

---

# Whitening for Self-Supervised Representation Learning

---

Aleksandr Ermolov<sup>1</sup> Aliaksandr Siarohin<sup>1</sup> Enver Sangineto<sup>1</sup> Nicu Sebe<sup>1</sup>

## Abstract

Most of the current self-supervised representation learning (SSL) methods are based on the *contrastive loss* and the instance-discrimination task, where augmented versions of the same image instance (“positives”) are contrasted with instances extracted from other images (“negatives”). For the learning to be effective, many negatives should be compared with a positive pair, which is computationally demanding. In this paper, we propose a different direction and a new loss function for SSL, which is based on the *whitening* of the latent-space features. The whitening operation has a “scattering” effect on the batch samples, avoiding degenerate solutions where all the sample representations collapse to a single point. Our solution does not require asymmetric networks and it is conceptually simple. Moreover, since negatives are not needed, we can extract multiple positive pairs from the same image instance. The source code of the method and of all the experiments is available at: <https://github.com/htdt/self-supervised>.

## 1. Introduction

One of the current main bottlenecks in deep network training is the dependence on large annotated training datasets, and this motivates the recent surge of interest in unsupervised methods. Specifically, in self-supervised representation learning (SSL), a network is (pre-)trained without any form of manual annotation, thus providing a means to extract information from unlabeled-data sources (e.g., text corpora, videos, images from the Internet, etc.). In self-supervision, label-based information is replaced by a prediction problem using some form of *context* or using a *pretext* task. Pioneering work in this direction was done in Natural Language Processing (NLP), in which the co-occurrence

of words in a sentence is used to learn a language model (Mikolov et al., 2013a;b; Devlin et al., 2019). In Computer Vision, typical contexts or pretext tasks are based on: (1) the temporal consistency in videos (Wang & Gupta, 2015; Misra et al., 2016; Dwibedi et al., 2019), (2) the spatial order of patches in still images (Noroozi & Favaro, 2016; Misra & van der Maaten, 2019; Hénaff et al., 2019) or (3) simple image transformation techniques (Ji et al., 2019; He et al., 2019; Wu et al., 2018). The intuitive idea behind most of these methods is to collect pairs of *positive* and *negative* samples: two positive samples should share the same semantics, while negatives should be perceptually different. A triplet loss (Sohn, 2016; Schroff et al., 2015; Hermans et al., 2017; Wang & Gupta, 2015; Misra et al., 2016) can then be used to learn a metric space representing the human perceptual similarity. However, most of the recent studies use a contrastive loss (Hadsell et al., 2006) or one of its variants (Gutmann & Hyvärinen, 2010; van den Oord et al., 2018; Hjelm et al., 2019), while Tschannen et al. (2019) show the relation between the triplet and the contrastive losses.

It is worth noticing that the success of both kinds of losses is strongly affected by the number and the quality of the negative samples. For instance, in the case of the triplet loss, a common practice is to select *hard/semi-hard* negatives (Schroff et al., 2015; Hermans et al., 2017). On the other hand, Hjelm et al. (2019) have shown that the contrastive loss needs a large number of negatives to be competitive. This implies using batches with a large size, which is computationally demanding, especially with high-resolution images. In order to alleviate this problem, Wu et al. (2018) use a *memory bank* of negatives, which is composed of feature-vector representations of all the training samples. He et al. (2019) conjecture that the use of large and fixed-representation vocabularies is one of the keys to the success of self-supervision in NLP. The solution proposed in MoCo He et al. (2019) extends Wu et al. (2018) using a memory-efficient queue of the last visited negatives, together with a *momentum encoder* which preserves the intra-queue representation consistency. Chen et al. (2020a) have performed large-scale experiments confirming that a large number of negatives (and therefore a large batch size) is required for the contrastive loss to be efficient.

Very recently, Grill et al. (2020) proposed an alternative

---

<sup>1</sup>Department of Information Engineering and Computer Science (DISI), University of Trento, Italy. Correspondence to: Aleksandr Ermolov <[aleksandr.ermolov@unitn.it](mailto:aleksandr.ermolov@unitn.it)>.

direction, in which only positives are used, together with two networks, where the *online* network tries to predict the representation of a positive extracted by the *target* network. Despite the large success of BYOL (Grill et al., 2020), the reason why the two networks can avoid a collapsed representation (e.g., where all the images are mapped to the same point) is still unclear (Fetterman & Albrecht, 2020; Tian et al., 2020b; Chen & He, 2020; Richemond et al., 2020). According to (Fetterman & Albrecht, 2020; Tian et al., 2020b), one of the important ingredients which is *implicitly* used in BYOL to avoid degenerate solutions, is the use of the Batch Norm (BN) (Ioffe & Szegedy, 2015) in the projection/prediction heads (see Sec. 3.1). In this paper we propose to generalize this finding and we show that, using a full *whitening* of the latent space features is *sufficient* to avoid collapsed representations, without the need of additional momentum networks (Grill et al., 2020), siamese networks with stop-gradient operations (Chen & He, 2020) or the use of specific, batch-based optimizers like LARS (You et al., 2017; Fetterman & Albrecht, 2020).

In more detail, we propose a new SSL loss function, which first *scatters* all the sample representations in a spherical distribution<sup>1</sup> and then *penalizes* the positive pairs which are far from each other. Specifically, given a set of samples  $V = \{\mathbf{v}_i\}$ , corresponding to the current mini-batch of images  $B = \{x_i\}$ , we first project the elements of  $V$  onto a spherical distribution using a *whitening* transform (Siarohin et al., 2019). The whitened representations  $\{\mathbf{z}_i\}$ , corresponding to  $V$ , are normalized and then used to compute a Mean Squared Error (MSE) loss which accumulates the error considering only positive pairs  $(\mathbf{z}_i, \mathbf{z}_j)$ . To avoid a representation collapse, we do not need to *contrast* positives against negatives as in the contrastive loss or in the triplet loss because the optimization process leads to shrinking the distance between positive pairs and, indirectly, scatters the other samples to satisfy the overall spherical-distribution constraint.

In summary, our contributions are the following:

- We propose a new SSL loss function, Whitening MSE (W-MSE). W-MSE constrains the batch samples to lie in a spherical distribution and it is an alternative to positive-negative instance contrasting methods.
- Our loss does not need a large number of negatives, thus we can include more positives in the current batch. We indeed demonstrate that multiple positive pairs extracted from one image improve the performance.
- We empirically show that our W-MSE loss outperforms the commonly adopted contrastive loss and it is com-

petitive with respect to state-of-the-art SSL methods like (Grill et al., 2020; Chen & He, 2020).

## 2. Background and Related Work

A typical SSL method is composed of two main components: a *pretext task*, which exploits some a-priori knowledge about the domain to automatically extract supervision from data, and a *loss function*. In this section we briefly review both aspects, and we additionally analyse the recent literature concerning feature whitening.

**Pretext Tasks.** The temporal consistency in a video provides an intuitive form of self-supervision: temporally-close frames usually contain a similar semantic content (Wang & Gupta, 2015; van den Oord et al., 2018). Misra et al. (2016) extended this idea using the relative temporal order of 3 frames, while Dwibedi et al. (2019) used a *temporal cycle consistency* for self-supervision, which is based on comparing two videos sharing the same semantics and computing inter-video frame-to-frame nearest neighbour assignments.

When dealing with still images, the most common pretext task is *instance discrimination* (Wu et al. (2018)): from a training image  $x$ , a composition of data-augmentation techniques are used to extract two different views of  $x$  ( $x_i$  and  $x_j$ ). Commonly adopted transformations are: image cropping, rotation, color jittering, Sobel filtering, etc. The learner, which is usually composed of an encoder and a projection head, is then required to discriminate  $(x_i, x_j)$  from other views extracted from other samples (Wu et al., 2018; Ji et al., 2019; He et al., 2019; Chen et al., 2020a).

Denoising auto-encoders (Vincent et al., 2008) add random noise to the input image and try to recover the original image. Xu et al. (2020) enforce consistency across different image resolutions. More sophisticated pretext tasks consist in predicting the spatial order of image patches (Noroozi & Favaro, 2016; Misra & van der Maaten, 2019) or in reconstructing large masked regions of the image (Pathak et al., 2016). Hjelm et al. (2019); Bachman et al. (2019) compare the holistic representation of an input image with a patch of the same image. Hénaff et al. (2019) use a similar idea, where the comparison depends on the patch order: the appearance of a given patch should be predicted given the appearance of the patches which lie above it in the image.

We use standard data augmentations (Chen et al., 2020a) to get positive pairs, which is a simple solution and does not require a pretext-task specific network architecture (Hjelm et al., 2019; Bachman et al., 2019; Hénaff et al., 2019).

**Loss functions.** Denoising auto-encoders use a *reconstruction loss* which compares the generated image with the input image before adding noise. Other generative methods use an *adversarial loss* in which a discriminator provides supervisory information to the generator (Donahue et al., 2017;

<sup>1</sup>Here and in the following, with “spherical distribution” we mean a distribution with a zero-mean and an identity-matrix covariance.

Donahue & Simonyan, 2019).

Early SSL (deep) discriminative methods used a *triplet loss* (Wang & Gupta, 2015; Misra et al., 2016): given two *positive* images  $x_i, x_j$  and a *negative*  $x_k$  (Sec. 1), together with their corresponding latent-space representations  $\mathbf{z}_i, \mathbf{z}_j, \mathbf{z}_k$ , this loss penalizes those cases in which  $\mathbf{z}_i$  and  $\mathbf{z}_k$  are closer to each other than  $\mathbf{z}_i$  and  $\mathbf{z}_j$  plus a margin  $m$ :

$$L_{Triplet} = -\max(\mathbf{z}_i^T \mathbf{z}_k - \mathbf{z}_i^T \mathbf{z}_j + m, 0). \quad (1)$$

Most of the recent SSL discriminative methods are based on some *contrastive loss* (Hadsell et al., 2006) variant, in which  $\mathbf{z}_i$  and  $\mathbf{z}_j$  are contrasted against a set of negative pairs. Following the common formulation proposed by van den Oord et al. (2018), the contrastive loss is given by:

$$L_{Contrastive} = -\log \frac{\exp(\mathbf{z}_i^T \mathbf{z}_j / \tau)}{\sum_{k=1, k \neq i}^K \exp(\mathbf{z}_i^T \mathbf{z}_k / \tau)}, \quad (2)$$

where  $\tau$  is a *temperature* hyperparameter which should be manually set and the sum in the denominator is over a set of  $K - 1$  negative samples. Usually  $K$  is the size of the current batch, i.e.,  $K = 2N$ , being  $N$  the number of positive pairs. However, as shown by Hjelm et al. (2019), the contrastive loss (2) requires a large number of negative samples to be competitive. Wu et al. (2018); He et al. (2019) use a set of negatives much larger than the current batch, by pre-computing representations of old samples. SimCLR (Chen et al., 2020a) uses a simpler, but computationally very demanding, solution based on large batches.

Tschannen et al. (2019) show that the success of the contrastive loss is likely related to learning a metric space, similarly to what happens with a triplet loss, while Wang & Isola (2020) investigate the uniformity and the alignment properties of the  $L_2$  normalized contrastive loss. In the same paper, the authors propose two new losses ( $\mathcal{L}_{\text{uniform}}$  and  $\mathcal{L}_{\text{align}}$ ) which explicitly optimize these characteristics.

In BYOL (Grill et al. (2020)), given a pair of positives  $(x_i, x_j)$ ,  $x_i$  is fed to the “online” network, which should predict the output of the “target” network, where the latter receives  $x_j$  as input and its parameters are a running average of the former. Concurrently with our work, Chen & He (2020) have simplified this scheme, introducing SimSiam, where both samples  $x_i$  and  $x_j$  are encoded using the same network (i.e., a Siamese architecture with a shared encoder). Both BYOL and SimSiam include an additional asymmetric prediction head to compare the latent representations of  $x_i$  and  $x_j$ , a stop-gradient operation with respect to the element of the positive pair not used by the prediction head, and an MSE loss of the  $L_2$ -normalized latent representations. In these methods, both the projection and the prediction head

include BN. Chen & He (2020) show that these BN layers are a crucial component of SimSiam, and their removal results in a dramatic performance degradation. Fetterman & Albrecht (2020) and Tian et al. (2020b) have empirically confirmed the importance of BN in BYOL, and they show that BN allows BYOL to avoid collapsing the representation of all images to a constant value, which would make the MSE computation equal to zero. Our work can be seen as a generalization of this finding *with a much simpler network architecture* and without the need to rely on asymmetric learning protocols (see Sec. 3.1). Moreover, our loss formulation is simpler also because it does not require a proper setting of hyperparameters such as  $\tau$  in Eq. 2 or  $m$  in Eq. 1.

Finally, another recent line of work is based on clustering approaches (Bautista et al., 2016; Caron et al., 2018; Zhuang et al., 2019). For instance, SwAV (Caron et al., 2020) computes a cross-entropy loss over the image-to-cluster prototype assignments, and it is one of the state-of-the-art SSL methods. One of the ingredients of SwAV is the use of multiple positives, and in Sec. 3.1 we show that our proposal can exploit multiple crops in a more efficient way.

**Feature Whitening.** We adopt the efficient and stable Cholesky decomposition (Dereniowski & Marek, 2004) based *whitening* transform proposed by Siarohin et al. (2019) to project our latent-space vectors into a spherical distribution (see Sec. 3). Note that Huang et al. (2018); Siarohin et al. (2019) use whitening transforms in the intermediate layers of the network for a completely different task: extending BN to a multivariate batch normalization.

### 3. The Whitening MSE Loss

Given an image  $x$ , we extract an embedding  $\mathbf{z} = f(x; \theta)$  using an encoder network  $f(\cdot; \theta)$  parametrized with  $\theta$  (more details below). We require that: (1) the image embeddings are drawn from a non-degenerate distribution (the latter being a distribution where, e.g., all the representations collapse to a single point), and (2) positive image pairs  $(x_i, x_j)$ , which share a similar semantics, should be clustered close to each other. We formulate this problem as follows:

$$\min_{\theta} \mathbb{E}[\text{dist}(\mathbf{z}_i, \mathbf{z}_j)], \quad (3)$$

$$s.t. \text{cov}(\mathbf{z}_i, \mathbf{z}_i) = \text{cov}(\mathbf{z}_j, \mathbf{z}_j) = I, \quad (4)$$

where  $\text{dist}(\cdot)$  is a distance between vectors,  $I$  is the identity matrix and  $(\mathbf{z}_i, \mathbf{z}_j)$  corresponds to a positive pair of images  $(x_i, x_j)$ . With Eq. 4, we constrain the distribution of the  $\mathbf{z}$  values to be non-degenerate, hence avoiding that all the probability mass is concentrated in a single point. Moreover, Eq. 4 makes all the components of  $\mathbf{z}$  to be linearly independent from each other, which encourages the different dimensions of  $\mathbf{z}$  to represent different semantic content. We

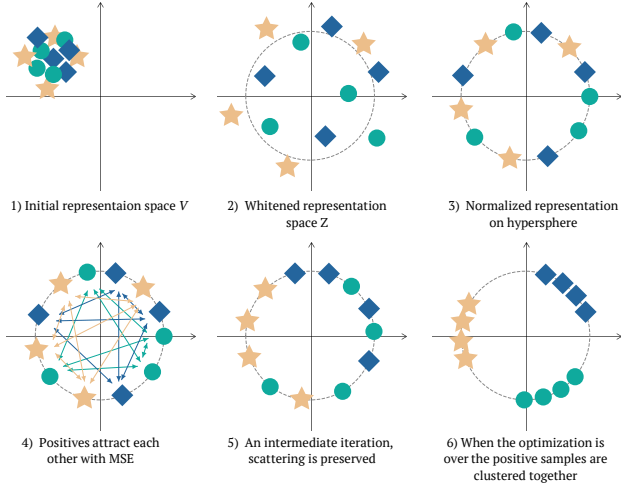


Figure 1. A schematic representation of the W-MSE based optimization process. Positive pairs are indicated with the same shapes and colors. (1) A representation of the batch features in  $V$  when training starts. (2, 3) The distribution of the elements after whitening and the  $L_2$  normalization. (4) The MSE computed over the normalized  $\mathbf{z}$  features encourages the network to move the positive pair representations closer to each other. (5) The subsequent iterations move closer and closer the positive pairs, while the relative layout of the other samples is forced to lie in a spherical distribution. (6) When the optimization is over the positive samples are clustered together.

define the distance with the cosine similarity, implemented with MSE between normalized vectors:

$$\begin{aligned} \text{dist}(\mathbf{z}_i, \mathbf{z}_j) &= \left\| \frac{\mathbf{z}_i}{\|\mathbf{z}_i\|_2} - \frac{\mathbf{z}_j}{\|\mathbf{z}_j\|_2} \right\|_2^2 \\ &= 2 - 2 \frac{\langle \mathbf{z}_i, \mathbf{z}_j \rangle}{\|\mathbf{z}_i\|_2 \cdot \|\mathbf{z}_j\|_2} \end{aligned} \quad (5)$$

In the Appendix we also include other experiments in which the cosine similarity is replaced by the Euclidean distance. We provide below the details on how positive image samples are collected, how they are encoded and how the above optimization is implemented.

First, similarly to [Chen et al. \(2020a\)](#), we obtain positive samples sharing the same semantics from a single image  $x$  and using standard image transformation techniques. Specifically, we use a composition of image cropping, grayscaling and color jittering transformations  $T(\cdot; \mathbf{p})$ . The parameters  $(\mathbf{p})$  are selected uniformly at random and independently for each *positive* sample extracted from the same image:  $x_i = T(x; \mathbf{p}_i)$ . We concisely indicate with  $\text{pos}(i, j)$  the fact that  $x_i$  and  $x_j$  ( $x_i, x_j \in B$ ,  $B$  the current batch) have been extracted from the same image.

The number of positive samples per image  $d$  may vary, trading off diversity in the batch and the amount of the training

signal. Favoring more negatives, most of the methods use only one positive pair ( $d = 2$ ). However, [Ji et al. \(2019\)](#) have demonstrated improved performance with 5 samples, while [Caron et al. \(2020\)](#) use 8 samples. In our MSE-based loss (see below), we use all the possible  $d(d - 1)/2$  combinations of positive samples. We include experiments for  $d = 2$  (1 positive pair) and  $d = 4$  (6 positive pairs). Note that our implementation includes *batch slicing* (described below), where the choice of  $d$  is related with the sub-batch size, and larger  $d$  values can produce instability issues.

For representation learning, we use a backbone *encoder* network  $E(\cdot)$ .  $E(\cdot)$ , trained without human supervision, will be used in Sec. 4 for evaluation using standard protocols. We use a standard ResNet ([He et al., 2016](#)) as the encoder, and  $\mathbf{h} = E(x)$  is the output of the average-pooling layer. This choice has the advantage to be simple and easily reproducible, in contrast to other methods which use encoder architectures specific for a given pretext task (see Sec. 2). Since  $\mathbf{h} \in \mathbb{R}^{512}$  or  $\mathbf{h} \in \mathbb{R}^{2048}$  is a high-dimensional vector, following [Chen et al. \(2020a\)](#) we use a nonlinear projection head  $g(\cdot)$  to project  $\mathbf{h}$  in a lower dimensional space:  $\mathbf{v} = g(\mathbf{h})$ , where  $g(\cdot)$  is implemented with a MLP with one hidden layer and a BN layer. The whole network  $f(\cdot)$  is given by the composition of  $g(\cdot)$  with  $E(\cdot)$  (see Fig. 2). Note that we do not use prediction heads like in ([Grill et al., 2020](#); [Chen & He, 2020](#)).

Given  $N$  original images and a batch of samples  $B = \{x_1, \dots, x_K\}$ , where  $K = Nd$ , let  $V = \{\mathbf{v}_1, \dots, \mathbf{v}_K\}$ , be the corresponding batch of features obtained as described above. In the proposed W-MSE loss, we compute the MSE over all  $Nd(d - 1)/2$  positive pairs, where constraint 4 is satisfied using the reparameterization of the  $\mathbf{v}$  variables with the whitened variables  $\mathbf{z}$ :

$$L_{W-MSE}(V) = \frac{2}{Nd(d - 1)} \sum \text{dist}(\mathbf{z}_i, \mathbf{z}_j), \quad (6)$$

where the sum is over  $(\mathbf{v}_i, \mathbf{v}_j) \in V$ ,  $\text{pos}(i, j) = \text{true}$ ,  $\mathbf{z} = \text{Whitening}(\mathbf{v})$ , and:

$$\text{Whitening}(\mathbf{v}) = W_V(\mathbf{v} - \boldsymbol{\mu}_V). \quad (7)$$

In Eq. 7,  $\boldsymbol{\mu}_V$  is the mean of the elements in  $V$ :  $\boldsymbol{\mu}_V = \frac{1}{K} \sum_k \mathbf{v}_k$ , while the matrix  $W_V$  is such that:  $W_V^\top W_V = \Sigma_V^{-1}$ , being  $\Sigma_V$  the covariance matrix of  $V$ :

$$\Sigma_V = \frac{1}{K - 1} \sum_k (\mathbf{v}_k - \boldsymbol{\mu}_V)(\mathbf{v}_k - \boldsymbol{\mu}_V)^\top. \quad (8)$$

For more details on how  $W_V$  is computed, we refer to the Appendix. Equation 7 performs the full whitening of each  $\mathbf{v}_i \in V$  and the resulting set of vectors  $Z = \{\mathbf{z}_1, \dots, \mathbf{z}_K\}$



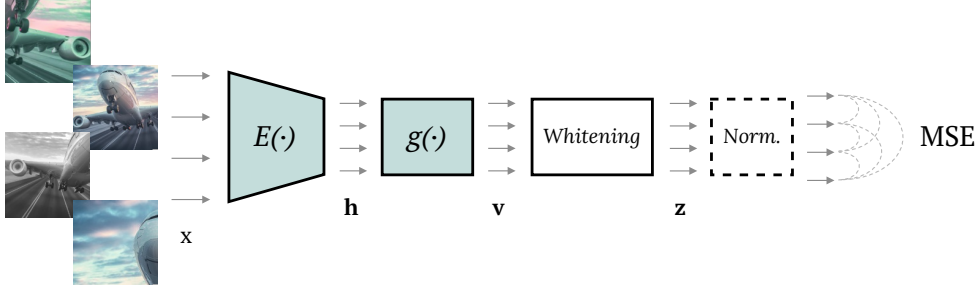


Figure 2. A scheme of our training procedure. First,  $d$  ( $d = 4$  in this case) positive samples are generated using augmentations. These images are transformed into vectors with the encoder  $E(\cdot)$ . Next, they are projected onto a lower dimensional space with a projection head  $g(\cdot)$ . Then, Whitening projects these vectors onto a spherical distribution, optionally followed by an  $L_2$  normalization. Finally, the dashed curves show all the  $d(d - 1)/2$  comparisons used in our W-MSE loss.

lies in a zero-centered distribution with a covariance matrix equal to the identity matrix (Fig. 1).

The intuition behind the proposed loss is that Eq. 6 penalizes positives which are far apart from each other, thus leading  $g(E(\cdot))$  to shrink the inter-positive distances. On the other hand, since  $Z$  must lie in a spherical distribution, the other samples should be “moved” and rearranged in order to satisfy constraint 4 (see Fig. 1).

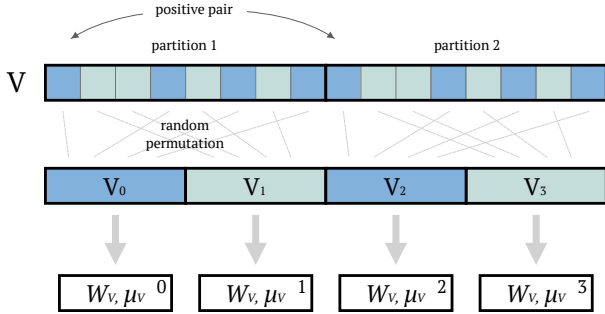


Figure 3. Batch slicing.  $V$  is first partitioned in  $d$  parts ( $d = 2$  in this example). We randomly permute the first part and we apply the same permutation to the other  $d - 1$  parts. Then, we further split all the partitions and we create sub-batches ( $V_i$ ). Each  $V_i$  is independently used to compute the sub-batch specific whitening matrix  $W_V^i$  and centroid  $\mu_V^i$ .

**Batch Slicing.** The estimation of the MSE in Eq. 6 depends on the whitening matrix  $W_V$ , which may have a high variance over consecutive-iteration batches  $V_t, V_{t+1}, \dots$ . For this reason, inspired by the resampling methods (Efron, 1982), given a batch  $V$ , we slice  $V$  in different non-overlapping sub-batches and we compute a whitening matrix independently for each sub-batch. In more detail, we first partition the batch in  $d$  parts, being  $d$  the number of positives extracted from one image. In this way, each partition contains elements extracted from different original images (i.e., no

pair of positives is included in a single partition, see Fig. 3). Then, we randomly permute the elements of the each partition, using the same permutation for all the partitions. Next, each partition is further split in sub-batches, using the heuristic that the size of each sub-batch ( $V_i$ ) should be equal to the size of embedding ( $|\mathbf{v}|$ ) times 2 (this prevents instability issues when computing the covariance matrices). For each  $V_i$ , we use only its elements to compute a corresponding whitening matrix  $W_V^i$ , which is used to whiten the elements of  $V_i$  only (Fig. 3). In the loss computation (Eq. 6), all the elements of all the sub-batches are used, thus implicitly alleviating the differences among the different whitening matrices. Finally, it is possible to repeat the whole operation several times and to average the result to get a more robust estimate of Eq. 6. Note that, despite using smaller batches may increase the instability in computing the whitening matrix, this is compensated by having a different  $W_V^i$  for each  $V_i$ , and by the possibility to iterate the sampling process with different permutations for the same  $V$ .

### 3.1. Discussion

In a common *instance-discrimination* task (Sec. 2), e.g., solved using Eq. 2, the similarity of a positive pair ( $\mathbf{z}_i^T \mathbf{z}_j$ ) is contrasted with the similarity computed with respect to all the other samples ( $\mathbf{z}_k$ ) in the batch ( $\mathbf{z}_i^T \mathbf{z}_k, 1 \leq k \leq K, k \neq i$ ). However,  $\mathbf{z}_k$  and  $\mathbf{z}_i$ , extracted from different image instances, can occasionally share the same semantics (e.g.,  $x_i$  and  $x_k$  are two different image *instances* of the unknown “cat” *class*). Conversely, the proposed W-MSE loss does not force all the instance samples to lie far from each other, but it only imposes a soft constraint (Eq. 4), which avoids degenerate distributions.

Note that previous work (He et al., 2019; Hénaff et al., 2019; Chen et al., 2020a) highlighted that BN may be harmful for learning semantically meaningful representations because the network can “cheat” and exploit the batch statistics in order to find a trivial solution to Eq. 2. However, our

whitening transform (Eq. 7) is applied only to the very last layer of the network  $f(\cdot)$  (see Fig. 2) and it is not used in the intermediate layers, which is instead the case of BN. Hence, our  $f(\cdot)$  cannot learn to exploit subtle inter-sample dependencies introduced by batch-statistics because of the lack of other learnable layers on top of the  $z$  features.

Similarly to Eq. 6, in BYOL (Grill et al., 2020) an MSE loss is used to compare the latent representations of two positives computed by slightly different networks without contrasting positives with negatives (Sec. 2). However, the MSE loss alone could be trivially minimized by a collapsed distribution, where both networks output sample-independent representations. The reason why this does not happen is still an open problem and this has opened a debate in the community (Fetterman & Albrecht, 2020; Tian et al., 2020b; Chen & He, 2020; Richemond et al., 2020). Specifically, it seems that the BN layers, included in the projection/prediction heads of BYOL are *one of the* ingredients which help to avoid degenerate solutions (Fetterman & Albrecht, 2020; Tian et al., 2020b) (see Sec. 1 and 2). Our W-MSE can be seen as a generalization of this implicit property of BYOL, in which the  $z$  values of the current batch are full-whitened, so preventing possible collapsing effects of the MSE loss. Importantly, we reach this result without the need of specific asymmetric architectures (e.g., momentum networks (Grill et al., 2020), prediction heads, etc.) or sophisticated training protocols (e.g., stop-gradient, slow-convergence of the prediction head (Chen & He, 2020), etc.).

Finally, note that using BN alone without whitening (Eq. 7), and without the aforementioned asymmetric learning, is *not sufficient*. Indeed, if we minimize the MSE after feature standardization (i.e., BN), the network can easily find a solution where, e.g., all the dimensions of the embedding represent the same feature or are correlated to each other. For instance, in preliminary experiments with CIFAR-10, we replaced whitening with BN, and the network converged to a 0 loss value after 50 epochs. Nevertheless, the linear classification accuracy was very poor: 68.15%.

Last but not least, the use of multiple positives extracted from the same image (i.e.,  $d > 2$ ) has been recently proposed in SwAV (Caron et al., 2020). However, different from our proposal, in SwAV, the multi-crop strategy is based on multiple-resolution crops, and, most importantly, in our case, we can compute  $d(d-1)$  inter-positive differences in Eq. 6 with only  $d$  forward passes, while, in SwAV, the number of comparisons grows linearly with  $d$ .

## 4. Experiments

In our experiments we use the following **datasets**.

- CIFAR-10 and CIFAR-100 (Krizhevsky & Hinton,

2009), two small-scale datasets composed of  $32 \times 32$  images with 10 and 100 classes, respectively.

- ImageNet (Deng et al., 2009), the well-known large-scale dataset with about 1.3M training images and 50K test images, spanning over 1000 classes.
- Tiny ImageNet (Le & Yang, 2015), a reduced version of ImageNet, composed of 200 classes with images scaled down to  $64 \times 64$ . The total number of images is: 100K (training) and 10K (testing).
- ImageNet-100 (Tian et al., 2020a), a random 100-class subset of ImageNet.
- STL-10 (Coates et al., 2011), also derived from ImageNet, with  $96 \times 96$  resolution images and more than 100K training samples.

**Setting.** The goal of our experiments is to compare W-MSE with state-of-the-art SSL losses, isolating the effects of other settings, such as the architectural choices. For this reason, in the small and medium size dataset experiments of Tab. 1, we use the same encoder  $E(\cdot)$ , ResNet-18, for all the compared methods and, similarly, we use ResNet-50 for the ImageNet-based experiments in Tab. 3. When we do not report previously published results, we independently select the best hyperparameter values for each method and each dataset. In each method, the latent-space features are  $L_2$  normalized, unless otherwise specified. In Tab. 1, *SimCLR (our repro.)* refers to our implementation of the contrastive loss (Eq. 2) following the details in (Chen et al., 2020a), with temperature  $\tau = 0.5$ . In the same table, *BYOL (our repro.)* is our reproduction of (Grill et al., 2020). For this method we use the exponential moving average with cosine increasing, starting from 0.99. *W-MSE 2* and *W-MSE 4* correspond to our method with  $d = 2$  and  $d = 4$  positives extracted per image, respectively. For CIFAR-10 and CIFAR-100, the slicing sub-batch size is 128, for Tiny ImageNet and STL-10, it is 256. In the Tiny ImageNet and STL-10 experiments with W-MSE 2, we use 4 iterations of batch slicing, while in all the other experiments we use only 1 iteration.

**Implementation details.** For the small and medium size datasets, we use the Adam optimizer (Kingma & Ba, 2014). For all the compared methods (including ours), we use the same number of epochs and the same learning rate schedule. Specifically, for CIFAR-10 and CIFAR-100, we use 1,000 epochs with learning rate  $3 \times 10^{-3}$ ; for Tiny ImageNet, 1,000 epochs with learning rate  $2 \times 10^{-3}$ ; for STL-10, 2,000 epochs with learning rate  $2 \times 10^{-3}$ . We use learning rate warm-up for the first 500 iterations of the optimizer, and a 0.2 learning rate drop 50 and 25 epochs before the end. We use a mini-batch size of  $K = 1024$  samples. The dimension of the hidden layer of the projection head  $g(\cdot)$  is 1024. The

weight decay is  $10^{-6}$ . Finally, we use an embedding size of 64 for CIFAR-10 and CIFAR-100, and an embedding of size of 128 for STL-10 and Tiny ImageNet. For ImageNet-100 we use a configuration similar to the Tiny ImageNet experiments, and 240 epochs of training. Finally, in the ImageNet experiments (Tab. 3), we use the implementation and the hyperparameter configuration of (Chen et al., 2020b) (same number of layers in the projection head, etc.) based on their open-source implementation<sup>2</sup>, the only difference being the learning rate and the loss function (respectively, 0.075 and the contrastive loss in (Chen et al., 2020b) vs. 0.1 and Eq. 6 in W-MSE 4).

**Image Transformation Details.** We extract crops with a random size from 0.2 to 1.0 of the original area and a random aspect ratio from  $3/4$  to  $4/3$  of the original aspect ratio, which is a commonly used data-augmentation technique (Chen et al., 2020a). We also apply horizontal mirroring with probability 0.5. Finally, we apply color jittering with configuration (0.4, 0.4, 0.4, 0.1) with probability 0.8 and grayscaling with probability 0.1. For ImageNet and ImageNet-100, we follow the details in (Chen et al., 2020a): crop size from 0.08 to 1.0, stronger jittering (0.8, 0.8, 0.8, 0.2), grayscaling probability 0.2, and Gaussian blurring with 0.5 probability. In all the experiments, at testing time we use only one crop (standard protocol).

**Evaluation Protocol.** The most common evaluation protocol for unsupervised feature learning is based on *freezing* the network encoder ( $E(\cdot)$ , in our case) after unsupervised pre-training, and then train a supervised *linear classifier* on top of it. Specifically, the linear classifier is a fully-connected layer followed by softmax, which is plugged on top of  $E(\cdot)$  after removing the projection head  $g(\cdot)$ . In all the experiments, we train the linear classifier for 500 epochs using the Adam optimizer and the labeled training set of each specific dataset, without data augmentation. The learning rate is exponentially decayed from  $10^{-2}$  to  $10^{-6}$ . The weight decay is  $5 \times 10^{-6}$ . In our experiments, we also include the accuracy of a k-nearest neighbors classifier (k-nn,  $k = 5$ ). The advantage of using this classifier is that it does not require additional parameters and training, and it is deterministic.

#### 4.1. Comparison with the state of the art

Tab. 1 shows the results of the experiments on small and medium size datasets. For W-MSE, 4 samples are generally better than 2. The contrastive loss performs the worst in most cases. The W-MSE 4 accuracy is the best on CIFAR-10 and CIFAR-100, while BYOL leads on STL-10 and Tiny ImageNet, although the gap between the two methods is marginal. In the Appendix, we plot the linear classification accuracy during training for the STL-10 dataset. The

plot shows that W-MSE 4 and BYOL have a similar performance during most of the training. However, in the first 120 epochs, BYOL significantly underperforms W-MSE 4 (e.g., the accuracy after 20 epochs: W-MSE 4, 79.98%; BYOL, 73.24%), indicating that BYOL requires a “warmup” period. On the other hand, W-MSE performs well from the beginning. This property is useful in those domains which require a rapid adaptation of the encoder, e.g., due to the change of the data distribution in continual learning or in reinforcement learning.

Tab. 2 shows the results on a larger dataset (ImageNet-100). In that table, MoCo is the contrastive-loss based method proposed in (He et al., 2019), and  $\mathcal{L}_{\text{align}}$  and  $\mathcal{L}_{\text{uniform}}$  are the two losses proposed in (Wang & Isola, 2020) (Sec. 1-2). Note that, while W-MSE (2 and 4) in Tab. 2 refer to our method with a *ResNet-18* encoder, the other results are reported from (Wang & Isola, 2020), where a much larger-capacity network (i.e., a *ResNet-50*) is used as the encoder. *Despite this large difference in the encoder capacity*, both versions of W-MSE significantly outperform the other two compared methods in this dataset. Tab. 2 also shows that W-MSE 2, the version of our method without multi-cropping, is highly competitive, being its classification accuracy significantly higher than state-of-the-art methods.

Finally, in Tab. 3 we show the ImageNet results using 100 and 400 training epochs, and we compare W-MSE 4 with the results of other state-of-the-art approaches as reported in (Chen & He, 2020). Despite some configuration details are different (e.g., the depth of the projection head, etc.), in all cases the encoder is a ResNet-50. However, SwAV refers to the reproduction of (Caron et al., 2020) used in (Chen & He, 2020), where no multi-crop strategy is adopted (hence,  $d = 2$ ), and a multi-crop version of SwAV may likely obtain significantly larger values. Tab. 3 shows that W-MSE 4 is the state of the art with 100 epochs and it is very close to the 400-epochs state of the art. These results confirm that our method is highly competitive, considering that we have not intensively tuned our hyperparameters and that our network is much simpler than other approaches.

#### 4.2. Contrastive loss with whitening

In this section, we analyse the effect of the whitening transform in combination with the contrastive loss. Specifically, we use the contrastive loss (Eq. 2) on whitened features  $\mathbf{z} = \text{Whitening}(\mathbf{v})$  (Eq. 7). Tab. 4 shows the results on CIFAR-10. The first row refers to the standard contrastive loss without whitening. Note that the difference with respect to Tab. 1 is due to the use of only 200 training epochs. The second row refers to Eq. 2, where the features ( $\mathbf{z}$ ) are computed using Eq. 7 and then  $L_2$  normalized, while in the last two rows,  $\mathbf{z}$  is not normalized. If the features are whitened and then normalized, we observed an unstable

<sup>2</sup><https://github.com/google-research/simclr>

Table 1. Classification accuracy (top 1) of a linear classifier and a 5-nearest neighbors classifier for different loss functions and datasets with a ResNet-18 encoder.

| Method                                   | CIFAR-10     |              | CIFAR-100    |              | STL-10       |              | Tiny ImageNet |              |
|--|--------------|--------------|--------------|--------------|--------------|--------------|---------------|--------------|
|  | linear       | 5-nn         | linear       | 5-nn         | linear       | 5-nn         | linear        | 5-nn         |
| SimCLR (Chen et al., 2020a) (our repro.) | 91.80        | 88.42        | 66.83        | 56.56        | 90.51        | 85.68        | 48.84         | 32.86        |
| BYOL (Grill et al., 2020) (our repro.)   | 91.73        | 89.45        | 66.60        | <b>56.82</b> | <b>91.99</b> | <b>88.64</b> | <b>51.00</b>  | <b>36.24</b> |
| W-MSE 2 (ours)                           | 91.55        | 89.69        | 66.10        | 56.69        | 90.36        | 87.10        | 48.20         | 34.16        |
| W-MSE 4 (ours)                           | <b>91.99</b> | <b>89.87</b> | <b>67.64</b> | 56.45        | 91.75        | 88.59        | 49.22         | 35.44        |

Table 2. Classification accuracy on ImageNet-100. Top 1 and 5 correspond to the accuracy of a linear classifier. W-MSE (2 and 4) are based on a ResNet-18 encoder. <sup>†</sup> indicates that the results are based on a ResNet-50 encoder and the values are reported from (Wang & Isola, 2020).

| Method   | top 1        | top 5        | 5-nn         |
|--|--------------|--------------|--------------|
| MoCo (He et al., 2019) <sup>†</sup>  | 72.80        | 91.64        | -            |
| $\mathcal{L}_{\text{align}}$ and $\mathcal{L}_{\text{uniform}}$<br>(Wang & Isola, 2020) <sup>†</sup> | 74.60        | 92.74        | -            |
| W-MSE 2 (ours)   | 76.00        | 93.14        | 67.04        |
| W-MSE 4 (ours)   | <b>79.02</b> | <b>94.46</b> | <b>71.32</b> |

training, with divergence after a few epochs. The unnormalized version with whitening converged, but its accuracy is worse than the standard contrastive loss (both normalized and unnormalized).

These experiments show that the whitening transform alone does not improve the SSL performance, and, used jointly with negative contrasting, it may be harmful. Conversely, we use whitening in our W-MSE to avoid a collapsed representation (Eq. 4) when only positives are used (Eq. 3).

## 5. Conclusion

In this paper we proposed a new SSL loss, W-MSE, which is alternative to common loss functions used in the field. Differently from the triplet loss and the contrastive loss, both of which are based on comparing a pair of positive instances against other samples, W-MSE computes only the inter-positive distances, while using a whitening transform to avoid degenerate solutions. Our proposal is similar to recent SSL methods like BYOL (Grill et al., 2020) and SimSiam (Chen & He, 2020), which use only positives to reduce the dependence on large batch sizes (Chen et al., 2020a). However, differently from BYOL and SimSiam, which adopt asymmetric network architectures and training protocols, our solution to avoid collapsed representations is much sim-

Table 3. Classification accuracy (top 1) of a linear classifier on ImageNet with a ResNet-50 encoder. All results but ours are reported from (Chen & He, 2020). <sup>†</sup> The reproduction of SwAV in (Chen & He, 2020) does not include a multi-crop strategy.

| Method                                 | 100 epochs   | 400 epochs  |
|--|--------------|-------------|
| SimCLR (Chen et al., 2020a)            | 66.5         | 69.8        |
| MoCo v2 (Chen et al., 2020c)           | 67.4         | 71.0        |
| BYOL (Grill et al., 2020)              | 66.5         | <b>73.2</b> |
| SwAV (Caron et al., 2020) <sup>†</sup> | 66.5         | 70.7        |
| SimSiam (Chen & He, 2020)              | 68.1         | 70.8        |
| W-MSE 4 (ours)                         | <b>69.43</b> | 72.56       |

Table 4. CIFAR-10: accuracy of the contrastive loss with whitened features, trained for 200 epochs.

| Whitened features | $L_2$ normalized | linear   | 5-nn  |
|-------------------|------------------|----------|-------|
| $\times$          | $\checkmark$     | 89.66    | 86.55 |
| $\checkmark$      | $\checkmark$     | diverged |       |
| $\times$          | $\times$         | 79.48    | 76.60 |
| $\checkmark$      | $\times$         | 77.39    | 74.14 |

pler, while achieving a classification accuracy which is, in most cases, comparable or superior to the state-of-the-art methods. Despite asymmetry in learning and whitening are alternative solutions, a combination is possible. This may be a direction of future investigation.

## Acknowledgements

This work was supported by EU H2020 project AI4Media No. 951911 and by the EUREGIO project OLIVER.



## References

- Bachman, P., Hjelm, R. D., and Buchwalter, W. Learning representations by maximizing mutual information across views. In *NeurIPS*, 2019.
- Bautista, M. A., Sanakoyeu, A., Tikhoncheva, E., and Ommer, B. CliqueCNN: deep unsupervised exemplar learning. In *NeurIPS*, 2016.
- Caron, M., Bojanowski, P., Joulin, A., and Douze, M. Deep clustering for unsupervised learning of visual features. In *ECCV*, 2018.
- Caron, M., Misra, I., Mairal, J., Goyal, P., Bojanowski, P., and Joulin, A. Unsupervised learning of visual features by contrasting cluster assignments. In *NeurIPS*, 2020.
- Chen, T., Kornblith, S., Norouzi, M., and Hinton, G. A simple framework for contrastive learning of visual representations. *arXiv:2002.05709*, 2020a.
- Chen, T., Kornblith, S., Swersky, K., Norouzi, M., and Hinton, G. Big self-supervised models are strong semi-supervised learners. *arXiv:2006.10029*, 2020b.
- Chen, X. and He, K. Exploring simple siamese representation learning. *arXiv:2011.10566*, 2020.
- Chen, X., Fan, H., Girshick, R. B., and He, K. Improved baselines with momentum contrastive learning. *arXiv:2003.04297*, 2020c.
- Coates, A., Ng, A. Y., and Lee, H. An analysis of single-layer networks in unsupervised feature learning. In *AISTATS*, 2011.
- Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*, 2009.
- Dereniowski, D. and Marek, K. Cholesky factorization of matrices in parallel and ranking of graphs. In *5th Int. Conference on Parallel Processing and Applied Mathematics*, 2004.
- Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. BERT: Pre-training of deep bidirectional transformers for language understanding. In *NAACL*, 2019.
- Donahue, J. and Simonyan, K. Large scale adversarial representation learning. In *NeurIPS*, 2019.
- Donahue, J., Krähenbühl, P., and Darrell, T. Adversarial feature learning. In *ICLR*, 2017.
- Dwibedi, D., Aytar, Y., Tompson, J., Sermanet, P., and Zisserman, A. Temporal cycle-consistency learning. In *CVPR*, 2019.
- Efron, B. *The jackknife, the bootstrap, and other resampling plans*, volume 38. Siam, 1982.
- Fetterman, A. and Albrecht, J. Understanding self-supervised and contrastive learning with bootstrap your own latent (BYOL). <https://untitled-ai.github.io/understanding-self-supervised-contrastive-learning.html>, 2020.
- Grill, J.-B., Strub, F., Altché, F., Tallec, C., Richemond, P. H., Buchatskaya, E., Doersch, C., Pires, B. A., Guo, Z. D., Azar, M. G., Piot, B., Kavukcuoglu, K., Munos, R., and Valko, M. Bootstrap your own latent: A new approach to self-supervised learning. *arXiv:2006.07733*, 2020.
- Gutmann, M. and Hyvärinen, A. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, 2010.
- Hadsell, R., Chopra, S., and LeCun, Y. Dimensionality reduction by learning an invariant mapping. In *CVPR*, 2006.
- He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *CVPR*, pp. 770–778, 2016.
- He, K., Fan, H., Wu, Y., Xie, S., and Girshick, R. Momentum contrast for unsupervised visual representation learning. *arXiv:1911.05722*, 2019.
- Hénaff, O. J., Razavi, A., Doersch, C., Eslami, S. M. A., and van den Oord, A. Data-efficient image recognition with contrastive predictive coding. *arXiv:1905.09272*, 2019.
- Hermans, A., Beyer, L., and Leibe, B. In defense of the triplet loss for person re-identification. *arXiv:1703.07737*, 2017.
- Hjelm, R. D., Fedorov, A., Lavoie-Marchildon, S., Grewal, K., Bachman, P., Trischler, A., and Bengio, Y. Learning deep representations by mutual information estimation and maximization. In *ICLR*, 2019.
- Huang, L., Yang, D., Lang, B., and Deng, J. Decorrelated batch normalization. In *CVPR*, 2018.
- Ioffe, S. and Szegedy, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, 2015.
- Ji, X., Henriques, J. F., and Vedaldi, A. Invariant information clustering for unsupervised image classification and segmentation. In *ICCV*, 2019.

- Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *arXiv:1412.6980*, 2014.
- Krizhevsky, A. and Hinton, G. Learning multiple layers of features from tiny images. *Technical Report*, 2009.
- Le, Y. and Yang, X. Tiny imagenet visual recognition challenge. 2015.
- Mikolov, T., Chen, K., Corrado, G., and Dean, J. Efficient estimation of word representations in vector space. *arXiv:1301.3781*, 2013a.
- Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. Distributed representations of words and phrases and their compositionality. In *NIPS*, 2013b.
- Misra, I. and van der Maaten, L. Self-supervised learning of pretext-invariant representations. *arXiv:1912.01991*, 2019.
- Misra, I., Zitnick, C. L., and Hebert, M. Shuffle and learn: Unsupervised learning using temporal order verification. In *ECCV*, 2016.
- Noroozi, M. and Favaro, P. Unsupervised learning of visual representations by solving jigsaw puzzles. In *ECCV*, 2016.
- Pathak, D., Krähenbühl, P., Donahue, J., Darrell, T., and Efros, A. A. Context encoders: Feature learning by inpainting. *CVPR*, 2016.
- Richemond, P. H., Grill, J.-B., Altché, F., Tallec, C., Strub, F., Brock, A., Smith, S., De, S., Pascanu, R., Piot, B., and Valko, M. Byol works even without batch statistics. *arXiv:2010.10241*, 2020.
- Schroff, F., Kalenichenko, D., and Philbin, J. FaceNet: A unified embedding for face recognition and clustering. In *CVPR*, 2015.
- Siarohin, A., Sangineto, E., and Sebe, N. Whitening and coloring transform for GANs. In *International Conference on Learning Representations*, 2019.
- Sohn, K. Improved deep metric learning with multi-class n-pair loss objective. In *NIPS*, 2016.
- Tian, Y., Krishnan, D., and Isola, P. Contrastive multiview coding. In *ECCV*, 2020a.
- Tian, Y., Yu, L., Chen, X., and Ganguli, S. Understanding self-supervised learning with dual deep networks. *arXiv:2010.00578*, 2020b.
- Tschannen, M., Djolonga, J., Rubenstein, P. K., Gelly, S., and Lucic, M. On mutual information maximization for representation learning. *arXiv:1907.13625*, 2019.
- van den Oord, A., Li, Y., and Vinyals, O. Representation learning with contrastive predictive coding. *arXiv:1807.03748*, 2018.
- Vincent, P., Larochelle, H., Bengio, Y., and Manzagol, P. Extracting and composing robust features with denoising autoencoders. In *ICML*, 2008.
- Wang, T. and Isola, P. Understanding contrastive representation learning through alignment and uniformity on the hypersphere. In *International Conference on Machine Learning*, 2020.
- Wang, X. and Gupta, A. Unsupervised learning of visual representations using videos. In *ICCV*, 2015.
- Wu, Z., Xiong, Y., Yu, S., and Lin, D. Unsupervised feature learning via non-parametric instance-level discrimination. *arXiv:1805.01978*, 2018.
- Xu, X., Chen, H., Moreno-Noguer, F., Jeni, L. A., and la Torre, F. D. 3D human shape and pose from a single low-resolution image with self-supervised learning. In *ECCV*, 2020.
- You, Y., Gitman, I., and Ginsburg, B. Large batch training of convolutional networks. *arXiv:1708.03888*, 2017.
- Zhuang, C., Zhai, A. L., and Yamins, D. Local aggregation for unsupervised learning of visual embeddings. *arXiv:1903.12355*, 2019.

## A. Training Dynamics

Fig. 4 and 5 show the training dynamics for methods compared in Tab. 1. All the curves are smoothed with a 0.3 moving average for a better readability (curves before smoothing are shown semi-transparent).

## B. Cholesky Whitening and Backproagation

We compute  $W_V$  (Eq. 8) following (Siarohin et al., 2019) and using the Cholesky decomposition. The Cholesky decomposition is based on the factorization of the covariance symmetric matrix using two triangular matrices:  $\Sigma_V = LL^\top$ , where  $L$  is a lower triangular matrix. Once we  $L$  is computed, we compute the inverse of  $L$ , and we get:  $W_V = L^{-1}$ . Note that the Cholesky decomposition is fully differentiable and it is implemented in all of the major frameworks, such as PyTorch and TensorFlow. However, for the sake of completeness, we provide below the gradient computation.

### B.1. Gradient Computation

We provide here the equations for whitening differentiation as reported in (Siarohin et al., 2019). Let  $Z$  be the whitened version of the batch  $V$ , i.e.,  $Z = W_V(V - \mu_V)$ . The gradient  $\frac{\partial L}{\partial V}$  can be computed by:

$$\frac{\partial L}{\partial V} = \frac{2}{K-1} \frac{\partial L}{\partial \Sigma} V + W_V^T \frac{\partial L}{\partial Z}. \quad (9)$$

where the partial derivative  $\frac{\partial L}{\partial Z}$  is backpropogated, while  $\frac{\partial L}{\partial \Sigma}$  is computed as follows:

$$\frac{\partial L}{\partial \Sigma} = -\frac{1}{2} W_V^T \left( P \circ \frac{\partial L}{\partial W_V} W_V^T + \left( P \circ \frac{\partial L}{\partial W_V} W_V^T \right)^T \right) W_V \quad (10)$$

In (10),  $\circ$  is Hadamard product, while  $\frac{\partial L}{\partial W_V}$  is:

$$\frac{\partial L}{\partial W_V} = \frac{\partial L}{\partial Z} V^T, \quad (11)$$

and  $P$  is:

$$P = \begin{pmatrix} \frac{1}{2} & 0 & \dots & 0 \\ 1 & \frac{1}{2} & \ddots & 0 \\ 1 & \ddots & \ddots & 0 \\ 1 & \dots & 1 & \frac{1}{2} \end{pmatrix}.$$

## C. Training time complexity

Following (Siarohin et al., 2019), the complexity of the whitening transform is  $O(k^3 + Mk^2)$ , where  $k$  is the embedding dimension and  $M$  is the size of the sub-batch used in the batch slicing process. Since  $k < M$  (see Sec. 3),

the whitening transform is  $O(Mk^2)$ , which is basically equivalent to the forward pass of  $M$  activations in a fully-connected layer connecting two layers of  $k$  neurons each. In fact, the training time is dominated by other architectural choices which are usually more computationally demanding than the loss computation. For instance, BYOL (Grill et al., 2020) needs 4 forward passes through 2 networks for each pair of positives. Hence, to evaluate the wall-clock time, we measure the time spent for one mini-batch iteration by all the methods compared in Tab. 1. We use the STL-10 dataset, a ResNet-18 encoder and a server with one Nvidia Titan Xp GPU. Time of one iteration: Contrastive, 459ms; BYOL, 602ms; W-MSE 2, 478ms; W-MSE 4, 493ms. The 19ms difference between Contrastive and W-MSE 2 is due to the whitening transform. Since the factual time is mostly related to the sample forward and backward passes, the  $d(d-1)$  positive comparisons in Eq. 6, do not significantly increase the wall-clock time of W-MSE 4 with respect to W-MSE 2.

## D. Euclidean distance

Table 5. Classification accuracy (top 1) using the Euclidean distance (unnormalized embeddings) on STL-10.

| Method              | linear | 5-nn  |
|---------------------|--------|-------|
| SimCLR (our repro.) | 78.00  | 71.07 |
| BYOL (our repro.)   | 80.83  | 74.94 |
| W-MSE 2             | 89.91  | 85.56 |
| W-MSE 4             | 90.40  | 87.09 |

The cosine similarity is a crucial component in most of the current self-supervised learning approaches. This is usually implemented with an  $L_2$  normalization of the latent-space representations, which corresponds to projecting the features on the surface of the unit hypersphere. However, in our W-MSE, the whitening transform projects the representation onto a spherical distribution (intuitively, we can say on the whole unit hypersphere). Preserving the module of the features before the  $L_2$  normalization may be useful in some applications, e.g., clustering the features after the projection head using a Gaussian mixture model. Tab. 5 shows an experiment on the STL-10 dataset where we use unnormalized embeddings for all the methods (and  $\tau = 1$  for the contrastive loss). Comparing Tab. 5 with Tab. 1, the accuracy decrease of W-MSE is significantly smaller than in the other methods.

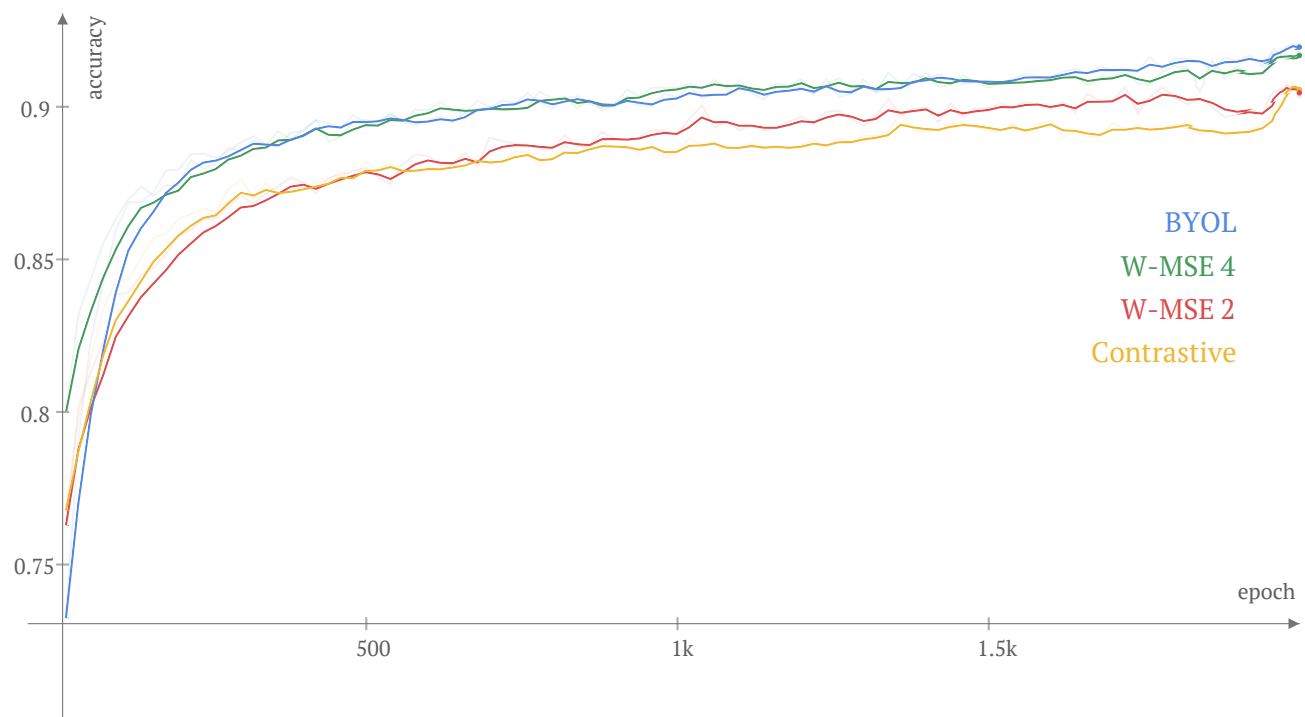


Figure 4. Training dynamics on the STL-10 dataset (linear-classifier based evaluation).

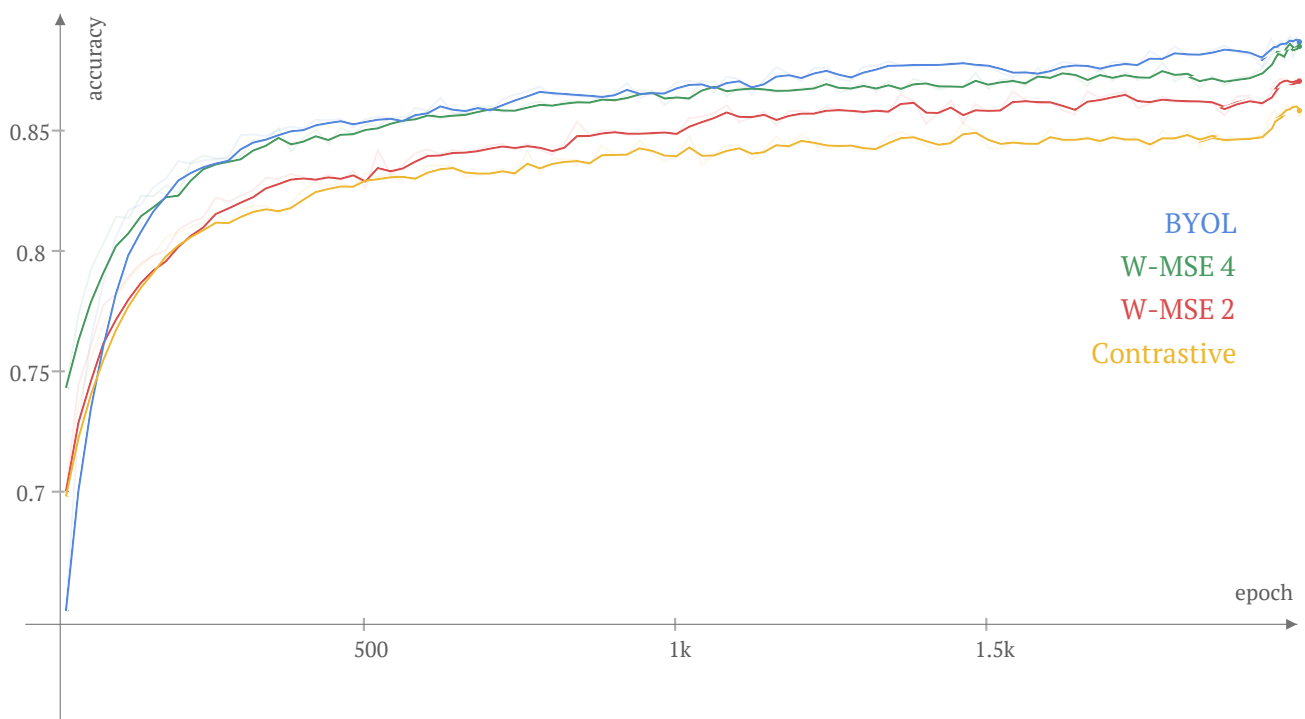


Figure 5. Training dynamics on the STL-10 dataset (5-nn classifier based evaluation).