firstname}.lastname@esat.kuleuven.be} \texttt{{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 \cite{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.
More recently, self-distillation methods have been gaining traction \cite{12,8,24,33}. These similarity maximization algorithms avoid trivial solutions by using asymmetry. This asymmetry can take the form of an additional predictor \cite{12,24} on one branch, using stop-gradients \cite{12,8,33,24}, a momentum encoder \cite{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 \cite{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 \cite{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 \cite{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 Adaptive learning algorithm based on Similarity 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 \cite{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 \cite{38}, image rotation prediction \cite{23} and more \cite{17,36,41,2}. Recently, there has been a shift towards learning features that are invariant to semantic preserving data augmentations \cite{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 \cite{27,32,39,44,47,46}.

**Neighbor bootstrapping** In order to generate strong semantic positive pairs less reliant on heuristics, NNCLR \cite{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, \cite{30} proposes to bootstrap multiple neighbors for a single query.

**Clustering methods** Clustering methods \cite{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 \cite{25,11,13} reducing the need for (very) large batch sizes. \cite{19} uses memory banks for mining positives while \cite{16} makes use of memory banks for mining both positives and negatives.
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 \( z \in \mathcal{Z} \) of an image \( x \in \mathcal{X} \), i.e., \( z = f(x) \) 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 \((x, x^+) \in \mathcal{N}^+\), an oracle \( \mathcal{N}^- \) indicating valid negative pairs of images \((x, x^-) \in \mathcal{N}^-\) 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 \( d(f(x), f(x^+)) \) and the repelling terms are of the form \( d(f(x), f(x^-)) \). 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 \( s^+ = f(x)^T f(x^+) \) and \( s^- = f(x)^T f(x^-) \). \((x, x^+)\) is sampled from \( \mathcal{N}^+ \) and \((x, x^-)\) 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 \( x \).

**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 of 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

\[
\mathcal{L}_{\text{distil}} = d(f(x), f'(x^+)) \tag{2}
\]

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

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., \((t(x), t'(x))\) 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 \((z, z')\) of the same image \( x \) and a FIFO queue \( Q \) of previously computed representations (with \( |Q| < |\mathcal{D}| \)), positive pairs are defined as \((z', \text{NN}(z, Q))\), where the nearest neighbor operator is defined as:

\[
\text{NN}(z, Q) = \arg\min_{q \in Q} \|z - q\|_2 \tag{3}
\]
Note here that the positive pairs are defined in the latent space $Z$ 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-supervision 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 $D$ 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 $Z \in \mathbb{R}^{N \times d}$, 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 $z_i$ of an augmentation of the $i$-th image $x_i \in D$ (i.e. $f(t(x_i))$ with $t \sim \mathcal{T}$) is updated in the cache. As such, $Z$ holds a latent representation for every image in the dataset at all times. Given a latent representation $z_i$ of the $i$-th image and the cache $Z$, we can define a similarity metric $m_i(j)$ between image $i$ and all images $x_j$:

$$m_i(j) = z_i^T Z_j$$  \hspace{1cm} (4)

where $Z_j$ 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:

$$s_i(j) = \frac{\exp(m_i(j)/\tau)}{\sum_{k \in [|D|]} \exp(m_i(k)/\tau)}, \quad i, j \in [|D|]$$  \hspace{1cm} (5)

where $\tau$ 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:

$$p(x_j|x_i) = s_i(j), \quad x_i, x_j \in D$$  \hspace{1cm} (6)

To approximate the oracle $N^+$ 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 $(t(x_i), t'(x_j))$ with $x_j$ sampled from $p(x_j|x_i)$ and with $t$ and $t'$ sampled from $\mathcal{T}$. Note that $Z$ contains features for all images, not excluding image $x_i$. 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 $(t(x_i), t'(x_j))$ 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 $\tau$.

#### 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 are mapped via the encoder $f$. If this distance is low compared to that of the latent representations of other images in the cache $Z$, then it means that the encoder $f$ is good at mapping images similar to $x_i$ close together. In that case, we can expect the vicinity of the queried image $x_i$ to also share semantic content with image $x_i$ and can therefore use elements of the vicinity to form a positive pair with $x_i$. If this distance is too high, we default to a standard positive pair composed of two augmentations of the same input image. Mathematically, if $\arg \max_{x'} p(x_j|x_i) = x_i$, then we sample $x_j$, from $p(x_j|x_i)$ and use a positive pair $(t(x_i), t'(x_j))$ with $t$ and $t'$ sampled from $\mathcal{T}$. Otherwise, we use a standard positive pair $(t(x_i), t'(x_j))$.

#### 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
As long as no \( w \) similarity metrics have been computed yet, we default to using standard positive pairs generated from 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. \( s^{(e)}(j) = s^{\text{win}}(j) \) and \( p(x_j|x_i) = p^{\text{win}}(x_j|x_i) \).

### 3.4. Memory and compute overhead

**Memory overhead**

In practice storing \( w \) versions of \( m_i^{(e)}(j) \) with \( e \in \mathcal{W}_E \) 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 \((1.3M)^2 \times 4 \times w \approx 300TB\) which is infeasible. 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 \( S^{(e)} \) with \( |S^{(e)}| = K \). 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:

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

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

**Compute overhead**

The compute overhead is limited to the
projection of a representation $z$ onto the cache $Z$ which is embarrassingly parallelizable on GPU. This requires about $d/D = 0.5B$ 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 (pre-training 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 $1000 \times d$ 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 $z$ 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.

#### 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 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 $x_i \in \mathcal{D}$ (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 $X^+$ using the labels from ImageNet-1k [15]. We sample a positive pair as

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

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).

<table>
<thead>
<tr>
<th>$p$</th>
<th>$k$-NN (top-1)</th>
<th>$k$-NN (top-5)</th>
<th>linear (top-1)</th>
<th>linear (top-5)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>74.3</td>
<td>90.5</td>
<td>75.8</td>
<td>92.7</td>
</tr>
<tr>
<td>0.5</td>
<td>74.9</td>
<td>90.9</td>
<td>76.3</td>
<td>93.0</td>
</tr>
</tbody>
</table>
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.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>SimSiam [12]</td>
<td>ResNet-50</td>
<td>100</td>
<td>44.13</td>
<td>94.88</td>
<td>51.46</td>
<td>78.94</td>
<td>94.19</td>
<td>68.12</td>
<td>88.27</td>
<td>91.12</td>
<td>76.39</td>
</tr>
<tr>
<td>SimSiam + NN</td>
<td>ResNet-50</td>
<td>100</td>
<td>43.44</td>
<td>94.4</td>
<td>50.52</td>
<td>76.69</td>
<td>93.85</td>
<td>67.04</td>
<td>88.66</td>
<td>90.41</td>
<td>75.63 (-0.76)</td>
</tr>
<tr>
<td>SimSiam + AdaSim</td>
<td>ResNet-50</td>
<td>100</td>
<td>45.71</td>
<td>94.91</td>
<td>51.71</td>
<td>78.87</td>
<td>94.54</td>
<td>68.38</td>
<td>88.91</td>
<td>91.0</td>
<td>76.79 (+0.30)</td>
</tr>
<tr>
<td>DINO-2 [3]</td>
<td>ResNet-50</td>
<td>100</td>
<td>40.19</td>
<td>92.65</td>
<td>45.84</td>
<td>79.58</td>
<td>90.00</td>
<td>63.22</td>
<td>80.35</td>
<td>90.58</td>
<td>72.80</td>
</tr>
<tr>
<td>DINO-2 + NN</td>
<td>ResNet-50</td>
<td>100</td>
<td>40.19</td>
<td>92.65</td>
<td>45.84</td>
<td>79.58</td>
<td>90.00</td>
<td>63.22</td>
<td>80.35</td>
<td>90.58</td>
<td>72.80</td>
</tr>
<tr>
<td>DINO-2 + AdaSim</td>
<td>ResNet-50</td>
<td>100</td>
<td>38.80</td>
<td>92.69</td>
<td>46.58</td>
<td>79.54</td>
<td>89.68</td>
<td>64.52</td>
<td>81.52</td>
<td>90.41</td>
<td>72.97 (+0.17)</td>
</tr>
<tr>
<td>DINO-2 [8]</td>
<td>ViT-S/16</td>
<td>800</td>
<td>52.78</td>
<td>98.4</td>
<td>56.42</td>
<td>81.87</td>
<td>96.54</td>
<td>76.06</td>
<td>96.04</td>
<td>94.26</td>
<td>81.55</td>
</tr>
<tr>
<td>DINO-2 + NN</td>
<td>ViT-S/16</td>
<td>800</td>
<td>52.78</td>
<td>98.4</td>
<td>56.42</td>
<td>81.87</td>
<td>96.54</td>
<td>76.06</td>
<td>96.04</td>
<td>94.26</td>
<td>81.55</td>
</tr>
<tr>
<td>DINO-2 + AdaSim</td>
<td>ViT-S/16</td>
<td>800</td>
<td>56.54</td>
<td>98.93</td>
<td>58.04</td>
<td>82.35</td>
<td>96.96</td>
<td>77.23</td>
<td>96.54</td>
<td>94.78</td>
<td>82.67 (+1.12)</td>
</tr>
<tr>
<td>Supervised</td>
<td>ResNet-50</td>
<td></td>
<td>58.35</td>
<td>97.61</td>
<td>73.68</td>
<td>80.83</td>
<td>94.19</td>
<td>76.23</td>
<td>97.45</td>
<td>93.78</td>
<td>84.02</td>
</tr>
</tbody>
</table>

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 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 and on average (last column).
### Ablation study over AdaSim specific hyperparameters.

The ablation is run over 800 epochs. If not otherwise specified, the values of the hyperparameters are $\tau = 0.2$, $w = 10$, and $K = 3$.

<table>
<thead>
<tr>
<th>Temperature ($\tau$)</th>
<th>Window size ($w$)</th>
<th>Support size ($K$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>0.05</td>
<td>5</td>
<td>3</td>
</tr>
<tr>
<td>0.1</td>
<td>10</td>
<td>5</td>
</tr>
<tr>
<td>0.2</td>
<td>50</td>
<td>10</td>
</tr>
<tr>
<td>0.4</td>
<td>20</td>
<td>20</td>
</tr>
</tbody>
</table>

- **k-NN top-1 training accuracy** shows how often the query image $x_i$ and its “nearest neighbor” $x_j$ 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 $x_i$ and its second “nearest neighbor” $\arg\max_{x_j \neq x_i} p^{win}(x_j | x_i)$ 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 $x_i$ along with the sampling distribution $p^{win}(x_j | x_i)$ (overlayed in green) and its associated support $S^{win}$ 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 $[0, 1]$) 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.

![Image](image_url)
References


