# Self-Supervised Learning with an Information Maximization Criterion

**Serdar Ozsoy**[1,2] **Shadi Hamdan**[1,3] **Sercan Ö. Arik**[4] **Deniz Yuret**[1,3] **Alper T. Erdogan**[1,2]
[1]KUIS AI Center, Koc University, Turkey [2]EEE Department, Koc University, Turkey
[3]CE Department, Koc University, Turkey [4]Google Cloud AI Research, Sunnyvale, CA
{sozsoy19, shamdan17, dyuret, alperdogan}@ku.edu.tr soarik@google.com

## Abstract

Self-supervised learning allows AI systems to learn effective representations from large amounts of data using tasks that do not require costly labeling. Mode collapse, i.e., the model producing identical representations for all inputs, is a central problem to many self-supervised learning approaches, making self-supervised tasks, such as matching distorted variants of the inputs, ineffective. In this article, we argue that a straightforward application of information maximization among alternative latent representations of the same input naturally solves the collapse problem and achieves competitive empirical results. We propose a self-supervised learning method, CorInfoMax, that uses a second-order statistics-based mutual information measure that reflects the level of correlation among its arguments. Maximizing this correlative information measure between alternative representations of the same input serves two purposes: (1) it avoids the collapse problem by generating feature vectors with non-degenerate covariances; (2) it establishes relevance among alternative representations by increasing the linear dependence among them. An approximation of the proposed information maximization objective simplifies to a Euclidean distance-based objective function regularized by the log-determinant of the feature covariance matrix. The regularization term acts as a natural barrier against feature space degeneracy. Consequently, beyond avoiding complete output collapse to a single point, the proposed approach also prevents dimensional collapse by encouraging the spread of information across the whole feature space. Numerical experiments demonstrate that CorInfoMax achieves better or competitive performance results relative to the state-of-the-art SSL approaches.

## 1 Introduction

Self-supervised learning (SSL) is an important paradigm with significant impact in artificial intelligence [1–6]. In particular, SSL can significantly reduce the need for expensively-obtained labeled data. Furthermore, the internal representations learned by SSL approaches are more suitable for task generalization in transfer learning applications [7]. Recently, SSL methods have even been demonstrated to achieve comparable performance to their supervised counterparts [8, 9].

The primary approach for SSL is to define a pretext objective based on unlabeled data, whose optimization leads to rich and useful representations for downstream applications. For example, for visual processing, a common way to construct a pretext task is to obtain similar features for two different augmentations of the same input, which exploits the expected invariance of representations to particular transformations of the input. However, the corresponding optimization setting should avoid the (total) "collapse problem," which is described as generating the same feature output for all inputs. Figure 1.(a) illustrates a toy view of the total collapse problem, where all latent vector outputs converge to a single point during training. We can also define its less severe version, which

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

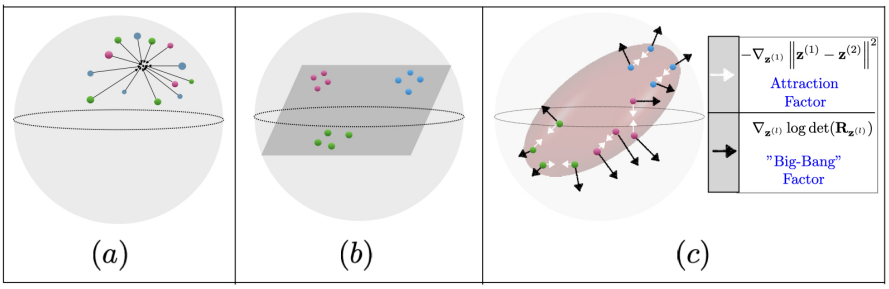


Figure 1: Depictions in latent space - (a) "Total Output Collapse": all latent vectors converge to the same point, (b) "Dimensional Output Collapse": latent vectors are restricted to a strict subspace of the latent space, (c) Gradient dynamics of the CorInfoMax based on (6): the ellipsoid surface reflects the average spread pattern of latent vectors.

is referred to as "dimensional collapse," where all latent vectors lie in a strict subspace of the whole latent space as illustrated by Figure 1.(b).

This article proposes an SSL approach based on the maximization of a form of mutual information among the alternative latent representations obtained from the same input. The mutual information maximization approach would serve two fundamental purposes for SSL:

(i). (*Similarity*) It would enforce the representations obtained for the same input source to be related to or dependent on each other,

(ii). (*No Collapse*) Since the entropy of each alternative representation is lower bounded by the mutual information, maximizing mutual information ensures that latent representations have non-degenerate distributions avoiding collapse.

Therefore, the information maximization approach provides a natural solution to the problems targeted by the existing SSL approaches, which are briefly surveyed in Sec. 2.2.

The conventional choice for mutual information measure is Shannon Mutual Information (SMI) [10]. For SSL, the fundamental limitations on the precise computation of SMI [11–13] impose serious challenges on model training. Especially when mutual information is large, we require drastically more samples (exponentially related to SMI) to obtain a reliable estimate of SMI [13]. However, the use of large batch sizes to improve precision may not be a favorable solution, as (i) it may have an adverse impact on generalization performance [14, 15], (ii) it can significantly increase memory requirements. Furthermore, estimating and optimizing SMI can significantly increase the computational burden. Finally, SMI maximization may induce a nonlinear dependence between the representations corresponding to the same input, where the resultant partitioning of the latent space may not be suitable as an input to a simple linear classifier.

To address these issues, we can consider a second-order-statistics-based mutual information measure, referred to as log-determinant mutual information (LDMI) [16, 17]. The joint entropy measure corresponding to LDMI is the log-determinant of the diagonally perturbed covariance matrix. The zero perturbation case boils down to the Shannon differential entropy for Gaussian distributed vectors. Unlike SMI, which reflects general dependence between its arguments, LDMI reflects linear dependence and is computationally more efficient.

In this article, we propose a novel SSL approach, correlative information maximization (CorInfoMax), derived from the LDMI measure. The loss function for CorInfoMax is obtained through the first-order approximation of LDMI and restricting the linear dependence to the identity map. The resulting optimization objective is a Euclidean distance-based loss function regularized by the log-determinant of the latent vector covariance matrix. This regularization term encourages the spreading of the feature vectors and acts as a natural barrier against the dimensional degeneracy and collapse in the feature space. Our numerical experiments confirm that CorInfoMax learns effective representations that perform well in downstream tasks.

The main contributions can be summarized as follows:

- Based on information theoretical grounds, we propose a novel framework, CorInfoMax, with an interpretable loss function that has explicit components for minimizing variance of positive samples ('the attraction factor') and for making full use of the embedding space to avoid dimensional collapse ('the big-bang factor').
- We introduce a computationally-efficient approximation to LDMI that simplifies it by using an identity mapping instead of a general linear mapping. This approximation directly minimizes representation variance, which is desired for self-supervised learning.
- CorInfoMax relies only on second-order statistics, does not require negative samples, and does not force embeddings to be uncorrelated.
- In addition to these theoretically appealing features, the proposed CorInfoMax framework achieves either better or similar performance results compared to the state-of-the-art SSL approaches.

The following is the organization of the article: Section 2 provides a discussion of the relevant literature related to the proposed approach. We introduce the log-determinant entropy and mutual information measures by highlighting LDMI as a measure of linear dependence in Section 3. In the same section, we derive the objective function of CorInfoMax as a variation on the LDMI measure. Section 4 introduces the proposed correlative information maximization-based self-supervised learning method. Section 5 provides the numerical experiments illustrating the performance of the proposed approach. The appendix provided in the supplementary document contains the details related to these experiments. Finally, Section 6 presents the discussion and conclusions.

## 2 Related work

### 2.1 Determinant maximization for unsupervised learning

The determinant maximization criterion utilized in our framework has been used as an effective algorithmic tool in unsupervised matrix factorization methods such as nonnegative matrix factorization (NMF) [18, 19], simplex-structured matrix factorization (SSMF) [20], sparse component analysis (SCA) [21], bounded component analysis (BCA) [22] and polytopic matrix factorization (PMF) [23].

In the generative models of these frameworks, the input data is assumed to be linear transformations of some latent vectors. Furthermore, these latent vectors are assumed to be sufficiently scattered in their domain. Maximizing the determinant of the latent covariance matrix spreads the latent vector estimates to capture this presumed scattering of the generative model samples. Similarly, the log-determinant of the latent vector covariance matrix in the CorInfoMax objective causes the spreading of latent vectors in their ambient space and, therefore, avoids collapse. According to [17], the determinant maximization criterion used in these matrix factorization frameworks is essentially equivalent to correlative information maximization between input and its latent factors based on the LDMI objective, where the latent vectors are constrained to lie in their domain determined by the generative model.

In the CorInfoMax approach, we utilize determinant maximization for correlative information maximization among latent space vectors rather than between inputs and their latent space representations.

### 2.2 Handling collapse for SSL

Collapse is a central concern for SSL with the same input in computer vision, and we can categorize the existing approaches according to how they handle this issue. For example, contrastive methods, such as SimCLR [8] and MoCo [24], define an objective that pushes features for different inputs (negative samples) away from each other while keeping the representations corresponding to the same input (positive samples) close to each other. The number and the choice of the negative samples appear as critical factors for the performance and scalability of contrastive methods. Another category of SSL is the distillation methods, such as BYOL [4] and SimSiam [25], which avoid collapse through the asymmetry of alternative encoder branches and algorithmic tuning [26]. Other SSL approaches such as DeepCluster [27], SeLa [28] and SvAW [9] enforce a cluster structure in the feature space to prevent constant output. As a different line of algorithms, decorrelated batch normalization (DBN) [29] Barlow Twins [5], Whitening MSE (W-MSE) [30] and VICReg [6], use feature decorrelation [31] as a means to avoid information collapse. The effective whitening mechanisms used in these methods aim for the isotropic spread of information inside the feature space which also prevents "dimensional collapse" illustrated in Figure 1.(b). The reference [32] constructs an SSL loss function using the Hilbert-Schmidt Independence Criterion, a kernel-based independence measure, to maximize the dependence

between alternative representations of the same input. Finally, [33] proposed graph representations for the embeddings of positive samples and the corresponding spectral decomposition algorithm.

The proposed CorInfoMax approach is most related to the decorrelation-based methods discussed above, mainly due to correlation-based measures. However, unlike the decorrelation methods, CorInfoMax does not constrain latent vectors to be uncorrelated. Instead, it avoids covariance matrix degeneracy by using its log-determinant as a regularizer loss function. Furthermore, the information maximization principle is more direct and explicit for the CorInfoMax algorithm.

### 2.3 Information maximization for unsupervised learning

The maximization of SMI has been proposed as an unsupervised learning mechanism in different but related contexts. As one of the earliest approaches, Linsker proposed maximum information transfer from input data to its latent representation and showed that it is equivalent to maximizing the determinant of the output covariance under the Gaussian distribution assumption [34]. Around the same time frame, Becker & Hinton [35] put forward a representation learning approach based on the maximization of (an approximation of) the SMI between the alternative latent vectors obtained from the same image. The most well-known application is the Independent Component Analysis (ICA) Infomax algorithm [36] for separating independent sources from their linear combinations. The ICA-Infomax algorithm targets to maximize the mutual information between mixtures and source estimates while imposing statistical independence among outputs. The Deep Infomax approach [37] extends this idea to unsupervised feature learning by maximizing the mutual information between the input and output while matching a prior distribution for the representations.

It is essential to underline that the proposed method clearly distinguishes itself from the Deep Infomax in [37]: Our objective is not to maximize the mutual information between inputs and outputs of a deep network. Instead, we maximize the mutual information content of the alternative latent representations of the same input. From this point of view, our approach is closer to what is aimed at by [35]. However, we use a different (correlative) information measure, which is computationally more efficient and induces a special form of linear dependence among alternative latent representations of the same input, which may be more desirable considering the goal of generating features for a linear classifier.

## 3 CorInfoMax as a criterion based on log-determinant mutual information

This section derives the optimization objective for the CorInfoMax framework as a variation on the LDMI measure. As described in Appendix A, the LDMI between random vectors $\mathbf{x}$ and $\mathbf{y}$ is given by

$$\begin{aligned} I_{LD}^{(\varepsilon)}(\mathbf{x};\mathbf{y}) \quad = \quad & \frac{1}{4}(\log\det(\mathbf{R}_{\mathbf{x}}+\varepsilon\mathbf{I})+\log\det(\mathbf{R}_{\mathbf{y}}+\varepsilon\mathbf{I}) \\ & -\log\det(\mathbf{R}_{\mathbf{x}}-\mathbf{R}_{\mathbf{xy}}(\mathbf{R}_{\mathbf{y}}+\varepsilon\mathbf{I})^{-1}\mathbf{R}_{\mathbf{xy}}^{T}+\varepsilon\mathbf{I}) \\ & -\log\det(\mathbf{R}_{\mathbf{y}}-\mathbf{R}_{\mathbf{xy}}^{T}(\mathbf{R}_{\mathbf{x}}+\varepsilon\mathbf{I})^{-1}\mathbf{R}_{\mathbf{xy}}+\varepsilon\mathbf{I})), \end{aligned} \tag{1}$$

where $\mathbf{R}_{\mathbf{x}}$ and $\mathbf{R}_{\mathbf{y}}$ are auto-covariance matrices for $\mathbf{x}$ and $\mathbf{y}$, respectively, and $\mathbf{R}_{\mathbf{xy}}$ is their cross-covariance matrix.

There are three potential advantages of using LDMI in (1) over SMI in (A.1) for SSL:

i. It is based only on second-order statistics. On the other hand, SMI is a statistic based on the joint PDFs of the argument vectors, and its accurate estimation has high sample complexity. In contrast, it is more practical to obtain (the estimates of) the auto-covariance and the cross-covariance matrices, as discussed in Section 4.
ii. There are benefits of information maximization principle in connection with self-supervised training. The maximization of SMI of two vectors induces a general, potentially nonlinear dependence between them, whereas the maximization of LDMI increases correlation, or linear dependence. Therefore, LDMI-based information maximization is expected to organize the feature space being more favorable for data-efficient and low-complexity linear or shallow supervised classifiers as targeted in SSL applications.
iii. The resulting LDMI-based objective functions are more interpretable – they involve the log-determinant of the projector-space covariance, whose maximization clearly avoids feature collapse, a significant concern in SSL methods.

The primary motivation for obtaining an LDMI variant is to increase the similarity between the alternative latent representations of the same input by enforcing the identity transformation between them.

Let $\mathbf{z}^{(1)}$ and $\mathbf{z}^{(2)}$ represent alternative latent representations corresponding to the same input. One potential SSL approach would be to pursue direct maximization of $I_{LD}^{(\varepsilon)}(\mathbf{z}^{(1)};\mathbf{z}^{(2)})$ in (1). As discussed earlier, this would maximize the correlation between $\mathbf{z}^{(1)}$ and $\mathbf{z}^{(2)}$, therefore inducing a linear dependence between them. For the SSL application, we would prefer the identity mapping to an arbitrary linear relationship such that the alternative latent representations corresponding to the same input concentrate in the same neighborhood of the latent space. Therefore, we modify the LDMI expression in (1) to impose this constraint. For this purpose, we apply the first-order Taylor series approximation, $\log\det(\mathbf{C}+\mathbf{D}) \approx \log\det(\mathbf{C})+\text{Tr}(\mathbf{D}^T\mathbf{C}^{-1})$, on the third term in the right side of (1), which provides

$$\begin{aligned}
&\log\det(\mathbf{R}_{\mathbf{z}^{(1)}} - \mathbf{R}_{\mathbf{z}^{(1)}\mathbf{z}^{(2)}}(\mathbf{R}_{\mathbf{z}^{(2)}}+\varepsilon\mathbf{I})^{-1}\mathbf{R}_{\mathbf{z}^{(1)}\mathbf{z}^{(2)}}^T + \varepsilon\mathbf{I}) \\
&\approx \frac{1}{\varepsilon}\text{Tr}(\mathbf{R}_{\mathbf{z}^{(1)}} - \mathbf{R}_{\mathbf{z}^{(1)}\mathbf{z}^{(2)}}\mathbf{R}_{\mathbf{z}^{(2)}}^{-1}\mathbf{R}_{\mathbf{z}^{(1)}\mathbf{z}^{(2)}}^T) + \log\det(\varepsilon\mathbf{I}) \\
&= \frac{1}{\varepsilon}\min_{\mathbf{A}_1,\mathbf{b}_1} E(\|\mathbf{z}^{(1)} - (\mathbf{A}_1\mathbf{z}^{(2)}+\mathbf{b}_1)\|_2^2) + P\log(\varepsilon). \qquad (2)
\end{aligned}$$

Therefore, the expression in (2) corresponds to the mean square error of the best linear (affine) MMSE estimator of $\mathbf{z}^{(1)}$ from $\mathbf{z}^{(2)}$, multiplied by $\varepsilon^{-1}$. Similarly, if we apply the same approximation to the fourth term on the right side of (1), we obtain

$$\begin{aligned}
&\log\det(\mathbf{R}_{\mathbf{z}^{(2)}} - \mathbf{R}_{\mathbf{z}^{(1)}\mathbf{z}^{(2)}}^T(\mathbf{R}_{\mathbf{z}^{(1)}}+\varepsilon\mathbf{I})^{-1}\mathbf{R}_{\mathbf{z}^{(1)}\mathbf{z}^{(2)}} + \varepsilon\mathbf{I}) \\
&\approx \frac{1}{\varepsilon}\min_{\mathbf{A}_2,\mathbf{b}_2} E(\|\mathbf{z}^{(2)} - (\mathbf{A}_2\mathbf{z}^{(1)}+\mathbf{b}_2)\|_2^2) + P\log(\varepsilon). \qquad (3)
\end{aligned}$$

In order to induce the identity mapping between $\mathbf{z}^{(1)}$ and $\mathbf{z}^{(2)}$, we constrain $\mathbf{A}_1 = \mathbf{A}_2 = \mathbf{I}$, and $\mathbf{b}_1 = \mathbf{b}_2 = \mathbf{0}$, which transforms both (2) and (3) into $\varepsilon^{-1}E(\|\mathbf{z}^{(1)} - \mathbf{z}^{(2)}\|_2^2) + \text{const}$. Therefore, scaling the expression in (1) by 4 and using this modification, we obtain

$$J(\mathbf{z}^{(1)},\mathbf{z}^{(2)}) = \log\det(\mathbf{R}_{\mathbf{z}^{(1)}}+\varepsilon\mathbf{I}) + \log\det(\mathbf{R}_{\mathbf{z}^{(2)}}+\varepsilon\mathbf{I}) - 2\varepsilon^{-1}E(\|\mathbf{z}^{(1)} - \mathbf{z}^{(2)}\|_2^2), \qquad (4)$$

where we ignored the constant terms. We refer to (4) as the stochastic CorInfoMax objective function. In Section 4, we propose a SSL method based on the optimization of this objective function.

# 4 LDMI based self-supervised learning

This section introduces the proposed correlative information maximization (CorInfoMax) approach for SSL. We start by describing the presumed setting for the pretext task, which is matching the latent representations of different augmentations of the same input in Section 4.1. Section 4.2 is the main section where we propose the correlative information maximization algorithm for SSL. Finally, we discuss the implementation complexity of the proposed approach in Section 4.3.

## 4.1 Self-supervised learning setting

We start by describing the presumed self-supervised learning setup, which is illustrated in Figure 2:

- *The input* is a sequence of tensors $\{\mathcal{X}[l] \in \mathbb{R}^{M\times K\times N}, l \in \mathbb{Z}\}$, where each index sample corresponds to a batch of $N$ images with dimensions $M \times K$. As discussed in Section 1, the pretext task that we consider is matching the latent representations of the two augmentations of the same input. Therefore, we first apply *the augmentation functions* $a_1(\cdot)$ and $a_2(\cdot)$ to obtain the transformed versions of the input sequence, i.e., $\{\mathcal{X}^{(q)}[l] = a_q(\mathcal{X}[l]) \in \mathbb{R}^{M\times K\times N}, l \in \mathbb{Z}\}, \quad q \in \{1,2\}$. We will illustrate particular choices of augmentations in the examples provided in Section 5. We note that the proposed setting can be potentially extended to multi-modal schemes where the augmentations are defined for multiple modalities of the same input, such as visual and sound.
- We represent *Siamese networks* to be trained by SSL with the mappings $f_q(\mathcal{U};\mathbb{W}_{DNN}^{(q)})$ for $q = 1, 2$, where $\mathcal{U} \in \mathbb{R}^{M\times K\times N}$ is the input, $\mathbb{W}_{DNN}^{(q)}$ represents the trainable parameters of the $q^{\text{th}}$ network.

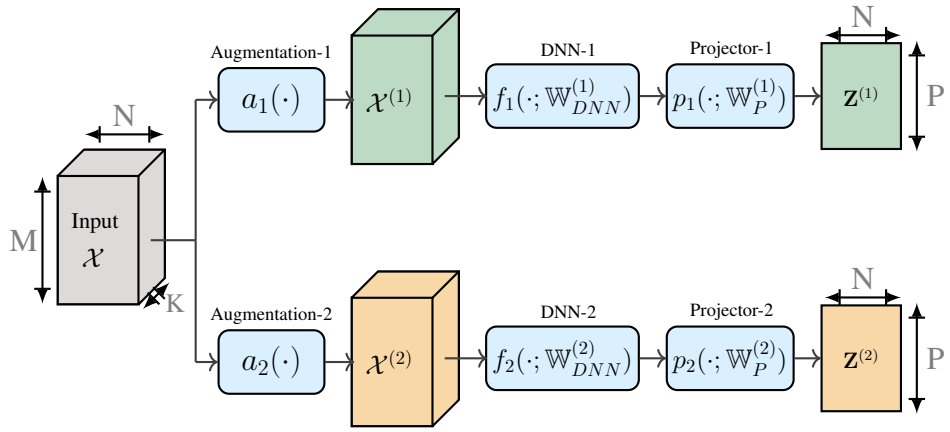


Figure 2: SSL setup: we consider two parallel encoder branches corresponding to two different augmentations of the same input $X$. Augmented views are fed into Siamese networks $f$ followed by projector $p$, basically a 3-layer MLP. $N$ stands for batch size, $M \times K$ for image dimension, $P$ for feature dimension of projector output.

- The outputs of the DNN for different augmented inputs are represented by $\mathbf{Y}^{(q)}[l] = f_q(\mathbb{X}^{(q)}[l];\mathbb{W}^{(q)}_{DNN}) \in \mathbb{R}^{F\times N}, \quad q \in \{1,2\}, l \in \mathbb{Z}$, where $F$ is the output feature dimension.
- To be used in self-supervised training, we project the outputs of the enconders using *projector networks* $p_q(\mathbf{Y};\mathbb{W}^{(q)}_P)$, $q = 1,2$, where $\mathbf{Y}$ is its input, and $\mathbb{W}^{(q)}_P$ is the set of trainable parameters of the $q^{\text{th}}$ projector.
- The outputs of the projector network for different augmentations are represented by $\mathbf{Z}^{(q)}[l] = p_q(\mathbf{Y}^{(q)}[l];\mathbb{W}^{(q)}_P) \in \mathbb{R}^{P\times N}, \quad q \in \{1,2\}, l \in \mathbb{Z}$, where $P$ is the projector head dimension.

For the numerical experiments in the article, we consider the weight-sharing setup where both encoders use the same network weights. We provide more details in Appendix B.

### 4.2 Correlative mutual information maximization

We define the CorInfoMax approach for SSL through the optimization problem

$$\underset{\mathbb{W}_{DNN},\,\mathbb{W}_P}{\text{maximize}} \quad \hat{J}(\mathbf{z}^{(1)};\mathbf{z}^{(2)})[l] \tag{5a}$$

where $\hat{J}^{(\varepsilon)}$ is the sample based estimate of (4) at the $l^{\text{th}}$-batch which can be written as

$$\hat{J}(\mathbf{z}^{(1)},\mathbf{z}^{(2)})[l] = \log\det(\hat{\mathbf{R}}_{\mathbf{z}^{(1)}}[l]+\varepsilon\mathbf{I}) + \log\det(\hat{\mathbf{R}}_{\mathbf{z}^{(2)}}[l]+\varepsilon\mathbf{I}) - \frac{2}{\varepsilon N}\|\mathbf{Z}^{(1)}[l]-\mathbf{Z}^{(2)}[l]\|_F^2, \tag{6}$$

where the rightmost term stands for the sample based estimate of the term $2\varepsilon^{-1}E(\|\mathbf{z}^{(1)}-\mathbf{z}^{(2)}\|_2^2)$ in (4), and $\hat{\mathbf{R}}_{\mathbf{z}^{(1)}}[l]$ and $\hat{\mathbf{R}}_{\mathbf{z}^{(2)}}[l]$ are the auto-covariance matrix estimates for the projector heads at the $l^{\text{th}}$ batch. If the batch size is large enough, these estimates can only be obtained from the current batch samples. However, due to hardware limitations and considering the fact that intermediate batch sizes offer better accuracy [14, 15], sufficiently-large batch sizes may not be possible to obtain reliable covariance estimates. Therefore, we adopt recursive covariance matrix estimation across batches [38]. The corresponding covariance update expressions take the form

$$\hat{\mathbf{R}}_{\mathbf{z}^{(q)}}[l] \quad = \quad \lambda\hat{\mathbf{R}}_{\mathbf{z}^{(q)}}[l-1] + (1-\lambda)\frac{1}{N}\tilde{\mathbf{Z}}^{(q)}[l]\tilde{\mathbf{Z}}^{(q)}[l]^T. q \in \{1,2\}, \tag{7}$$

where $0 \leq \lambda < 1$ is the forgetting factor, and $\tilde{\mathbf{Z}}^{(q)}[l]$ represents the batch of mean-centralized projector outputs defined as $\tilde{\mathbf{Z}}^{(q)}[l] = \mathbf{Z}^{(q)}[l] - \boldsymbol{\mu}^{(q)}[l]\mathbf{1}_N^T$, where $\{\boldsymbol{\mu}^{(q)}[l], q \in \{1,2\}\}$ represents the mean estimates for the projector outputs, updated by $\boldsymbol{\mu}^{(q)}[l] = \lambda\boldsymbol{\mu}^{(q)}[l-1] + (1-\lambda)\frac{1}{N}\mathbf{Z}^{(q)}\mathbf{1}_N$, for $q \in \{1,2\}$. (See Appendix C for CorInfoMax pseudocode.)

It is informative to inspect the terms in the objective function $\hat{J}(\mathbf{z}^{(1)}, \mathbf{z}^{(2)})$ in (6):

- Minimization of $\frac{2\varepsilon^{-1}}{N}\|\mathbf{Z}^{(1)}[l] - \mathbf{Z}^{(2)}[l]\|_F^2$, acts as a force to pull the representations of alternative augmentations toward each other, which we refer to as the *attraction factor*.
- Maximization of $\log\det(\hat{\mathbf{R}}_{\mathbf{z}^{(1)}}[l] + \varepsilon\mathbf{I})$ acts as a dispersion force causing non-degenerate (for $\varepsilon \approx 0$) expansion of projection vectors in the $P$-dimensional space, which we informally refer to as *big-bang factor*. Therefore, the covariance determinant acts as a regularization or barrier function avoiding collapse in the latent space. Consequently, it provides a convincing replacement for negative samples in contrastive methods to prevent feature collapse. Appendix K illustrates the eigenvalues of the projector covariance vector for the CIFAR-10 training, which demonstrates that the CorInfoMax criterion spreads information throughout the feature space, avoiding dimensional collapse.

Figure 1.(c) illustrates the learning dynamics of the CorInfoMax approach based on (6) with a toy picture. In this figure, the ellipsoid is a representative surface for the level set of the quadratic function $(\mathbf{z} - \boldsymbol{\mu}^{(q)})^T\mathbf{R}_{\mathbf{z}^{(q)}}^{-1}(\mathbf{z} - \boldsymbol{\mu}^{(q)})$, which reflects the spreading pattern of the latent vectors around their mean. The black arrows represent the gradient of the $\log\det(\mathbf{R}_{\mathbf{z}^{(q)}})$ regularization factor, which acts as a force to push latent vectors away from the center of the ellipsoid ($\boldsymbol{\mu}^{(q)}$) and therefore corresponds to the expansion force of the "big-bang" factor. In the same figure, the white arrows correspond to the gradients corresponding to the attraction factor, i.e., the Euclidean distance between two positive pairs. In summary, we can view the learning dynamics of CorInfomax SSL as an expansion in the latent space, while the representations of positive samples are attracted to each other. More information on the CorrInfoMax optimization setting is provided in the Appendix M. Furthermore, Appendix J contains a visualization of the embeddings obtained using the CorInfoMax criterion.

### 4.3 Computational complexity

The main difference between our and related SSL methods is the extra log-determinant computation. The log determinant is typically computed using a decomposition method such as the LU decomposition, whose gradient requires a matrix inversion. Both determinant and inversion have the same complexity as matrix multiplication [39], i.e. $O(n^\alpha)$ with $2 < \alpha \leq 3$ for multiplying two $n \times n$ matrices. In our experiments with GPUs, we observe an almost flat runtime cost up to $n = 1024$, which implies that the overhead due to the $\log\det$ cost is insignificant in practice. We find the extra cost of log determinant and its gradient to be negligible compared to the rest of the model computation, whose runtime is dominated by the encoder, as confirmed by the runtime experiments reported in Appendix G.

## 5 Experiments

### 5.1 Implementation details

*Datasets:* We perform experiments on CIFAR-10, CIFAR-100 [40], Tiny ImageNet [41], COCO [42], ImageNet-100 and ImageNet-1K [43] datasets[1]. For ImageNet-100, we use the same subset of ImageNet-1K [43] as related work [44–46]. (See Appendix D for more details.)

*Training Procedure:* The experiments consist of two consecutive stages: pretraining and linear evaluation. We first perform the unsupervised pretraining of the encoder network $f$ by applying the proposed CorInfoMax method described in Section 4.2 on the training dataset. After completing pretraining, we perform the linear evaluation, a standardized protocol to evaluate the quality of the learned representations [4, 8, 47].

For the linear evaluation stage, we first perform supervised training of the linear classifier using the representations obtained from the encoder network $f$ with frozen coefficients on the same training dataset. Then, we obtain the test accuracy results for the trained linear classifier based on the validation dataset.

[1]CorInfoMax's source code is publicly available in `https://github.com/serdarozsoy/corinfomax-ssl`

*Computational Resources:* For ImageNet-100 and ImageNet-1K, we pretrain our model on up to 8 A100 Cloud GPUs. The remaining datasets are trained using a single T4 and V100 Cloud GPU. Linear evaluations are performed using the same type and amount of computational resources. The details related to batch sizes are described in Section 5.1.3.

#### 5.1.1 Input augmentations

During the *pretraining stage*, two augmented versions of each input image are generated as shown in Figure 2. During this process, each image is cropped with random size, resized to the original resolution, followed by random applications of horizontal mirroring, color jittering, grayscale conversion, Gaussian blurring, and solarization. Since we use the same augmentation parameters as BYOL [4] and VicReg [6]: each augmentation branch uses the same probability values for these randomized operations except Gaussian blurring and solarization, which use different probabilities.

During the training phase of the *linear evaluation* stage, a single augmentation of each input image is produced by random cropping and resizing followed by a random horizontal flip. For the test phase of linear evaluation, we use resize and center crop augmentations, similar to [5, 6]. We provide more details about the augmentations in Appendix E.

#### 5.1.2 Network architecture

*Encoder Network*: For CIFAR datasets, we use a modified form of ResNet-18 architecture [48] similar to [8, 25, 33]. We use standard ResNet-50([48]) for Tiny ImageNet, ImageNet-100 and ImageNet-1K, and also standard ResNet-18([48]) for ImageNet-100. In all cases, the last fully connected layer is removed. Therefore, the encoder output size is 512 and 2048 for ResNet-18 and ResNet-50, respectively. The encoder network $f$ shares weights between augmented branches (see Appendix B).

*Projector Network:* The output of the encoder network is fed into the projector network, as in Figure 2. The projector network $p$ is a 3-layer MLP, with ReLU activation functions for the hidden layers and linear activation functions for the output layer. The projector dimensions are 2048-2048-64 for the CIFAR-10 dataset, 4096-4096-128 for the CIFAR-100, Tiny ImageNet and ImageNet-100, and 8192-8192-512 for the ImageNet-1K. Finally, we perform the $\ell_2$-normalization on the projector output.

*Linear Classifier:* For the linear evaluation phase, we employ a standard linear classifier whose input is the weight-frozen encoder network's output.

#### 5.1.3 Optimization

For pretraining, we use 1000 epochs with a batch size of 512 for CIFAR datasets, and 800 epochs with a batch size of 1024 for Tiny ImageNet. For ImageNet-100 experiments, we use 400 epochs for ResNet-18 and 200 epochs for ResNet-50, with a batch size of 1024 for both. ImageNet-1K experiments are conducted as 100 epochs with a batch size of 1536. We use the SGD optimizer with a momentum of 0.9 and a weight decay of $1 \times 10^{-4}$. The initial learning rate is 0.5 for CIFAR datasets and Tiny ImageNet, 1.0 for ImageNet-100, and 0.2 for ImageNet-1K. These learning rates follow the cosine decay with a linear warmup schedule.

We use the modified form of (6) as our objective function, where we replace $\frac{2\varepsilon^{-1}}{N}$ with $\alpha$, the attraction coefficient, which we consider as a separate hyper-parameter. In our experiments, the diagonal perturbation is $\varepsilon = 1 \times 10^{-8}$, while $\alpha = 250$ for CIFAR-10, $\alpha = 1000$ for CIFAR-100, $\alpha = 2000$ for ImageNet-1K, and $\alpha = 500$ for Tiny ImageNet and ImageNet-100. The forgetting factor $\lambda = 0.01$ for all datasets except Tiny ImageNet and ImageNet-1K, which have $\lambda = 0.1$. Details with coefficients of our loss function are provided in Appendix F.

For linear evaluation, the linear classifier is trained for 100 epochs with a batch size of 256 for all datasets. We used the SGD optimizer with a momentum of 0.9, and without weight decay. For all datasets except ImageNet-1K, cosine decay schedule is utilized with initial and minimum learning rates of 0.2 and $2 \times 10^{-3}$ respectively. For ImageNet-1K, we use a step scheduler with a starting value of 25, which is reduced by a factor of 10 every 20 epoch.

### 5.2 Results

We evaluate the learned representations from the CorInfoMax pretraining by following the linear evaluation protocol explained in Sec. 5.1. Table 1 shows that CorInfoMax achieves state-of-the-art performance in linear classification after pretraining. Due to the limited size of the validation sets (5-10K), differences less than $\approx 0.5\%$ are not statistically significant. The full comparison is provided in Table 10 in Appendix H. It is also interesting to observe the progress of the LDMI measure during the CorInfoMax training process, which is illustrated in Appendix L.

Table 1: Top-1 accuracies (%) under linear evaluation on different datasets. Results are reported from [30, 33, 49] for CIFAR-10 and CIFAR-100, [33] for Tiny ImageNet,[46, 50] for ImageNet-100 (IN-100) in in ResNet-50, [49] for ImageNet-100 (IN-100) in ResNet-18, [25, 33] for ImageNet-1K (IN-1K). In the case of the result of a model in more than one resource, we integrate the largest score. We **bold** all top results that are statistically indistinguishable.

| Method | CIFAR-10 | CIFAR-100 | Tiny-IN | IN-100 | | IN-1K |
|---|---|---|---|---|---|---|
| | ResNet-18 | ResNet-18 | ResNet-50 | ResNet-18 | ResNet-50 | ResNet-50 |
| SimCLR [8] | 91.80 | 66.83 | 48.12 | 77.04 | - | 66.5 |
| SimSiam [25] | 91.40 | 66.04 | 46.76 | 78.72 | 81.6 | 68.1 |
| Spectral [33] | 92.07 | 66.18 | 49.86 | - | - | 66.97 |
| BYOL [4] | 92.58 | 70.46 | - | **80.32** | 78.76 | **69.3** |
| W-MSE 2 [30] | 91.55 | 66.10 | - | 69.06 | - | - |
| MoCo-V2 [51] | **92.94** | 69.89 | - | 79.28 | - | 67.4 |
| Barlow [5] | 92.10 | 70.90 | - | **80.38** | - | 68.7 |
| VICReg [6] | 92.07 | 68.54 | - | 79.40 | - | 68.6 |
| CorInfoMax | **93.18** | **71.61** | **54.86** | **80.48** | **82.64** | 69.08 |

We compare the semi-supervised learning performance of CorInfoMax with VICReg [6]. For a fair comparison, VICReg is pretrained for 100 epochs on ImageNet-1K, then semi-supervised learning performances of both models are evaluated by fine-tuning the encoders with $1(\%)$ and $10(\%)$ of the labeled ImageNet-1K dataset. The results are presented in Table 2, while details are provided in Appendix F.5.

Table 2: Top-1 accuracies (%) under semi-supervised classification on ImageNet-1K dataset after 100 epoch pretraining. VICReg is pretrained and evaluated using hyper-parameters reported in [6].

| Method | 1% of samples | 10% of samples |
|---|---|---|
| VICReg | 44.75 | 62.16 |
| CorInfoMax | 44.89 | 64.36 |

We experimented with transfer learning on object detection and instance segmentation with the same procedure and code of MoCo[52], and reproduce the results with the MoCo V2 [53] checkpoint. Our algorithm shows competitive performance with MoCo V2 [53], related details are provided in Appendix I.

## 6 Discussions and conclusions

We proposed a novel SSL framework based on the correlative information maximization, CorInfoMax, which provides a natural solution for obtaining organized representations that are scattered in the latent space and avoiding collapse. Our experiments demonstrate the state-of-the-art performance of CorInfoMax in numerous downstream tasks. The loss function of CorInfoMax does not have any significant impact on training time. Future work can further improve CorInfoMax focusing on shortcomings like improved selection of hyperparameter and augmentation, a core problem in SSL overall.

## Acknowledgments and Disclosure of Funding

This work/research was supported by KUIS AI Center Research Award. We gratefully acknowledge the support of Google Cloud Credit Award.

## References

[1] Ronan Collobert and Jason Weston. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, pages 160–167, 2008.

[2] Carl Doersch, Abhinav Gupta, and Alexei A Efros. Unsupervised visual representation learning by context prediction. In *Proceedings of the IEEE international conference on computer vision*, pages 1422–1430, 2015.

[3] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[4] Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Guo, Mohammad Gheshlaghi Azar, et al. Bootstrap your own latent-a new approach to self-supervised learning. *Advances in Neural Information Processing Systems*, 33:21271–21284, 2020.

[5] Jure Zbontar, Li Jing, Ishan Misra, Yann LeCun, and Stéphane Deny. Barlow twins: Self-supervised learning via redundancy reduction. In *International Conference on Machine Learning*, pages 12310–12320. PMLR, 2021.

[6] Adrien Bardes, Jean Ponce, and Yann LeCun. Vicreg: Variance-invariance-covariance regularization for self-supervised learning. *arXiv preprint arXiv:2105.04906*, 2021.

[7] Linus Ericsson, Henry Gouk, and Timothy M Hospedales. How well do self-supervised models transfer? In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5414–5423, 2021.

[8] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR, 2020.

[9] Mathilde Caron, Ishan Misra, Julien Mairal, Priya Goyal, Piotr Bojanowski, and Armand Joulin. Unsupervised learning of visual features by contrasting cluster assignments. *Advances in Neural Information Processing Systems*, 33:9912–9924, 2020.

[10] Thomas M. Cover and Joy A. Thomas. *Elements of information theory*. Wiley-Interscience, 2006.

[11] Ben Poole, Sherjil Ozair, Aaron Van Den Oord, Alex Alemi, and George Tucker. On variational bounds of mutual information. In *International Conference on Machine Learning*, pages 5171–5180. PMLR, 2019.

[12] Michael Tschannen, Josip Djolonga, Paul K Rubenstein, Sylvain Gelly, and Mario Lucic. On mutual information maximization for representation learning. In *International Conference on Learning Representations*, 2019.

[13] David McAllester and Karl Stratos. Formal limitations on the measurement of mutual information. In *International Conference on Artificial Intelligence and Statistics*, pages 875–884. PMLR, 2020.

[14] Sam McCandlish, Jared Kaplan, Dario Amodei, and OpenAI Dota Team. An empirical model of large-batch training. *arXiv preprint arXiv:1812.06162*, 2018.

[15] Dominic Masters and Carlo Luschi. Revisiting small batch training for deep neural networks. *arXiv preprint arXiv:1804.07612*, 2018.

[16] Zhanghao Zhouyin and Ding Liu. Understanding neural networks with logarithm determinant entropy estimator. *arXiv preprint arXiv:1401.3420*, 2021.

[17] Alper T Erdogan. An information maximization based blind source separation approach for dependent and independent sources. In *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2022.

[18] Xiao Fu, Kejun Huang, Nicholas D Sidiropoulos, and Wing-Kin Ma. Nonnegative matrix factorization for signal and data analytics: Identifiability, algorithms, and applications. *IEEE Signal Process. Mag.*, 36(2):59–80, March 2019.

[19] Xiao Fu, Kejun Huang, Bo Yang, Wing-Kin Ma, and Nicholas D Sidiropoulos. Robust volume minimization-based matrix factorization for remote sensing and document clustering. *IEEE Transactions on Signal Processing*, 64(23):6254–6268, August 2016.

[20] Tsung-Han Chan, Wing-Kin Ma, ArulMurugan Ambikapathi, and Chong-Yung Chi. A simplex volume maximization framework for hyperspectral endmember extraction. *IEEE Transactions on Geoscience and Remote Sensing*, 49(11):4177–4193, May 2011.

[21] Eren Babatas and Alper T Erdogan. An algorithmic framework for sparse bounded component analysis. *IEEE Transactions on Signal Processing*, 66(19):5194–5205, August 2018.

[22] Huseyin A Inan and Alper T Erdogan. A convolutive bounded component analysis framework for potentially nonstationary independent and/or dependent sources. *IEEE Transactions on Signal Processing*, 63(1):18–30, November 2014.

[23] Gokcan Tatli and Alper T. Erdogan. Polytopic matrix factorization: Determinant maximization based criterion and identifiability. *IEEE Transactions on Signal Processing*, 69:5431–5447, 2021. doi: 10.1109/TSP.2021.3112918.

[24] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9729–9738, 2020.

[25] Xinlei Chen and Kaiming He. Exploring simple siamese representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15750–15758, 2021.

[26] Yuandong Tian, Xinlei Chen, and Surya Ganguli. Understanding self-supervised learning dynamics without contrastive pairs. In *International Conference on Machine Learning*, pages 10268–10278. PMLR, 2021.

[27] Mathilde Caron, Piotr Bojanowski, Armand Joulin, and Matthijs Douze. Deep clustering for unsupervised learning of visual features. In *Proceedings of the European conference on computer vision (ECCV)*, pages 132–149, 2018.

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

[29] Lei Huang, Dawei Yang, Bo Lang, and Jia Deng. Decorrelated batch normalization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 791–800, 2018.

[30] Aleksandr Ermolov, Aliaksandr Siarohin, Enver Sangineto, and Nicu Sebe. Whitening for self-supervised representation learning. In *International Conference on Machine Learning*, pages 3015–3024. PMLR, 2021.

[31] Tianyu Hua, Wenxiao Wang, Zihui Xue, Sucheng Ren, Yue Wang, and Hang Zhao. On feature decorrelation in self-supervised learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9598–9608, 2021.

[32] Yazhe Li, Roman Pogodin, Danica J Sutherland, and Arthur Gretton. Self-supervised learning with kernel dependence maximization. *Advances in Neural Information Processing Systems*, 34:15543–15556, 2021.

[33] Jeff Z HaoChen, Colin Wei, Adrien Gaidon, and Tengyu Ma. Provable guarantees for self-supervised deep learning with spectral contrastive loss. *Advances in Neural Information Processing Systems*, 34, 2021.

[34] Ralph Linsker. Self-organization in a perceptual network. *Computer*, 21(3):105–117, 1988.

[35] Suzanna Becker and Geoffrey E Hinton. Self-organizing neural network that discovers surfaces in random-dot stereograms. *Nature*, 355(6356):161–163, 1992.

[36] Anthony J Bell and Terrence J Sejnowski. An information-maximization approach to blind separation and blind deconvolution. *Neural computation*, 7(6):1129–1159, 1995.

[37] R Devon Hjelm, Alex Fedorov, Samuel Lavoie-Marchildon, Karan Grewal, Phil Bachman, Adam Trischler, and Yoshua Bengio. Learning deep representations by mutual information estimation and maximization. *arXiv preprint arXiv:1808.06670*, 2018.

[38] Justinian Rosca, Deniz Erdogmus, José C Príncipe, and Simon Haykin. *Independent Component Analysis and Blind Signal Separation: 6th International Conference, ICA 2006, Charleston, SC, USA, March 5-8, 2006, Proceedings*, volume 3889. Springer, 2006.

[39] James R Bunch and John E Hopcroft. Triangular factorization and inversion by fast matrix multiplication. *Mathematics of Computation*, 28(125):231–236, 1974.

[40] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images.(2009), 2009.

[41] Ya Le and Xuan Yang. Tiny imagenet visual recognition challenge. *CS 231N*, 7(7):3, 2015.

[42] Tsung-Yi Lin, Michael Maire, Serge J. Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: common objects in context. In David J. Fleet, Tomás Pajdla, Bernt Schiele, and Tinne Tuytelaars, editors, *Computer Vision - ECCV 2014 - 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V*, volume 8693 of *Lecture Notes in Computer Science*, pages 740–755. Springer, 2014. doi: 10.1007/978-3-319-10602-1_48. URL `https://doi.org/10.1007/978-3-319-10602-1_48`.

[43] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009. doi: 10.1109/CVPR.2009.5206848.

[44] Yonglong Tian, Dilip Krishnan, and Phillip Isola. Contrastive multiview coding. *arXiv preprint arXiv:1906.05849*, 2019.

[45] Yannis Kalantidis, Mert Bulent Sariyildiz, Noe Pion, Philippe Weinzaepfel, and Diane Larlus. Hard negative mixing for contrastive learning. *Advances in Neural Information Processing Systems*, 33:21798–21809, 2020.

[46] Songwei Ge, Shlok Mishra, Chun-Liang Li, Haohan Wang, and David Jacobs. Robust contrastive learning using negative samples with diminished semantics. *Advances in Neural Information Processing Systems*, 34, 2021.

[47] Alexander Kolesnikov, Xiaohua Zhai, and Lucas Beyer. Revisiting self-supervised visual representation learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 1920–1929, 2019.

[48] 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*, pages 770–778, 2016.

[49] Victor Guilherme Turrisi da Costa, Enrico Fini, Moin Nabi, Nicu Sebe, and Elisa Ricci. Solo-learn: A library of self-supervised methods for visual representation learning. *Journal of Machine Learning Research*, 23(56):1–6, 2022.

[50] Hankook Lee, Kibok Lee, Kimin Lee, Honglak Lee, and Jinwoo Shin. Improving transferability of representations via augmentation-aware self-supervision. *Advances in Neural Information Processing Systems*, 34, 2021.

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

[52] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. *arXiv preprint arXiv:1911.05722*, 2019.

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

[54] Thomas Kailath, Ali H Sayed, and Babak Hassibi. *Linear estimation*. Prentice Hall, 2000.

[55] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask R-CNN. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 2980–2988, 2017. doi: 10.1109/ICCV.2017.322.

[56] Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-SNE. *Journal of machine learning research*, 9(11), 2008.

## Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes] As claimed in the abstract and introduction, in the article we provide a novel self supervised learning framework based on the information maximization principle and provide numerical experiment results reflecting its performance.
   (b) Did you describe the limitations of your work? [Yes] We stated sensitivity to hyperparameter selections as a limitation in Section 6.
   (c) Did you discuss any potential negative societal impacts of your work? [N/A] We believe that our work does not have any potential negative societal impacts.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes] We have read the ethics review guidelines and our paper conforms to them.
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes] Theoretical discussions in Section 3 and Section 4 clearly provide the mathematical settings used in our derivations.
   (b) Did you include complete proofs of all theoretical results? [Yes] We show the details of the derivations of our proposed objective in Section 3.
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] We include the Python codes of our experiments in supplementary materials and the details of the experiment results in Section 5 and in Appendix.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] We provided all the details about training, its phases, hyperparameter selections in Section 5 and in Appendix.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] In Appendix, we provide information on the confidence interval of our experimental results.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] We reported the details about the computation resources and computation times in Section 5 and in Appendix.

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] We cite references for all the data sets, experiment results and network models in Section 5.
   (b) Did you mention the license of the assets? [N/A]
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes] We include our code as supplementary material.
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

## A Background on information measures

The most natural and conventional means to measure uncertainty of a real-valued random vector $\mathbf{x}$ is to use the joint Shannon differential entropy defined by [10]

$$h(\mathbf{x}) \quad = \quad - \int\limits_{\mathrm{dom}(f_{\mathbf{x}})} \log(f_{\mathbf{x}}(x)) f_{\mathbf{x}}(x) \mathrm{d}x = -E(\log(f_{\mathbf{x}}(\mathbf{x}))),$$

where $f_{\mathbf{x}}$ is the joint probability density function (pdf) of the components of $\mathbf{x}$. The corresponding Shannon mutual information (SMI) between the random vectors $\mathbf{x}$ and $\mathbf{y}$ is defined as

$$I(\mathbf{x};\mathbf{y}) \quad = \quad h(\mathbf{x}) - h(\mathbf{x}\,|\,\mathbf{y}) \tag{A.1}$$

where $h(\mathbf{x}\,|\,\mathbf{y}) = -E(\log(f_{\mathbf{x}\,|\,\mathbf{y}}(\mathbf{x}\,|\,\mathbf{y})))$ is the conditional entropy. Shannon mutual information is a measure of the dependence between its arguments.

For a given $r$-dimensional random vector $\mathbf{x}$ with the pdf $f_{\mathbf{x}}(\mathbf{x})$, log-determinant (LD) entropy is defined as [16, 17]

$$h_{LD}^{(\varepsilon)}(\mathbf{x}) = \frac{1}{2}\log\det(\mathbf{R}_{\mathbf{x}} + \varepsilon\mathbf{I}) + \frac{r}{2}\log(2\pi e), \tag{A.2}$$

where $\mathbf{R}_{\mathbf{x}}$ is the auto-covariance matrix of $\mathbf{x}$, and $\varepsilon$ is a small nonnegative parameter defined for the diagonal perturbation on $\mathbf{R}_{\mathbf{x}}$. We note that $h_{LD}^{(\varepsilon)}(\mathbf{x})$ is equivalent to Shannon's entropy when $\mathbf{x}$ is a Gaussian vector with covariance $\mathbf{R}_{\mathbf{x}}$ and $\varepsilon$ is equal to zero. However, we should underline that it is a standalone uncertainty measure solely based on the second-order statistics, which reflects linear dependence.

The joint LD-entropy of an $r$-dimensional random vector $\mathbf{x}$ and a $q$-dimensional random vector $\mathbf{y}$ is defined as the LD-entropy of the cascaded vector $\begin{bmatrix} \mathbf{x}^T & \mathbf{y}^T \end{bmatrix}^T$, i.e.,

$$h_{LD}^{(\varepsilon)}(\begin{bmatrix} \mathbf{x}^T & \mathbf{y}^T \end{bmatrix}^T) = \frac{1}{2}\log\det\left(\mathbf{R}_{\begin{bmatrix} \mathbf{x} \\ \mathbf{y} \end{bmatrix}} + \varepsilon\mathbf{I}\right) + \frac{r+q}{2}\log(2\pi e)$$

where

$$\mathbf{R}_{\begin{bmatrix} \mathbf{x} \\ \mathbf{y} \end{bmatrix}} = \begin{bmatrix} \mathbf{R}_{\mathbf{x}} & \mathbf{R}_{\mathbf{xy}} \\ \mathbf{R}_{\mathbf{xy}}^T & \mathbf{R}_{\mathbf{y}} \end{bmatrix}$$

is the covariance matrix of $\begin{bmatrix} \mathbf{x}^T & \mathbf{y}^T \end{bmatrix}^T$, $\mathbf{R}_{\mathbf{y}}$ is the auto-covariance matrix of $\mathbf{y}$, and $\mathbf{R}_{\mathbf{xy}}$ is the cross-covariance matrix of $\mathbf{x}$ and $\mathbf{y}$. Using the determinant decomposition based on Schur's complement [54], we can write

$$\begin{aligned} h_{LD}^{(\varepsilon)}(\begin{bmatrix} \mathbf{x}^T & \mathbf{y}^T \end{bmatrix}^T) \quad &= \frac{1}{2}\log\det(\mathbf{R}_{\mathbf{y}} + \varepsilon\mathbf{I}) + \frac{q}{2}\log(2\pi e) \\ &+ \frac{1}{2}\log\det(\mathbf{R}_{\mathbf{x}} - \mathbf{R}_{\mathbf{xy}}(\mathbf{R}_{\mathbf{y}} + \varepsilon\mathbf{I})^{-1}\mathbf{R}_{\mathbf{xy}}^T + \varepsilon\mathbf{I}) + \frac{r}{2}\log(2\pi e) \\ &= h_{LD}^{(\varepsilon)}(\mathbf{y}) + h_{LD}^{(\varepsilon)}(\mathbf{x}|\mathbf{y}), \end{aligned}$$

where we defined

$$h_{LD}^{(\varepsilon)}(\mathbf{x}|\mathbf{y}) = \frac{1}{2}\log\det(\mathbf{R}_{\mathbf{x}} - \mathbf{R}_{\mathbf{xy}}(\mathbf{R}_{\mathbf{y}} + \varepsilon\mathbf{I})^{-1}\mathbf{R}_{\mathbf{xy}}^T + \varepsilon\mathbf{I}) + \frac{r}{2}\log(2\pi e), \tag{A.3}$$

as the conditional LD-entropy. We note that the argument of the $\log\det$ function in (A.3) is the auto-covariance of the error for linearly estimating $\mathbf{x}$ from $\mathbf{y}$ with respect to the minimum mean square estimation (MMSE) criterion for $\varepsilon \to 0$ (see, for example, Theorem 3.2.2 of [54]). More precisely, given that $\mu_{\mathbf{x}}$ and $\mu_{\mathbf{y}}$ represent the means of $\mathbf{x}$ and $\mathbf{y}$, respectively, $\hat{\mathbf{x}}_{MMSE} = \mathbf{R}_{\mathbf{xy}}\mathbf{R}_{\mathbf{y}}^{-1}(\mathbf{y}-\mu_{\mathbf{y}})+\mu_{\mathbf{x}}$ is the best linear MMSE estimate of $\mathbf{x}$ from $\mathbf{y}$. Therefore, if we define $\mathbf{e}_{MMSE} = \mathbf{x} - \hat{\mathbf{x}}_{MMSE}$, the auto-covariance matrix of $\mathbf{e}_{MMSE}$ is given by $\mathbf{R}_{\mathbf{e}_{MMSE}} = \mathbf{R}_{\mathbf{x}} - \mathbf{R}_{\mathbf{xy}}\mathbf{R}_{\mathbf{y}}^{-1}\mathbf{R}_{\mathbf{xy}}^T$, which is the argument of the $\log\det$ function in (A.3) for $\varepsilon \to 0$. As a result, we can view $h_{LD}^{(\varepsilon)}(\mathbf{x}|\mathbf{y})$, which

is the LD-Entropy of $\mathbf{e}_{MMSE}$, as a measure of the remaining uncertainty after linearly (affinely) estimating $\mathbf{x}$ from $\mathbf{y}$ based on the MMSE criterion.

The LD-mutual information (LDMI) measure is defined based on (A.2) and (A.3) as follows:

$$\begin{aligned} I_{LD}^{(\varepsilon)}(\mathbf{x};\mathbf{y}) &= h_{LD}^{(\varepsilon)}(\mathbf{x}) - h_{LD}^{(\varepsilon)}(\mathbf{x}\,|\,\mathbf{y}) \\ &= \frac{1}{2}\log\det(\mathbf{R}_{\mathbf{x}}+\varepsilon\mathbf{I}) - \frac{1}{2}\log\det(\mathbf{R}_{\mathbf{x}} - \mathbf{R}_{\mathbf{xy}}(\mathbf{R}_{\mathbf{y}}+\varepsilon\mathbf{I})^{-1}\mathbf{R}_{\mathbf{xy}}^T + \varepsilon\mathbf{I}). \end{aligned} \quad \text{(A.4)}$$

Taking the average of (A.4) with its symmetric version obtained by exchanging $\mathbf{x}$ and $\mathbf{y}$, we can obtain an alternative but equivalent expression for LDMI:

$$\begin{aligned} I_{LD}^{(\varepsilon)}(\mathbf{x};\mathbf{y}) &= \frac{1}{4}\log\det(\mathbf{R}_{\mathbf{x}}+\varepsilon\mathbf{I}) + \frac{1}{4}\log\det(\mathbf{R}_{\mathbf{y}}+\varepsilon\mathbf{I}) \\ &\quad -\frac{1}{4}\log\det(\mathbf{R}_{\mathbf{x}} - \mathbf{R}_{\mathbf{xy}}(\mathbf{R}_{\mathbf{y}}+\varepsilon\mathbf{I})^{-1}\mathbf{R}_{\mathbf{xy}}^T + \varepsilon\mathbf{I}) \\ &\quad -\frac{1}{4}\log\det(\mathbf{R}_{\mathbf{y}} - \mathbf{R}_{\mathbf{yx}}(\mathbf{R}_{\mathbf{x}}+\varepsilon\mathbf{I})^{-1}\mathbf{R}_{\mathbf{yx}}^T + \varepsilon\mathbf{I}). \end{aligned} \quad \text{(A.5)}$$

The following lemma asserts that $I_{LD}^{(\varepsilon)}(\mathbf{x},\mathbf{y})$ is an information measure reflecting the correlation or "linear dependence" between two vectors [17]:

**Lemma 1** *Let $\mathbf{x},\mathbf{y}$ be random vectors with the auto-covariance matrices $\mathbf{R}_{\mathbf{x}} > 0$ and $\mathbf{R}_{\mathbf{y}}$, respectively, and the cross-covariance matrix $\mathbf{R}_{\mathbf{xy}}$. Then,*

- $I_{LD}^{(\varepsilon)}(\mathbf{x};\mathbf{y}) \geq 0$,
- $I_{LD}^{(\varepsilon)}(\mathbf{x};\mathbf{y}) = 0$ *if and only if* $\mathbf{R}_{\mathbf{xy}} = \mathbf{0}$, *that is,* $\mathbf{x}$ *and* $\mathbf{y}$ *are uncorrelated.*

## B The weight-shared SSL setup

In all our experiments we use weight-sharing between two branches:

$$\begin{aligned} f_1(\cdot;\mathbb{W}_{DNN}^{(1)}) &= f_2(\cdot;\mathbb{W}_{DNN}^{(2)}) = f(\cdot;\mathbb{W}_{DNN}) \\ p_1(\cdot;\mathbb{W}_{P}^{(1)}) &= p_2(\cdot;\mathbb{W}_{P}^{(2)}) = p(\cdot;\mathbb{W}_{P}), \end{aligned}$$

The two-branch network in Figure 2 is equivalent to the setup illustrated in Figure A.1. The augmentation outputs $\mathcal{X}^{(1)}$ and $\mathcal{X}^{(2)}$ are input to the same DNN, i.e., $f(\cdot;\mathbb{W}_{DNN})$ whose output is input to the projection network $p(\cdot;\mathbb{W}_P)$.

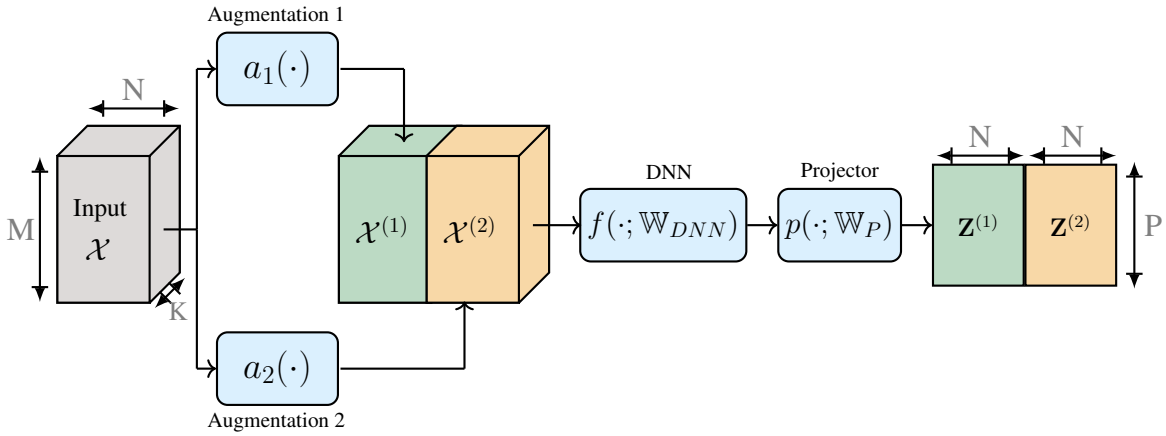


Figure A.1: Self-supervised learning set-up with weight sharing.

## C Pseudocode

Algorithm 1 (next page) describes the main steps for the CorInfoMax approach in a PyTorch-style pseudocode format.

**Algorithm 1:** PyTorch-style pseudocode for CorInfoMax

```
# f:  encoder with projector network
# N: batch size, D: projector output dimension
# R1 and R2:  covariance matrices are initialized as identity (DXD)
# mu1 and mu2:  means vectors are initiliazed as zero (D)
# lambda:  forgetting factor, alpha:  attraction coefficient
# mse_loss:  mean squared error loss, @:  matrix multiplication
for x in loader:  # load input batch
   # random augmentations
   x1, x2 = augmentation(x)
   # projector outputs
   z1 = f(x1)
   z2 = f(x2)
   # mean estimation
   mu1_update = z1.mean(0)
   mu2_update = z2.mean(0)
   mu1 = lambda * mu1 + (1 - lambda) * mu1_update
   mu2 = lambda * mu2 + (1 - lambda) * mu2_update
   # covariance matrix estimation
   z1_hat = z1 - mu1
   z2_hat = z2 - mu2
   R1_update = (z1_hat.T @ z1_hat) / N
   R2_update = (z2_hat.T @ z2_hat) / N
   R1 = lambda * R1 + (1 - lambda) * R1_update
   R2 = lambda * R2 + (1 - lambda) * R2_update
   # loss calculation
   cor_loss = - (logdet(R1) + logdet(R2)) / D # bing-bang factor
   sim_loss = mse_loss(z1, z2) # attraction factor
   loss = cor_loss + alpha * sim_loss
   # optimization
   loss.backward()
   optimizer.step()
```

## D Datasets

- The *CIFAR*-10 dataset [40] consists of $32 \times 32$ images with 10 classes. There are 5000 training images and 1000 validation images for each class.
- The *CIFAR*-100 dataset [40] consists of $32 \times 32$ images with 100 classes. There are 500 training images and 100 validation images for each class.
- The *Tiny ImageNet* dataset [41] consists of $64 \times 64$ images with 200 classes. There are 500 training images and 50 validation images for each class.
- The *ImageNet*-1*K* dataset [43] has 1281167 different sizes of images from 1000 classes as training set. Set of 50000 validation images is treated as a test dataset for evaluation purposes.
- The *ImageNet*-100 dataset [44–46] contains 100 sub-classes of the ImageNet dataset [43], which consists of images with variety of sizes. There are 1300 train images and 50 validation images for each class.
- The *COCO* dataset [42] consists of 164K images with annotations for object detection, instance segmentation, captioning, keypoint detection, and per-pixel segmentation. The training split contains 118K images, while the validation set contains 5K images. The test set contains 41K images.

Images in all these datasets have three color channels.

# E Image augmentations

## E.1 Augmentations during pretraining

During CorInfoMax pretraining, we use the following set of augmentations:

- Random resized cropping: cropping a random area of the input image with a scale parameter $(0.08, 1.0)$. Then resizing that cropped area to $32 \times 32$ for CIFAR datasets, $64 \times 64$ for Tiny ImageNet, and $224 \times 224$ for ImageNet-100 and ImageNet-1K.
- Horizontal flipping: mirroring the input image horizontally (left-right).
- Color jittering: changing the color properties of the input image. Brightness, contrast, and saturation are (uniform) randomly selected from $[max(0, 1 - \text{offset}), 1 + \text{offset}]$. Hue is selected from $[-\text{value}, \text{value}]$. The offset and value parameters used are given in Table 3.
- Grayscale: converting the RGB image to a grayscale image with three channels using $(0.2989 \times r + 0.587 \times g + 0.114 \times b)$.
- Gaussian blurring: smoothing the input image by filtering with a Gaussian kernel. The radius parameter for the kernel is selected uniformly between $0.1$ and $2.0$ pixels.
- Solarization: inverting image pixel values by subtracting the maximum value. We keep the result if it is above the threshold value; otherwise, replace it with the original value. The default threshold value is $128$.

The spatial dimensions of images input to the encoder networks are $32 \times 32$ for CIFAR datasets, $64 \times 64$ for Tiny ImageNet, and $224 \times 224$ for ImageNet-100 and ImageNet-1K. All pretraining augmentation parameters are listed in Table 3.

Table 3: Augmentation parameters are used in pretraining. Aug-1 and Aug-2 refer to augmentations for each branch. Transformations are selected independently for the two branches with the given probability values. The offset and maximum values determine the interval for uniform selection.

| Transformation | Aug-1 | Aug-2 |
|---|---|---|
| Random resized cropping probability | 1.0 | 1.0 |
| Horizontal flipping probability | 0.5 | 0.5 |
| Color Jitter (CJ) probability | 0.8 | 0.8 |
| CJ - Brightness offset | 0.4 | 0.4 |
| CJ - Contrast offset | 0.4 | 0.4 |
| CJ - Saturation offset | 0.2 | 0.2 |
| CJ - Hue maximum value | 0.1 | 0.1 |
| Grayscale probabillity | 0.2 | 0.2 |
| Gaussian blur probability | 1.0 | 0.1 |
| Solarization probability | 0.0 | 0.2 |

## E.2 Augmentations during linear evaluation

In the linear evaluation stage, a single transformed version of the input image is generated during both the training and the test phases.

- In the *training phase*, the random-resized-crop and horizontal-flip operations are used as augmentations as in [4–6, 25]. For the random-resized-crop operation, the target sizes are $32 \times 32$ for CIFAR datasets, $64 \times 64$ for Tiny ImageNet, and $224 \times 224$ for ImageNet-100 and ImageNet-1K.
- In the *test phase* of ImageNet-100 and ImageNet-1K, we apply the same preprocessing operation used for the (linear evaluation) test phase of the ImageNet dataset in [4]: input images are resized to $256 \times 256$ then center cropped to $224 \times 224$. For the other datasets, we apply a similar preprocessing by preserving the resize-crop ratio, as in [33].

## E.3 Additional information about augmentation

As the last step of the augmentation process, we normalize each channel of the resulting tensors by the mean and standard deviation of that channel calculated over the whole input dataset. The

mean and standard deviation normalization values for the channels are $(0.4914, 0.4822, 0.4465)$ and $(0.247, 0.243, 0.261)$ respectively, for CIFAR-10. For CIFAR-100, the corresponding normalization values are $(0.5071, 0.4865, 0.4409)$ and $(0.2673, 0.2564, 0.2762)$; for Tiny ImageNet, ImageNet-100 and ImageNet-1K: $(0.485, 0.456, 0.406)$ and $(0.229, 0.224, 0.225)$. As an interpolation method for resizing, we use bicubic interpolation.

# F Hyper-parameters

To select the hyper-parameters, we tested the following values: For CIFAR datasets, we examined $\alpha = [250, 500, 1000]$, projector output dimensions [64, 128, 256], projector hidden dimensions [2048, 4096] with [2, 3] layers setting. For the Tiny ImageNet dataset, we tested $\alpha = [250, 500, 1000]$, the projector dimensions [4096-4096-128, 4096-4096-256], and learning rate $= [0.25, 0.5, 1.0]$ and the forgetting factor [0.1, 0.01]. For ImageNet-100, we found the current best parameters in the same parameters set after the Tiny ImageNet experiments.

For ImageNet-1K, we tried $\alpha = [1000, 2000, 3000]$. Using the results of previous experiments in smaller datasets, increasing the output dimension with batch size provides an increase in the test accuracy performance in a limited number of experiments. We get our reported result with the projector dimension 8192-8192-512 and a batch size of 1536.

For the covariance update expression in (7), we use the initialization $\hat{\mathbf{R}}_{\mathbf{z}^{(q)}}[0] = \mathbf{I}$ for $q = 1, 2$. We use $\boldsymbol{\mu}^{(1)}[0] = \boldsymbol{\mu}^{(1)}[0] = \mathbf{0}$ for the mean initializations. The forgetting factor is $\lambda = 0.01$. Diagonal perturbation constant for the covariance matrices is $\varepsilon = 1 \times 10^{-8}$.

Regarding all linear evaluations, we train the linear classifier for 100 epochs with a batch size of 256. We use the SGD optimizer with a momentum of 0.9, no weight decay. For all datasets except ImageNet-1K, a learning rate of 0.2 is chosen. We use the cosine decay learning rate rule as a scheduler with a minimum learning rate of 0.002. For ImageNet-1K, we use a step scheduler where the learning rate starts with 25 and reduces by a factor of 10 for each 20 epochs.

Below, we summarize the experimental details for each dataset:

## F.1 CIFAR-10 experiment

For CIFAR-10, the encoder network is ResNet-18 modified for CIFAR based on the small $(32 \times 32)$ input image size. The modifications are: using a smaller kernel size, $3 \times 3$ instead of $7 \times 7$, in the first convolutional layer, and dropping the max-pooling layer. For the projection block, we use a 3-layer MLP network with dimensions 2048-2048-64.

We pretrain for 1000 epochs using a batch size of 512 using an SGD optimizer with a momentum of 0.9, and maximum learning rate of 0.5. For the learning rate scheduler, we utilize the cosine decay schedule after a linear warm-up for 10 epochs. During the linear warm-up period, the learning rate starts at 0.003. At the end of the training, it reaches its minimum value, which is set as $1 \times 10^{-6}$. The weight decay parameter for the optimizer is 0.0001.

## F.2 CIFAR-100 experiment

For CIFAR-100, the encoder network is modified ResNet-18, as explained in the above part. For the projection network, we use a 3-layer MLP with sizes of 4096-4096-128.

We pretrain for 1000 epochs with a batch size of 512. We use SGD optimizer with a momentum of 0.9, and a maximum learning rate of 0.5. As a scheduler, we utilize cosine decay learning rate with linear warm-up for 10 epochs. The learning rate start value is 0.003 for the warm-up period. After reaching 0.5, the learning rate decays to $1 \times 10^{-6}$ until the end of the training. The weight decay parameter for the optimizer is 0.0001.

## F.3 Tiny ImageNet experiment

For Tiny ImageNet, the encoder network is standard ResNet-50. For the projection network, we use a 3-layer MLP with sizes of 4096-4096-128.

We pretrain our model for $800$ epochs to make it more comparable with other methods in Table 1, and use a batch size of $1024$. We use the SGD optimizer with a momentum of $0.9$, and a maximum learning rate of $0.5$. As a scheduler, we use the cosine decay learning rate with linear warm-up for $10$ epochs. The learning rate start value is $0.003$ for the warm-up period. After reaching $0.5$, the learning rate decays to $1 \times 10^{-6}$ until end of the training. The weight decay parameter for the optimizer is $0.0001$.

### F.4 ImageNet-100 experiments

#### F.4.1 ResNet-18

In this experiment for ImageNet-100, the encoder network is standard ResNet-18. We use a 3-layer MLP with sizes of 4096-4096-128 as the projection network.

We pretrain $400$ epochs to make it more comparable with other methods in Table 1, using a batch size of $1024$. We use the SGD optimizer with momentum of $0.9$, and maximum learning rate of $1.0$. As a scheduler, we use the cosine decay learning rate with linear warmup for $10$ epochs. The learning rate start value is $0.003$ for the warm-up period. After reaching $1.0$, the learning rate decays to $0.005$ until the end of the training. The weight decay parameter for the optimizer is $0.0001$.

#### F.4.2 ResNet-50

In this experiment for ImageNet-100, the encoder network is standard ResNet-50. We use a 3-layer MLP with sizes of 4096-4096-128 as the projection network.

We pretrain $200$ epochs to make it more comparable with other methods in Table 1, using a batch size of $1024$. We use the SGD optimizer with a momentum of $0.9$, and a maximum learning rate of $1.0$. As a scheduler, we use the cosine decay learning rate with linear warmup for $10$ epochs. The learning rate start value is $0.003$ for the warm-up period. After reaching $1.0$, the learning rate is decayed to $0.005$ until end of training. The weight decay parameter for the optimizer is $0.0001$.

### F.5 ImageNet-1K experiments

The encoder network is standard ResNet-50 in ImageNet-1K experiments. As a projection network in pretraining, a 3-layer MLP with sizes of 8192-8192-512 with batch-normalization is utilized. Note that the increase in the number of classes, relative to the Imagenet-100 dataset, translates into an increase in the projector dimension. This further translates into an increased batch size for more accurate estimation of projector covariance matrix with larger dimensions. We pretrain $100$ epochs with a batch size of $1536$. We use the SGD optimizer with a momentum of $0.9$, and a maximum learning rate of $0.2$. As a scheduler, we use the cosine decay learning rate with linear warmup for $10$ epochs. The learning rate start value is $0.003$ for the warm-up period. After reaching $0.2$, the learning rate is decayed to $1 \times 10^{-6}$ until end of training. The weight decay parameter for the optimizer is $0.0001$.

#### F.5.1 Semi-supervised learning

In semi-supervised learning, we fine-tune our pretrained model on ImageNet-1K. In contrast to linear evaluation, weights of the encoder also change. Fine-tuning runs $20$ epochs with a batch size of $256$. For fine-tuning with $1\%$ samples of ImageNet-1K, learning rate for the encoder network is $0.005$, and the learning rate for the linear classifier head is $30$. For fine-tuning with $10\%$ samples of ImageNet-1K, learning rate for the encoder network is $0.005$, and the learning rate for the linear classifier head is $20$. We used separate step-type learning schedulers for the header and the backbone components, where the learning rate is reduced by a factor of $10$ for the header and a factor of $5$ for the backbone network for every $5$-epoch interval.

We used a learning rate of $0.3$ with a batch size of $2048$ for the VICReg pretraining[6], then used published parameters and VICReg codes[6] to evaluate the fine-tuning results.

### F.6 Hyper-parameter sensitivities

We provide Top-1 test accuracy results after 1000 epochs of pretraining on the CIFAR-100 dataset for various hyper-parameter adjustments below. The results show that our approach is fairly robust to small hyper-parameter changes.

#### F.6.1 Attraction coefficient

Table 4: Test accuracy results for the different attraction coefficients ($\alpha$) with the setup which provides best result for CIFAR-100 dataset.

| Attraction coefficient ($\alpha$) | 125 | 250 | 500 | 1000 | 2000 |
|---|---|---|---|---|---|
| Top-1 accuracy | 69.04 | 71.19 | 71.34 | 71.61 | 69.96 |

#### F.6.2 Batch size

Table 5: Test accuracy results for the different batch sizes with the setup which provides best result for CIFAR-100 dataset.

| Batch sizes | 256 | 512 | 1024 |
|---|---|---|---|
| Top-1 accuracy | 70.32 | 71.61 | 69.98 |

#### F.6.3 Learning rate

Table 6: Test accuracy results for the different learning rates with the setup which provides best result for CIFAR-100 dataset.

| Learning rates | 0.25 | 0.5 | 1.0 |
|---|---|---|---|
| Top-1 accuracy | 71.32 | 71.61 | 69.95 |

#### F.6.4 Projector Output Dimension

Table 7: Test accuracy results for the different projector output dimensions with the setup which provides best result for CIFAR-100 dataset.

| Projector output dimension | 64 | 128 | 256 |
|---|---|---|---|
| Top-1 accuracy | 68.19 | 71.61 | 71.26 |

## G Algorithm runtime results

In Section 4.3 we described the computational complexity of the CorInfoMax approach and stated that its overall effect on runtime is not significant. In this section we provide some experimental evidence.

We selected VICReg [6] for comparison. We integrated their loss function to our code to eliminate differences in implementation of methods. We ran 10 epochs with both loss functions for different projector output dimensions changing between $64$ and $1024$, and the results are reported in Table 8.

Table 8: Runtime results for the CIFAR-10 dataset with a batch size of $512$ on a T4 Cloud GPU. Average seconds per epoch from 10 test runs is reported. The loss function of VICReg [6] is integrated to our code for comparison.

| Projector Dimensions | VICReg | CorInfoMax |
|---|---|---|
| 2048-2048-64 | 87.77 | 87.33 |
| 2048-2048-128 | 87.59 | 87.94 |
| 2048-2048-256 | 88.00 | 88.55 |
| 2048-2048-512 | 88.19 | 90.89 |
| 2048-2048-1024 | 89.35 | 102.1 |

Furthermore, we experiment with different batch and projector sizes in order to measure the proportion of time spent on the calculation of the log-determinant in the loss function as shown in Table 9. For these experiments, we use Imagenet-1K and ResNet-50 as the encoder on a V100 Cloud GPU on a machine $5$ cores of an Intel Xeon Gold $6248$ CPU. We find that the cost of the log-determinant operation is at worst around $5.5\%$ of the computation time for the range of hyperparameters we explore. We thus conclude that the log-determinant calculation does not contribute considerably to the overall training cost.

Table 9: Proportion of time spent in log-determinant calculation for Imagenet-1K and ResNet-50 with varying batch and projector sizes on a V100 Cloud GPU. Average results from $20$ test runs is reported.

| Batch Size | Projector Dimension | | | | |
|---|---|---|---|---|---|
| | 64 | 128 | 256 | 512 | 1024 |
| 256 | 0.40% | 0.56% | 1.06% | 2.53% | 5.54% |
| 512 | 0.20% | 0.28% | 0.53% | 1.27% | 2.83% |

## H Full comparison with solo-learn

Some of the results in Table 1 are from the solo-learn paper [49]. Table 10 shows the full results from [49] in comparison to CorInfoMax.

Table 10: Top-1 and Top-5 accuracies (%) under linear evaluation on CIFAR-10, CIFAR-100, and ImageNet-100 datasets with ResNet-18. We bold all top results that are statistically indistinguishable.

| Method | CIFAR-10 | | CIFAR-100 | | ImageNet-100 | |
|---|---|---|---|---|---|---|
| | Top-1 | Top-5 | Top-1 | Top-5 | Top-1 | Top-5 |
| Barlow Twins | 92.10 | 99.73 | 70.90 | **91.91** | **80.38** | **95.28** |
| BYOL | 92.58 | 99.79 | 70.46 | **91.96** | **80.32** | **94.94** |
| DeepCluster V2 | 88.85 | 99.58 | 63.61 | 88.09 | 75.40 | 93.22 |
| DINO | 89.52 | 99.71 | 66.76 | 90.34 | 74.92 | 92.92 |
| MoCo V2+ | **92.94** | 99.79 | 69.89 | 91.65 | 79.28 | **95.50** |
| NNCLR | 91.88 | 99.78 | 69.62 | 91.52 | **80.16** | **95.28** |
| ReSSL | 90.63 | 99.62 | 65.92 | 89.73 | 78.48 | 94.24 |
| SimCLR | 90.74 | 99.75 | 65.78 | 89.04 | 77.48 | 94.02 |
| SimSiam | 90.51 | 99.72 | 66.04 | 89.62 | 78.72 | 94.78 |
| SwAV | 89.17 | 99.68 | 64.88 | 88.78 | 74.28 | 92.84 |
| VICReg | 92.07 | 99.74 | 68.54 | 90.83 | 79.40 | **95.06** |
| W-MSE | 88.67 | 99.68 | 61.33 | 87.26 | 69.06 | 91.22 |
| CorInfoMax | **93.18** | 99.88 | **71.61** | **92.40** | **80.48** | **95.46** |

## I Transfer learning for object detection and instance segmentation example

In this section, we outline our experiments on object detection and instance segmentation in order to explore the capability of models trained using our approach to capture fine-grained details, which are otherwise not explicitly explored in classification tasks. We fine-tune our model after 100 epochs of pretraining on ImageNet-1k using Mask R-CNN [55] (C4 Backbone) on COCO [42], following the same procedure as MoCo [52]. To have a reference point in the same exact environment, we finetune the pretrained MoCo V2 [53] checkpoint for 200 epochs on ImageNet-1K. Table 11 shows the results for both tasks. We achieve similar performance as MoCo V2. We leave further optimization, experimentation on other tasks and datasets, and further exploration of the fine-grained features learned by our model to future work.

Table 11: AP, AP50, and AP75 for object detection and instance segmentation on COCO. All models have been trained on the 2017 training split and evaluated on the 2017 validation split.

| Method | COCO Detection | | | COCO Segmentation | | |
|---|---|---|---|---|---|---|
| | $AP_{50}$ | AP | $AP_{75}$ | $AP_{50}$ | AP | $AP_{75}$ |
| MoCo V2 | 60.52 | 40.77 | 44.19 | 57.33 | 35.56 | 38.12 |
| CorInfoMax | 60.56 | 40.50 | 43.89 | 57.22 | 35.34 | 37.70 |

## J Embedding visualization after pretraining

In Figure A.2, we use t-Distributed Stochastic Neighbor Embedding (t-SNE) [56] to visualize where embeddings of the test dataset are located after pretraining. We get embeddings from the output of the encoder network using our pretrained model, save them with their classification labels. Then using t-SNE, we visualize the 3D embedded space. We used the following t-SNE parameters: perplexity is 50, early exaggeration is 12, and iteration number is 1000, random initialization of embeddings is used, learning rate is 208.33. We also include a video in the supplementary material that shows the t-SNE plot with changing view angles.

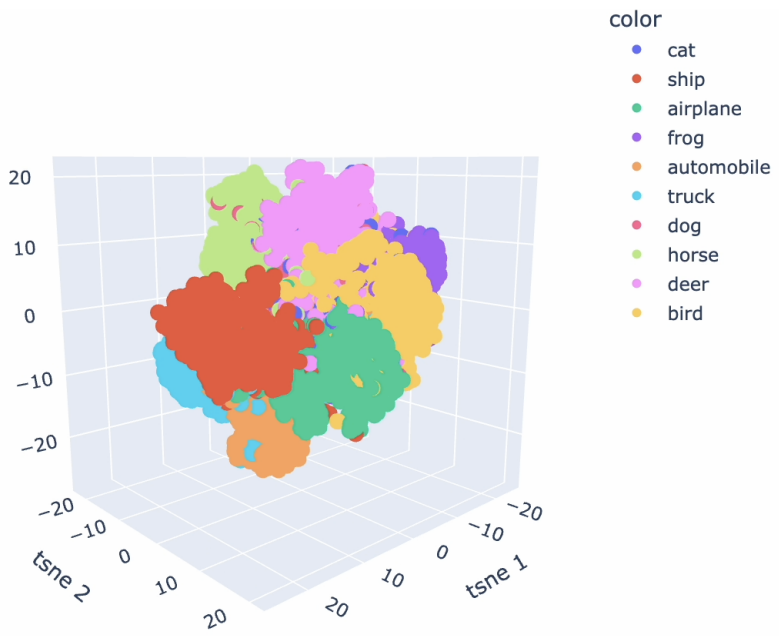


Figure A.2: t-SNE visualization of obtained embeddings of CIFAR-10 test dataset from the output of the encoder network after 1000-epoch pretraining. Each color represents one class of CIFAR-10.

## K Visualization of projector covariance matrix eigenvalues

The eigenvalues of the projector vector covariance matrix, $\hat{\mathbf{R}}_{\mathbf{z}^{(1)}}$ reflect the effective use of the embedding space. Due to the existence of the "big-bang factor", $\log\det(\hat{\mathbf{R}}_{\mathbf{z}^{(1)}}[l] + \varepsilon\mathbf{I})$, in (6), we expect that no dimensional collapse occurs (with very small $\varepsilon$), and eigenvalues take significant values. To confirm this expectation, we performed the following experiment: for the CIFAR-10 dataset, we ran the contrastive SimCLR algorithm and obtained its projector covariance matrix after training. Similarly, we naturally obtained the projector covariance matrix for the CorInfoMax approach. For both algorithms, we used a projector dimension of 128. Figure A.3 compares the sorted eigenvalues of both covariances. As can be observed from this figure, the effective embedding space dimension for the SimCLR algorithm is small, most of the energy is concentrated at about the first 30 eigenvalues. Furthermore, the smallest 9-eigenvalues are equal to 0, within numerical precision, indicating a dimensional collapse. On the contrary, the eigenvalues for the CorInfoMax algorithm are significant for all dimensions, hinting at the effective use of the embedding space.

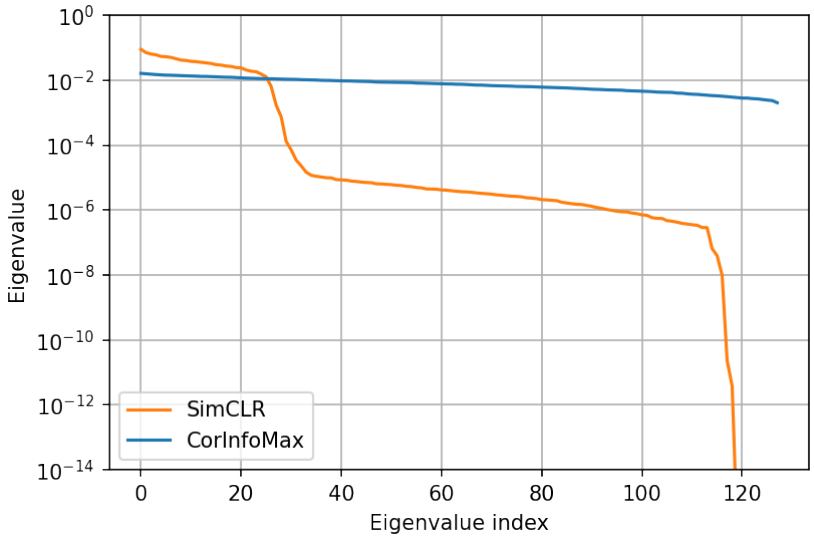


Figure A.3: Comparison of the sorted eigenvalues of the projector vector covariance matrix $\hat{\mathbf{R}}_{\mathbf{z}^{(1)}}$ for CorInfoMax and SimCLR algorithms, for CIFAR-10 dataset and projector dimension of 128.

## L Visualization of LD-mutual information evolution during pretraining

As the CorInfoMax objective function is derived from the LDMI measure, it would be interesting to observe its evolution in relation to algorithm epochs and (online) test accuracy. For this purpose, we performed two experiments with the CIFAR-10 and CIFAR-100 datasets, where we used the CorInfoMax loss function, but recorded the Training-LDMI values (based on the covariance estimates using (7) during training) and the test accuracy values. Figure A.4 shows the progress of the Training-LDMI and the test accuracy curves on the same graph for the CIFAR-10 dataset.

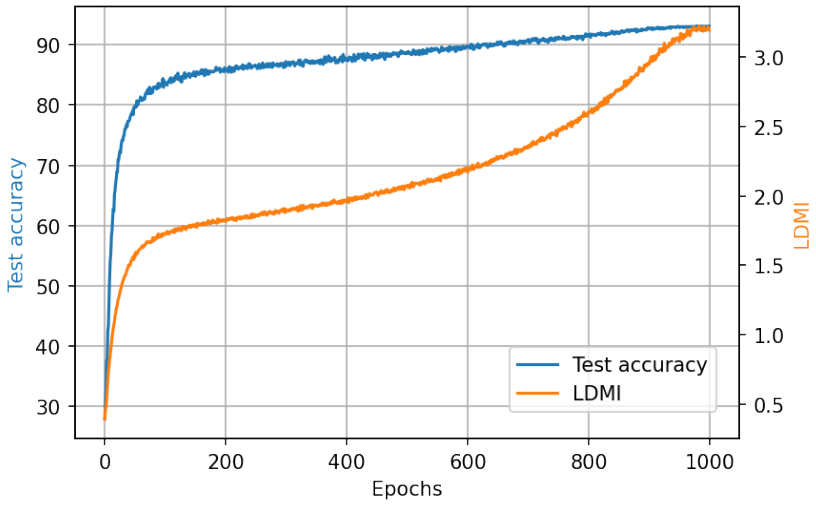


Figure A.4: Evolution of the LDMI measure and the test accuracy for the CIFAR-10 dataset as a function of the CorInfoMax algorithm epochs.

Similarly, Figure A.5 shows the progress of the Training-LDMI and the test accuracy curves on the same graph for the CIFAR-100 dataset.

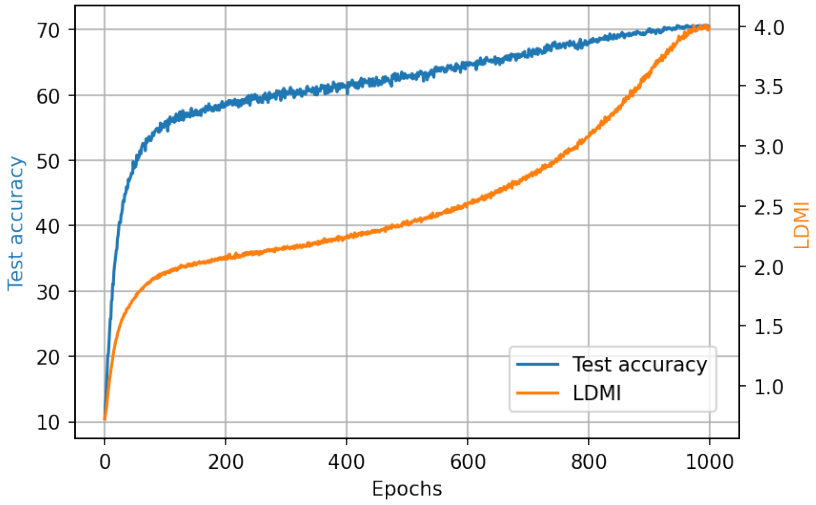


Figure A.5: Evolution of the LDMI measure and the test accuracy for the CIFAR-100 dataset as a function of the CorInfoMax algorithm epochs.

Both figures confirm that the Training-LDMI measure and the test accuracy increase together almost monotonically, ignoring the small variations.

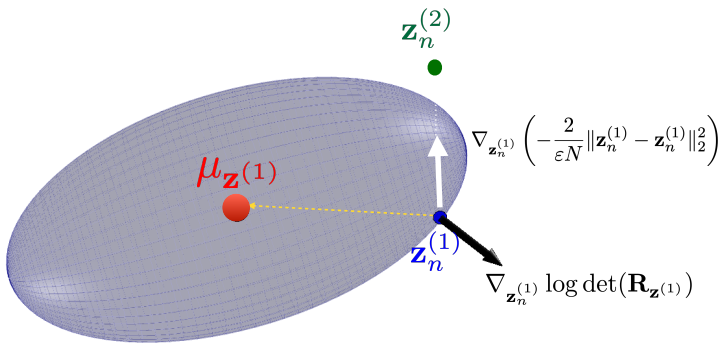


Figure A.6: Gradients of the objective function for CorInfoMax: The ellipsoid is the level surface of the quadratic function $q(\mathbf{z})$ in (A.9) containing $\mathbf{z}_n^{(1)}$, one of the projector-1 output samples, the black arrow represents the gradient of $\log\det(\mathbf{R}_{\mathbf{Z}^{(1)}})$ with respect to $\mathbf{z}_n^{(1)}$, the white arrow represents the gradient of $\frac{2}{\varepsilon N}\|\mathbf{z}_n^{(1)}-\mathbf{z}_n^{(2)}\|_2^2$ with respect to $\mathbf{z}_n^{(1)}$.

# M Supplementary notes on CorInfoMax criterion

## M.1 Gradients of the CorInfoMax objective

Let $\mathbf{z}_n^{(q)}$ represent the $n^{th}$ sample of the $q^{th}$ branch projector output for the current batch, where $n \in \{1,\ldots,N\}$ and $q=1,2$. We provide the gradient expressions of the CorInfoMax objective (6) with respect to the projector outputs, which are backpropagated to the train the encoder networks.

For the first term on the right side of (6), we can write

$$\nabla_{\mathbf{z}_n^{(1)}}\log\det(\hat{\mathbf{R}}_{\mathbf{Z}^{(1)}}[l]+\varepsilon\mathbf{I}) = \frac{(1-\lambda)}{N}(\hat{\mathbf{R}}_{\mathbf{Z}^{(1)}}[l]+\varepsilon\mathbf{I})^{-1}(\mathbf{z}_n^{(1)}-\boldsymbol{\mu}_{\mathbf{Z}^{(1)}}[l]), \tag{A.6}$$

and,

$$\nabla_{\mathbf{z}_n^{(2)}}\log\det(\hat{\mathbf{R}}_{\mathbf{Z}^{(1)}}[l]+\varepsilon\mathbf{I}) = 0.$$

Similarly, for the second term on the right side of (6), we can write

$$\nabla_{\mathbf{z}_n^{(2)}}\log\det(\hat{\mathbf{R}}_{\mathbf{Z}^{(2)}}[l]+\varepsilon\mathbf{I}) = \frac{(1-\lambda)}{N}(\hat{\mathbf{R}}_{\mathbf{Z}^{(2)}}[l]+\varepsilon\mathbf{I})^{-1}(\mathbf{z}_n^{(2)}-\boldsymbol{\mu}_{\mathbf{Z}^{(2)}}[l]),$$

and,

$$\nabla_{\mathbf{z}_n^{(1)}}\log\det(\hat{\mathbf{R}}_{\mathbf{Z}^{(2)}}[l]+\varepsilon\mathbf{I}) = 0.$$

Finally, for the rightmost term of (6), which is the Euclidian distance based loss, we can write

$$\nabla_{\mathbf{z}_n^{(1)}}\left(-\frac{2}{\varepsilon N}\|\mathbf{Z}^{(1)}[l]-\mathbf{Z}^{(2)}[l]\|_F^2\right) = \frac{4}{\varepsilon N}(\mathbf{z}_n^{(2)}-\mathbf{z}_n^{(1)}), \tag{A.7}$$

and

$$\nabla_{\mathbf{z}_n^{(2)}}\left(-\frac{2}{\varepsilon N}\|\mathbf{Z}^{(1)}[l]-\mathbf{Z}^{(2)}[l]\|_F^2\right) = \frac{4}{\varepsilon N}(\mathbf{z}_n^{(1)}-\mathbf{z}_n^{(2)}). \tag{A.8}$$

Inspecting the gradient expressions with respect to $\mathbf{z}_n^{(1)}$: The vector in (A.6) is the surface normal of the level set of the quadratic function

$$q(\mathbf{z}) = \frac{(1-\lambda)}{N}(\mathbf{z}-\boldsymbol{\mu}_{\mathbf{Z}^{(1)}}[l])^T(\hat{\mathbf{R}}_{\mathbf{Z}^{(1)}}[l]+\varepsilon\mathbf{I})^{-1}(\mathbf{z}-\boldsymbol{\mu}_{\mathbf{Z}^{(1)}}[l]), \tag{A.9}$$

at $\mathbf{z} = \mathbf{z}_n^{(1)}$, which is illustrated by the black vector in Figure A.6. Since $(\hat{\mathbf{R}}_{\mathbf{z}^{(1)}}[l] + \varepsilon\mathbf{I})^{-1}$ is positive definite, the inner product

$$\begin{aligned}
&\langle \nabla_{\mathbf{z}_n^{(1)}} \log\det(\hat{\mathbf{R}}_{\mathbf{z}^{(1)}}[l] + \varepsilon\mathbf{I}), \boldsymbol{\mu}_{\mathbf{z}^{(1)}}[l] - \mathbf{z}_n^{(1)} \rangle \\
&= -\frac{(1-\lambda)}{N}(\mathbf{z}_n^{(1)} - \boldsymbol{\mu}_{\mathbf{z}^{(1)}}[l])^T(\hat{\mathbf{R}}_{\mathbf{z}^{(1)}}[l] + \varepsilon\mathbf{I})^{-1}(\mathbf{z}_n^{(1)} - \boldsymbol{\mu}_{\mathbf{z}^{(1)}}[l])
\end{aligned}$$

is negative, which implies that the vector pointing from $\mathbf{z}_n^{(1)}$ towards the center $\mu_{z^{(1)}}[l]$, the yellow dashed arrow in Figure A.6, makes an obtuse angle with the gradient $\nabla_{\mathbf{z}_n^{(1)}} \log\det(\hat{\mathbf{R}}_{\mathbf{z}^{(1)}}[l] + \varepsilon\mathbf{I})$. Therefore, the gradient in (A.6) is pointing away from the center of the ellipsoid, encouraging the expansion.

On the other hand, the gradient expressions in (A.7) and (A.8) correspond to force pulling the positive samples $\mathbf{z}_n^{(1)}$ and $\mathbf{z}_n^{(2)}$ towards each other as indicated by the white arrow.

A video animation of sample points moving under these gradients is provided in the supplementary material.