Self-Supervised Learning with an Information Maximization Criterion

by

Serdar Özsoy

A Dissertation Submitted to the
Graduate School of Sciences and Engineering
in Partial Fulfillment of the Requirements for
the Degree of

Master of Science

in

Electrical and Electronics Engineering

KOÇ ÜNİVERSİTESİ

August 12, 2022

Self-Supervised Learning with an Information Maximization Criterion

Koç University

Graduate School of Sciences and Engineering

This is to certify that I have examined this copy of a master's thesis by

Serdar Özsoy

and have found that it is complete and satisfactory in all respects,
and that any and all revisions required by the final
examining committee have been made.

Committee Members:

_____

Prof. Alper T. Erdoğan (Advisor)

_____

Prof. Deniz Yüret

_____

Assoc. Prof. Erkut Erdem

Date: _____

To my wife...

ABSTRACT

Self-Supervised Learning with an Information Maximization Criterion
Serdar Özsoy
Master of Science in Electrical and Electronics Engineering
August 12, 2022

Self-supervised learning provides a solution to learn effective representations from large amounts of data without performing data labeling, which is often expensive in terms of time, effort, and cost.The main problem with the self-supervised learning approach, in general, is collapse, i.e., obtaining identical representations for all inputs while matching different representations generated from the same input. In this thesis, we argue that information maximization among latent representations of different versions of the same input naturally prevents collapse. To this end, we propose a novel self-supervised learning method, CorInfoMax, based on maximizing the second-order statistics-based measure of mutual information that reflects the degree of correlation between the latent representation arguments. Maximizing this correlative information measure between alternative latent representations of the same input serves two main purposes: (1) it avoids the collapse problem by generating feature vectors with non-degenerate covariances; (2) it increases the linear dependence between alternative representations, ensuring that they are related to each other. The proposed information maximization objective is simplified to an objective function based on Euclidean distance regularized by the log-determinant of the feature covariance matrix. Due to the regularization term acting as a natural barrier against feature space degeneracy, CorInfoMax also prevents dimensional collapse by enforcing representations to span across the entire feature space. Empirical experiments show that CorInfoMax achieves better or competitive performance results over state-of-the-art self-supervised learning methods across different tasks and datasets.

ÖZETÇE

Yüksek Lisans Tez Başlığı
Serdar Özsoy
Elektrik ve Elektronik Mühendisliği, Yüksek Lisans
12 Ağustos 2022

Kendi kendini denetleyen öğrenme, genellikle zaman, çaba ve maliyet açısından pahalı olan veri etiketleme gerçekleştirmeden, büyük miktarda veriden etkili temsilleri öğrenmek için bir çözüm sağlar. Genel olarak kendi kendini denetleyen öğrenme yaklaşımıyla ilgili temel sorun çökmedir, yani aynı girdiden üretilen farklı temsilleri eşleştirirken tüm girdiler için aynı temsilleri elde etmektir. Bu tezde, aynı girdinin farklı versiyonlarının gizli temsilleri arasındaki bilgi maksimizasyonunun doğal olarak çöküşü önlediğini ve farklı alt görevlerde rekabetçi ampirik sonuçlar elde ettiğini savunuyoruz. Bu amaçla, gizli temsil argümanları arasındaki korelasyon derecesini yansıtan, ikinci dereceden istatistik tabanlı karşılıklı bilgi ölçüsünü maksimize etmeye dayalı CorInfoMax adında yeni bir kendi kendini denetleyen öğrenme yöntemi öneriyoruz. Aynı girdinin alternatif gizli temsilleri arasında bu bağıntılı bilgi ölçüsünü en üst düzeye çıkarmak iki temel amaca hizmet eder: (1) dejenere olmayan kovaryanslara sahip özellik vektörleri üreterek çökme problemini önler; (2) alternatif temsiller arasındaki doğrusal bağımlılığı artırarak, birbirleri ile alakalı olmasını sağlar. Önerilen bilgi maksimizasyonu hedefi, özellik kovaryans matrisinin log-determinantı tarafından düzenlenen Öklid mesafesine dayalı bir amaç fonksiyonuna basitleştirilmiştir. Öznitelik alanı bozulmasına karşı doğal bir engel görevi gören düzenlileştirme terimi nedeniyle CorInfoMax, temsillerin tüm özellik alanı boyunca yayılmasını zorlayarak boyutsal çöküşü de önler. Ampirik deneyler, CorInfoMax'in farklı görevlerde ve veri kümelerinde en gelişmiş kendi kendini denetleyen öğrenme metotlarına göre daha iyi veya rekabetçi performans sonuçları elde ettiğini göstermektedir.

ACKNOWLEDGMENTS

First of all, I would like to express my deepest gratitude and appreciation to my thesis advisor Prof. Alper T. Erdoğan for his guidance, support and patience throughout this degree. He helped me to broaden my horizon with insightful discussions. He took his time to answer all my exhaustive questions with his knowledge and patience.

I would like to express my sincere gratitude to Prof. Deniz Yüret for both his participation in the jury committee and his unwavering support for the research project that forms the basis of my thesis. I was guided to the right point with his right questions and knowledge in every meeting.

I would also like to extend my gratitude to Assoc. Prof. Erkut Erdem for his participation in the jury committee. I had the opportunity to make useful additions to the thesis with his valuable feedback.

I would like to thank Sercan Arık for insightful suggestions. Thanks also to my colleague Shadi Hamdan for his support in scaling up the experiments.

I am also grateful to my former manager Ömer Faruk Özer for encouraging and supporting me to start this journey.

I would like to acknowledge KUIS AI Center for computational resources. I would also like to acknowledge Google for providing Google Cloud credit award in the research project that underpins my thesis.

Last but not least, I would like to thank my family for for their enduring support of me during thesis work. I'm deeply indebted to my beloved wife for her endless support, encouragement, and patience.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABBREVIATIONS

AMDIM	Augmented Multiscale Deep InfoMax
BYOL	Bootstrap Your Own Latent
CorInfoMax	Correlative Information Maximization
CPC	Contrastive Predictive Coding
DBN	Decorrelated Batch Normalization
DIM	Deep InfoMax
DNN	Deep Neural Network
GPU	Graphics Processing Unit
HSCI	Hilbert-Schmidt Independence Criterion
LD	Log-Determinant
LDMI	Log-Determinant Mutual Information
MoCo	Momentum Contrastive
MSE	Mean Squared Error
NCE	Noise-Contrastive Estimation
PCA	Principal Component Analysis
PDF	Probability Density Function
RMSE	Root Mean Squared Error
SMI	Shannon Mutual Information
SSL	Self Supervised Learning
t-SNE	t-distributed Stochastic Neighbor Embedding
VICReg	Variance-Invariance-Covariance Regularization

Chapter 1

INTRODUCTION

Due to its great success in the recent deep learning era, supervised learning is the primary method when having a large dataset with labels. Supervised learning uses data labels as a supervision signal for learning; model performance depends on the correct and consistent annotation of the training data. However, the annotation task is usually expensive in terms of effort, time, and cost for large datasets. This makes it difficult to create large labeled datasets and causes the use of only a small part of the data that is constantly produced in the world. In addition to large datasets, labeling can be difficult for small datasets that require extensive domain knowledge, such as medical data. Therefore, labeling is one of the main bottlenecks in supervised learning. Furthermore, the generalization ability of supervised learning is one of the primary concerns. Supervised models may experience performance degradation in downstream tasks and datasets that will differ from the data and task for which they were specifically trained. It is important to achieve high accuracy for different tasks and datasets as well as for specially trained tasks and datasets.

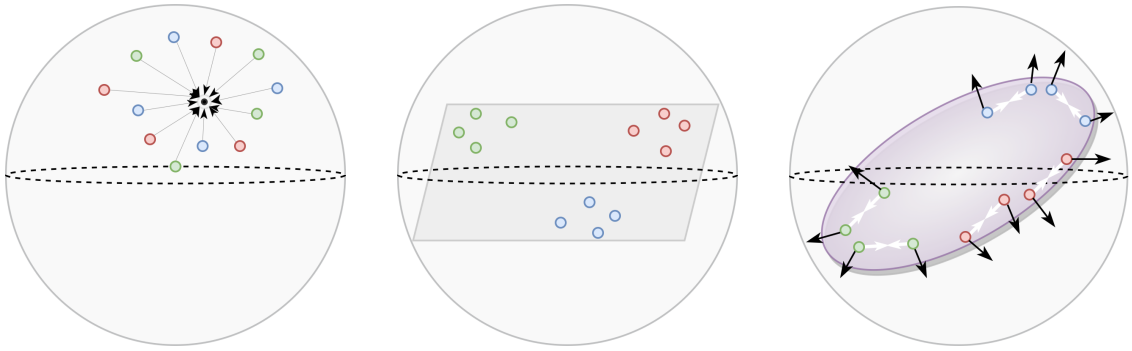
Self-supervised learning (SSL) has shown great potential to address these problems since its early stages, and recent studies have shown that the results of SSL compete with supervised learning [Chen and He, 2021, Zbontar et al., 2021, Bardes et al., 2021]. SSL uses the training data itself as a supervision signal. This can be done in various ways in different domains. Pretext tasks are specifically designed tasks to learn useful representations. For language modeling, the model can learn representations by predicting each next word using previous words [Mikolov et al., 2013]. In this case, the ground truth will be the data itself. For video tasks, the model can predict the next video frame, similar to language modeling [Agrawal et al., 2015]. Self-supervised

pretraining enables learning representations of input data without human annotations. These representations are useful for the downstream task. Although self-supervised learning is not a new term, great popularity comes from the success of large language models such as BERT [Devlin et al., 2018], GPT [Radford and Narasimhan, 2018], GPT-2 [Radford et al., 2019], T5 [Raffel et al., 2020], and GPT-3 [Brown et al., 2020].

Following language models, SSL usage in computer vision has also become widespread in recent years. Although complex pretext tasks such as rotation prediction of the rotated input image [Gidaris et al., 2018], relative position prediction of image patches [Doersch et al., 2015], and color prediction from the grayscale version of the input image [Zhang et al., 2016] had dominated in the early stages, input data augmentations combined with the use of contrastive loss over encoded representations were essentially groundbreaking in results [Chen et al., 2020a, He et al., 2020]. The gap between supervised learning and SSL has been reduced by newly proposed methods with different architectures and loss functions [Grill et al., 2020, Chen and He, 2021, Zbontar et al., 2021, Bardes et al., 2021].

In addition to language models and computer vision, speech recognition is another area of research in which impactful results have been published [Schneider et al., 2019, Baevski et al., 2020]. In a recent position paper [LeCun, 2022], self-supervised learning is proposed to train world models within a hierarchical architecture. The prevalence of self-supervised learning in different research directions seems to continue to increase.

Despite all this fascinating background and results, there are also bottlenecks for self-supervised learning. The main problem to solve in self-supervised learning is learning a trivial solution rather than meaningful representations. As a trivial solution, all feature representations are fixed to a constant vector, and this causes a collapse in representations as illustrated in Figure 1.1a. Recent work [Hua et al., 2021] has also shown that there exists another problem that representations can use only a small part of the embedding space with rank deficiency as in Figure 1.1b, even though representations do not collapse as constant vector. If we call the first problem *total collapse*, the second problem will be *dimensional collapse*, which



(a) (b) (c)

Figure 1.1: Depictions in latent space : (a)"Total collapse": all latent vectors converge to the same point, (b) "Dimensional collapse": latent vectors are restricted to a strict subspace of the latent space, (c) Gradient dynamics of the CorInfoMax based on (4.2): the ellipsoid surface reflects the average spread pattern of latent vectors. While effect of "big-bang factor" is shown with white arrows, effect of attraction factor is shown with black arrows for CorInfoMax.

causes performance degradation [Hua et al., 2021]. In the remainder of the work, *collapse* will be considered complete collapse, while the dimensional collapse will be specifically noted.

In this thesis, our proposed SSL approach maximizes a special form of mutual information between two different versions of the same input. This inherently provides solutions to SSL problems along with two key features. First, it increases the similarity between transformed versions of the same input. Second, it prevents collapse by forcing latent features to have non-degenerate distributions.

Although Shannon Mutual Information (SMI) is the first choice to use while maximizing information [Cover and Thomas, 2006], fundamental limitations in the precise calculation of SMI cause difficulties in the training of the SSL model [Poole et al., 2019, Tschannen et al., 2019, McAllester and Stratos, 2020]. We need many more samples to get a reliable SMI estimate, especially as the amount of mutual information is large [McAllester and Stratos, 2020]. Using large batch sizes may not be a viable solution to improve precision, as (i) it can have a negative impact on generalization performance and (ii) it increases memory requirements. Moreover, the estimation and optimization of SMI can significantly increase the computational

load. Finally, we suspect that the partitioning of the hidden space obtained may not be optimal as an input to a simple linear classifier, as SMI maximization can induce a non-linear dependence between representations belonging to the same input.

To overcome these problems, we can substitute second-order statistics based on a special form of mutual information measure, called log-determinant mutual information (LDMI) [Zhanghao Zhouyin and Ding Liu, 2021, Erdogan, 2022], instead of SMI. The joint entropy measure corresponding to LDMI is the log-determinant of the covariance matrix, whose diagonal elements are perturbed. If this perturbation is removed for Gaussian distributed vectors, LDMI boils down to the Shannon differential entropy. As an important feature, LDMI reflects linear dependence in a computationally efficient way.

In this thesis, we propose a new SSL approach derived from the LDMI measure, called CorInfoMax, correlative information maximization. The loss function of CorInfoMax is a first-order approximation of the LDMI and constrains the linear dependence to the identity map. The optimization objective is formed as a loss function based on the Euclidean distance regularized between the representations of different augmented versions of the same input by the log-determinant of the covariance matrix of the hidden vectors. This regularization term promotes the spreading of feature vectors across the whole feature space and acts as a natural barrier against dimensional and total collapses in that space. Our empirical results confirm that CorInfoMax learns effective representations that perform well on downstream tasks. The results of this thesis are submitted as [Ozsoy et al., 2022].

1.1 Major contributions

CorInfoMax, our novel SSL method, provides the following contributions.

- We propose a novel SSL method, CorInfoMax, which has information-theoretical grounds. The loss of CorInfoMax has two explicit components: getting closer to positive samples by minimizing the variance of them ('the attraction factor') and avoiding dimensional collapse by taking advantage of using full feature space ('big-bang factor'). It has an explainable working principle.

- CorInfoMax is a computationally efficient approximation of LDMI. CorInfoMax simplifies LDMI by using an identity mapping instead of a generic linear mapping. This approach directly minimizes the representation variance, as desired for self-supervised learning.

- CorInfoMax relies only on second-order statistics, which is convenient for implementation with low computational complexity.

- CorInfoMax does not require negative samples and does not force representations to be uncorrelated.

- CorInfoMax method achieves state-of-the-art or competitive results in downstream tasks.

1.2 Outline

This paper has been divided into the following parts.

- Chapter 2 gives a review of the literature related to CorInfoMax with a supplementary background.

- Chapter 3 introduces the LDMI measure, and we derive the objective function of CorInfoMax as a variation on the LDMI measure in the same chapter.

- Chapter 4 describes the SSL setting with CorInfoMax.

- Chapter 5 provides the experiments and the results.

- Chapter 6 includes a conclusion with future research directions.

Chapter 2

RELATED WORK

In this chapter, we review self-supervised learning methods, information maximization methods in self-supervised learning settings, and determinant maximization methods. In this thesis, we focus mostly on methods in the computer vision domain, but our method is applicable in different domains. Therefore, the scope of literature review mainly focuses on computer vision related methods.

2.1 Self-supervised learning methods

In [Becker and Hinton, 1992], it is introduced that maximizing the agreement between the outputs of different parts of the same input enables learning useful representations by preventing these outputs from being stuck at a constant value for all inputs. Recent self-supervised learning methods for computer vision are also based on a similar idea. The basic architecture is illustrated in Figure 2.1. If we denote t_i and t_j as sampled transformations from the set of T, we can represent the corresponding transformed versions from the input x as $x_i = t_i(x)$ and $x_j = t_j(x)$. If we feed transformed inputs x_i and x_j into the encoder f_w, the representations obtained will be $y_i = f_w(x_i)$ and $y_j = f_w(x_j)$ in the latent space. The main objective will be to minimize $\mathbb{E}[\text{dist}(y_i, y_j)]$ by changing encoder f_w parameters. When the distance between the output representations is minimized directly in this architecture, the optimal output values will usually be the same for all inputs, which is called collapse [Jing et al., 2021]. Since collapse is the main problem of this type of architecture; many different methods have been proposed to prevent collapse with additional architectures, different losses, and regularizations applied on this base method.

Siamese networks, introduced in [Bromley et al., 1993], contain two networks with different inputs, each of which forms a branch. When these branches converge

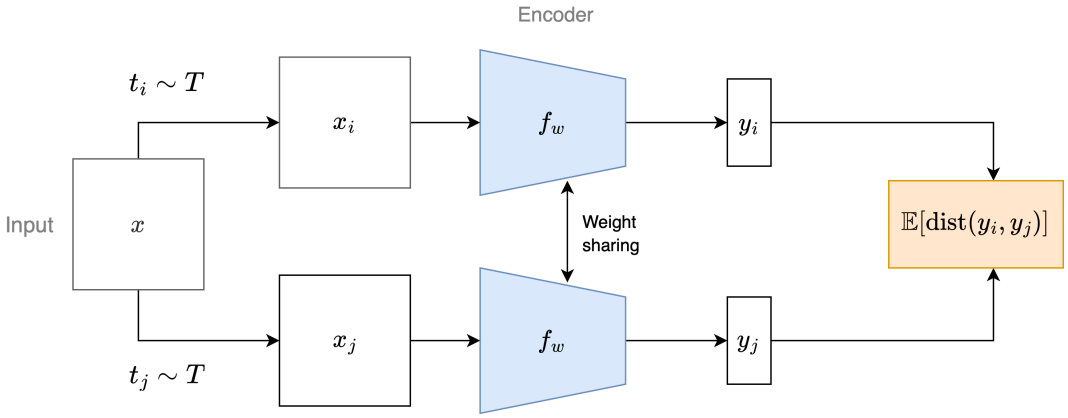


Figure 2.1: Basic SSL setup in Siamese networks architecture. t_i and t_j as sampled transformations from the set of T. x_i and x_j are transformed versions of input x. y_i and y_j are output representations of encoder f_w. "dist" stands for distance.

at the endpoint, the metric to be compared is calculated as in Figure 2.1. These subnetworks have the same weights in [Bromley et al., 1993, Koch et al., 2015, Chen et al., 2020a, Zbontar et al., 2021]. However, some approaches are defined as Siamese networks in which one branch has an additional MLP [Chen and He, 2021] or even the weights of the networks being tied without being completely equal [Grill et al., 2020]. This type of weight-sharing is termed "indirect" [Chen and He, 2021].

2.1.1 Pretext tasks-based methods

In pretext tasks, some information related to the input is changed or hidden by applying transformations over the input. This is a subtask for which the network predicts the hidden or changed property. In addition, there is also a type of pretext tasks that the network discriminates instances by similarity. The features learned during all these pretext tasks are useful for downstream tasks.

In Egomotion approach [Agrawal et al., 2015], Siamese networks use two consecutive image frames of the video data as input. The outputs of two networks are concatenated and fed into the third network to predict the transformation between the inputs. If inputs are collected from a mobile agent, the egomotion information is equal to the motion of the camera, i.e. odometry data. The network predicts the

transformation of the camera in six dimensions as a classification task.

In context prediction [Doersch et al., 2015], predicting the relative position of the image patches extracted from each sample image provides a way to learn the context of that sample image. As a classification problem, two patches are fed into convolutional encoders with shared weights, and the model classifies the correct position of the second patch among eight possible locations around the first patch. To prevent trivial solutions, patches are extracted with a gap between them, and color jittering is applied.

In the colorization task [Zhang et al., 2016], grayscale version of the input image is fed into the network. The network produces a colored version from a grayscale image. Instead of using Euclidean loss between the original input image and predicted image, the color space is quantized to prevent grayish results caused by averaging. With this modification, the task becomes a classification.

The context encoder [Pathak et al., 2016] reconstructs the missing region of the input data using the encoder-decoder architecture with the L2-distance-based reconstruction loss and adversarial loss together. The encoder network encapsulates the context information of the input to the hidden representation, and the following decoder creates the large missing part by using this hidden representation. This architecture forces encoder networks to learn semantic information rather than rely on local pixels around the missing part.

In the rotation-based pretext task [Gidaris et al., 2018], the input images are rotated by a set of degrees, such as [0°, 90°, 180°, 270°]. Each image rotated at each of those predefined degrees is used as input for the network. This network predicts the degree of rotation of each rotated image as a classification task. To correctly recognize the degree of rotation, the network learns useful representations.

JigSaw [Noroozi and Favaro, 2016] creates puzzles from training images by creating tiles as a 3x3 grid. Randomly permuted tiles are fed into the network. The network learns to predict the correct permutations and can reconstruct the 3x3 grid image by rearranging the tiles according to the permutation found.

Exemplar-CNN [Dosovitskiy et al., 2014] is an initial method using instance discrimination as a pretext task. A number of patches are subsampled from input

images in the training dataset as an initial step. A list of random transformations is applied to each patch, and a number of transformed versions of each patch build a new group called the surrogate class. The network is trained to discriminate these surrogate classes with the softmax classifier. The goal of the network is to find out whether the transformed patches are sourced from the same image. This target leads to learning features that are invariant to transformations while discriminating inputs. Although complex pretext tasks were more popular in some period, instance discrimination as pretext task became common with its use in combination with contrastive loss.

2.1.2 Contrastive learning-based methods

Contrastive learning methods are based on dynamics that bring transformed views of the same input closer in the embedding space while taking all other views farther from this input.

Contrastive-based losses

Contrastive loss [Chopra et al., 2005] is introduced as a similarity metric for a face verification problem. Follow-up research [Hadsell et al., 2006] shows that contrastive loss can be used to reduce dimension by learning the mapping function from the input space to the latent space. In both works, similar input vectors in the high-dimensional input space are expected to be close in the low-dimensional latent space. On the contrary, dissimilar input vectors should be distant from each other. Since only minimizing the distance between similar pairs causes a collapse, maximizing the distance between dissimilar pairs is used to prevent that collapse. As a result, the model learns invariant mapping by attracting similar pairs and repulsion of dissimilar pairs. The loss function is minimized by training with the Siamese network architecture [Bromley et al., 1993]; The input pairs of the training set are fed into the same two networks with the same weights as in Figure 2.2.

x_i and x_j are pairs of input images. Label $y = 0$ when x_i and x_j are similar images, and label $y = 1$ when x_i and x_j are dissimilar images. f_w is convolutional

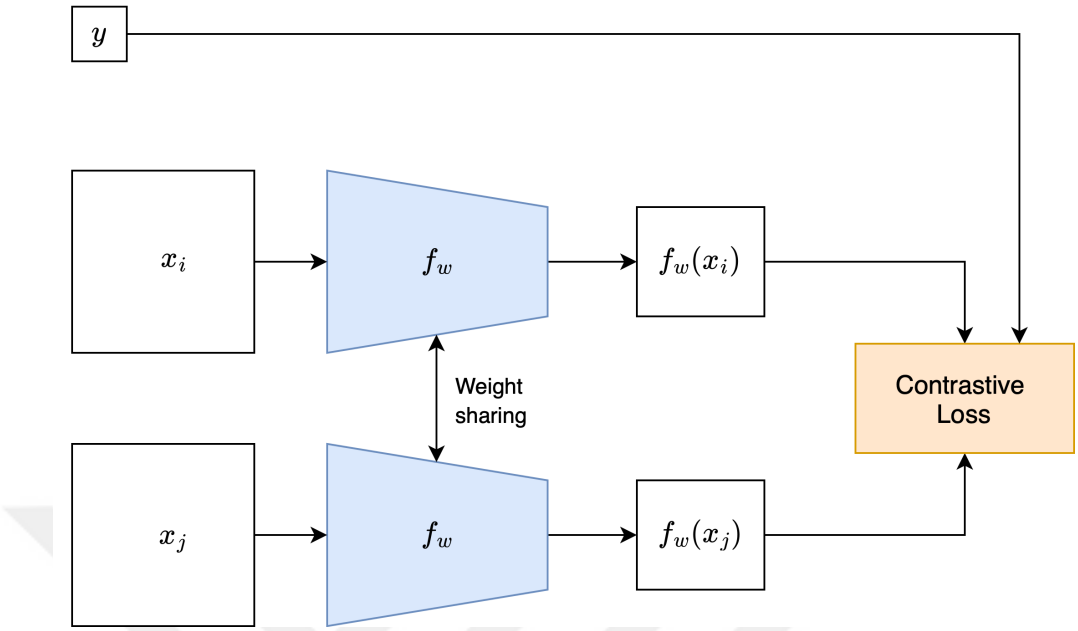


Figure 2.2: Contrastive loss in the Siamese architecture for the face verification problem in [Chopra et al., 2005]. x_i and x_j are pairs of input images. Label $y = 0$ when x_i and x_j are similar images, label $y = 1$ when x_i and x_j are dissimilar images. f_w is convolutional neural network and contrastive loss is provided in (2.1). The figure is adapted from [Chopra et al., 2005].

network, and the contrastive loss is

$$L_{con}(x_i, x_j; \theta) = (1-y)\|f_\theta(x_i) - f_\theta(x_j)\|_2^2 + (y)\max(0, m - \|f_\theta(x_i) - f_\theta(x_j)\|_2)^2 \quad (2.1)$$

where m is the radius distance for samples, called the margin parameter, which decides whether to include dissimilar pairs in the loss calculation.

Triplet loss [Schroff et al., 2015] introduced a new term as the anchor, and this anchor is also compared to an example with the same class (positive example) and an example with a different class (negative example) as in Figure 2.3. By the loss function, the positive example moves closer to the anchor, whereas the negative example moves away from the anchor as in Figure 2.4. x^a is anchor example, x^p is the same class example, x^n is a different class example. Triplet loss function is

$$L_{trip}(x^a, x^p, x^n; \theta) = \max(0, \|f_\theta(x^a) - f_\theta(x^p)\|_2^2 - \|f_\theta(x^a) - f_\theta(x^n)\|_2^2 + m) \quad (2.2)$$

where m is the distance margin that separates negative and positive pairs.

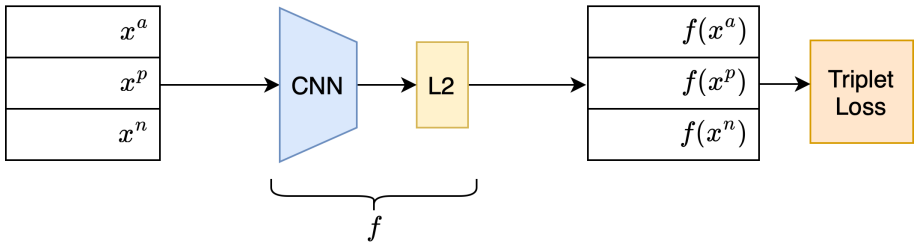


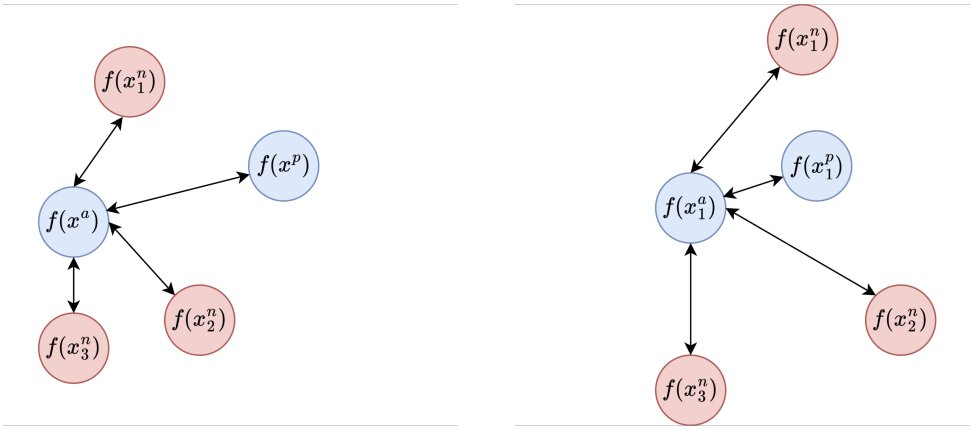
Figure 2.3: Triplet loss in FaceNet architecture [Schroff et al., 2015]. x^a is anchor example, x^p is the same class example, and x^n is a different class example. Encoded representations are ℓ_2-normalized before the loss calculation. f consists of CNN and ℓ_2-normalization. Triplet loss is provided in (2.2). The figure is adapted from [Schroff et al., 2015].

Figure 2.4: Triplet loss in training: (a) before gradient update (b) after gradient update. $f(x^a)$ is encoded representation of anchor example, $f(x^p)$ is encoded representation of the same class example, and $f(x^n)$ is encoded representation of a different class example. The figure is adapted from [Sohn, 2016].

As illustrated in Figure 2.5, triplet loss is generalized with (N+1)-tuplet loss [Sohn, 2016] which is defined to make a comparison between the anchor and (N-1) negative examples.

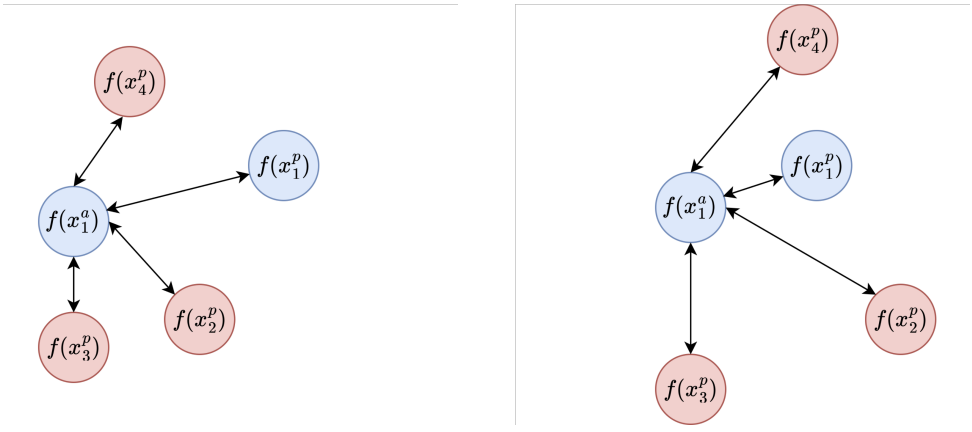
$$L_{(N+1)-tup}(x^a, x^p, \{x_i^n\}_{i=1}^{N-1}; \theta) = \log(1 + \sum_{i=1}^{N-1} \exp(f(x^a)^T f(x_i^n) - f(x^a)^T f(x^p)) \quad (2.3)$$

(N+1)-tuplet loss is impractical for large datasets in terms of the number of classes and examples due to requiring (N+1)M examples to pass in one forward call of the network for a batch with M anchors. Multi-class N-pair loss [Sohn, 2016] decreases it from (N+1)M to 2M with batch construction using positive examples of other pairs as negative examples as in Figure 2.6. According to the results of the



(a) (b)

Figure 2.5: (N+1)-tuplet loss in training: (a) before gradient update (b) after gradient update. $f(x^a)$ is encoded representation of anchor example, $f(x^p)$ is encoded representation of the same class example, and $f(x_i^n)$ is encoded representation of a different class example. (N+1)-tuplet loss is provided in (2.3). The figure is adapted from [Sohn, 2016].



(a) (b)

Figure 2.6: Multiclass N-pair loss in training: (a) before gradient update (b) after gradient update. $f(x_i^a)$ is encoded representation of anchor example, and $f(x_i^p)$ is encoded representation of the same class example. Number of evaluation is decreased with this effective batching strategy. Multiclass N-pair loss is provided in (2.4). The figure is adapted from [Sohn, 2016].

experiment in [Sohn, 2016], Multi-class N-pair loss provides faster convergence than contrastive loss and triplet loss.

$$L_{N-pair}(\{x_i^a, x_i^p\}_{i=1}^N; \theta) = \frac{1}{N}\sum_{i=1}^{N} \log(1 + \sum_{j \neq i} \exp(f(x_i^a)^T f(x_j^p) - f(x_i^a)^T f(x_i^p)) \quad (2.4)$$

Noise-contrastive estimation (NCE) [Gutmann and Hyvärinen, 2010, Gutmann and Hyvärinen, 2012, Mnih and Kavukcuoglu, 2013] is derived from the estimation problem for the parameters of unnormalized statistical models. $X = (x_1, ..., x_{T_d})$ dataset has unknown pdf p_d, which is modeled with $p_m(.; \theta)$. If the integral of pdf of the model is not equal to 1 for all θ, the model is called unnormalized. Although the partition function $Z(\alpha) = \int p_m^0(u; \alpha)du$ helps to obtain the integral of the unnormalized pdf of the model $p_m^0(.; \alpha)$ equal to 1 by $p_m^0(.; \alpha)/Z(\alpha)$, calculation $Z(\alpha)$ is problematic in terms of computational complexity and tractability. As in the proposed solution, the normalization constant is treated as one of the model parameters: $\ln p_m(.; \theta) = \ln p_m^0(.; \alpha) + c$ is defined where $\theta = \{\alpha, c\}$ and the normalization constant $c = \ln 1/Z$. This approach is gathered with the proper objective function which depends on binary classification, whether the sample is from the real dataset or from noise data generated from the chosen distribution.

While $X = (x_1, ..., x_{T_d})$ is observed data, $Y = (y_1, ..., y_{T_n})$ is generated noise samples from distribution $p_n(.)$. The union set of X and Y is $U = (u_1, ..., u_{T_d+T_n})$, $D = 1$ if u_t from dataset X, $D = 0$ if u_t from noise generated Y. Conditional probabilities are

$$p(u|D = 1; \theta) = p_m(u; \theta), \qquad p(u|D = 0) = p_n(u). \quad (2.5)$$

Prior probabilities are

$$p(D = 1) = \frac{T_d}{T_d + T_n}, \qquad p(D = 0) = \frac{T_n}{T_d + T_n}. \quad (2.6)$$

Posterior probabilities are

$$p(D = 1|u; \theta) = \frac{p_m(u; \theta)}{p_m(u; \theta) + kp_n(u)}, \; p(D = 0|u; \theta) = \frac{kp_n(u)}{p_m(u; \theta) + kp_n(u)} \quad (2.7)$$

where k is the ratio of T_n/T_d. If $p(D = 1|u; \theta)$ is denoted as

$$h(u; \theta) = \frac{\ln p_m(u; \theta) - \ln p_n(u)}{1 + k \exp(-u)}, \tag{2.8}$$

whose numerator part is log-ratio between $p_m(u; \theta)$ and $p_n(u)$, the denominator part comes from the parameterized logistic function. NCE loss can be written as

$$L_{NCE}(\theta) = -\frac{1}{T_d} \sum_{t=1}^{T_d} \ln[h(x_t; \theta)] - k \frac{1}{T_n} \sum_{t=1}^{T_n} \ln[1 - h(y_t; \theta)]. \tag{2.9}$$

InfoNCE loss is introduced as part of Contrastive Predictive Coding (CPC) [Oord et al., 2018], which depends on the prediction task of future samples in hidden space with autoregressive models. In addition to audio and text datasets, it can also be applied to image datasets as in Figure 2.7. InfoNCE allows us to compare the target positive example with random negative examples from the data distribution. $X = \{x_1, ...x_N\}$ is a set of N random samples. Single positive example is sampled from distribution $p(x_{t+k}|c_t)$ and $N - 1$ negative examples are sampled from distribution $p(x_{t+k})$, and the loss for InfoNCE is

$$L_{InfoNCE} = -\mathbb{E}_X[\log \frac{f_k(x_{t+k}, c_t)}{\sum_{x_j \in X} f_k(x_j, c_t)}] \tag{2.10}$$

Prediction of future samples x_{t+k} with generative model $p(x_{t+k}|c_t)$ is not targeted with InfoNCE. The target is to model a density ratio f_k as:

$$f_k(x_{t+k}, c_t) \propto \frac{p(x_{t+k}|c_t)}{p(x_{t+k})} \tag{2.11}$$

The logarithmic bilinear model is used for prediction as:

$$f_k(x_{t+k}, c_t) = \exp(z_{t+k}^T W_k c_t), \tag{2.12}$$

that provides a way to learn the distribution of x_t for k with density ratio $f_k(x_{t+k}, c_t)$ and the encoded representations z_{t+k}.

Instance-level discrimination [Wu et al., 2018] considers each particular instance as a new class. In the same work, NCE is used as a nonparametric softmax with a feature memory bank, which is applicable for datasets with a large number of classes. Since training may be unstable due to treating each example as a class, proximal

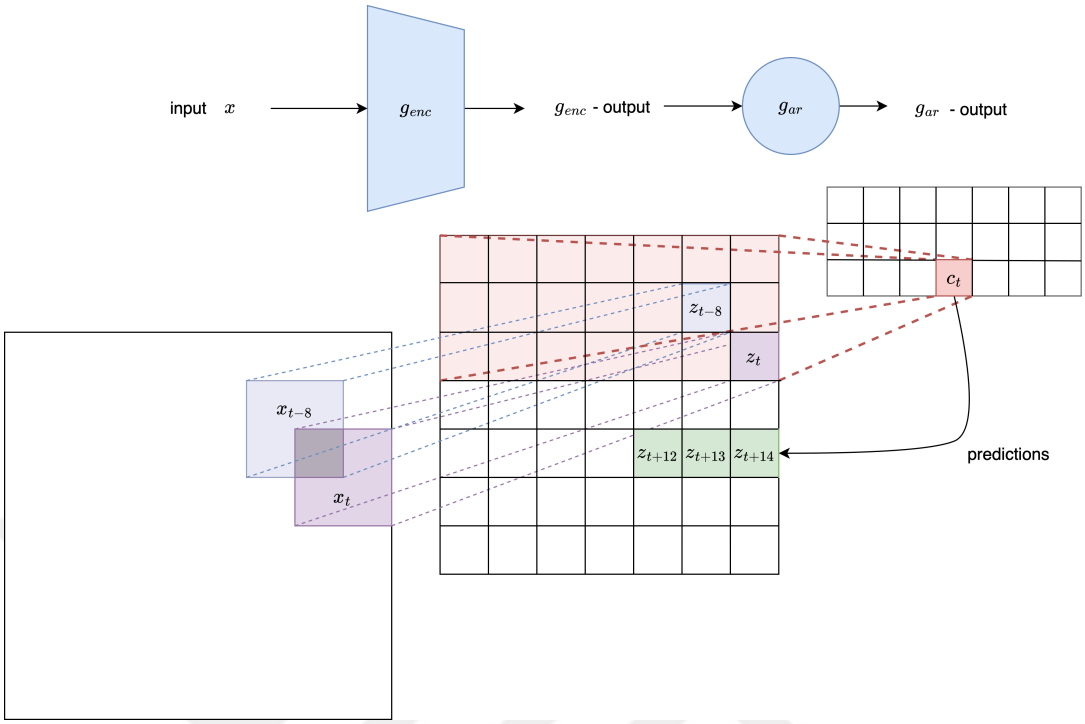


Figure 2.7: CPC for image datasets. x_t is a fixed-size patch in input image x. The output of the encoder g_{enc} is z, and z_t is a fixed-size patch in z. The autoregressive model g_{ar} forms c_t by the first three rows in z, then c_t is used to predict the green patches in z . x_{t+k} is a patch located after x_t in input x. $L_{InfoNCE}$ is provided in (2.10).

regularization is applied to solve this problem. In this setting, input images are $[x_1, ..., x_n]$, and the representations obtained after the encoder are $[v_1, ..., v_n]$. The noise distribution is formalized as uniform $P_n = 1/n$. The number of noise samples is k times more than the number of data samples. The posterior probability for the i^{th} sample having the feature v is

$$h(i, v) := P(D = 1|i, v) = \frac{P(i|v)}{P(i|v) + kP_n(i)} \tag{2.13}$$

For P_d, observed data distribution, feature representation for an image x_i is v. For P_n, feature representation for the randomly sampled other image is v' as a noise sample. Both representations are obtained from the memory bank and the loss is

$$L_{NCE}(\theta) = -E_{P_d}[\log h(i, v)] - mE_{P_n}[\log(1 - h(i, v'))]. \tag{2.14}$$

Proximal regularization is applied using the representation $v^{(t-1)}$ from the previous

iteration $t-1$ to provide smoothness during optimization:

$$L_{NCE}(\theta) = -E_{P_d}[\log h(i, v_i^{(t-1)}) - \lambda \|v_i^{(t)} - v_i^{(t-1)}\|_2^2] - mE_{P_n}[\log(1 - h(i, v'^{(t-1)}))]. \quad (2.15)$$

Contrastive methods

In general, contrastive methods use a loss function that makes the embeddings of dissimilar images diverge. Therefore, they need to have a large number of dissimilar samples and fulfill this requirement by using the memory bank or large batch sizes. This can lead to a memory bottleneck during implementation of these methods.

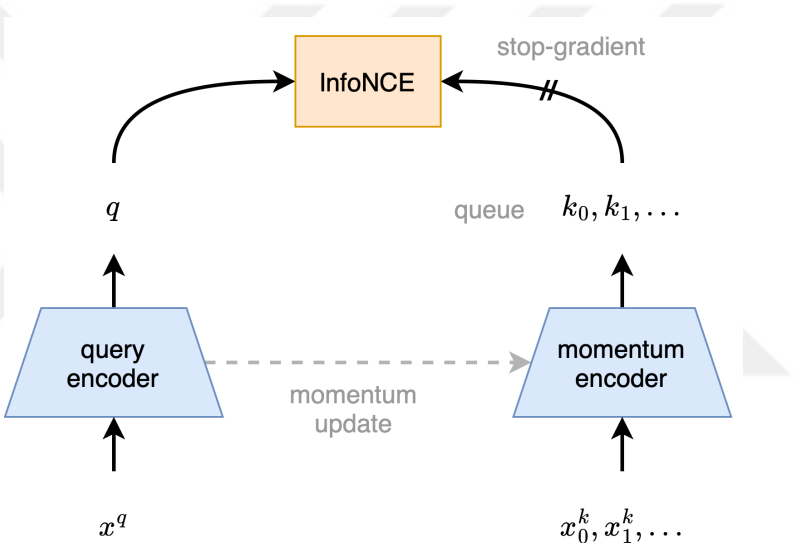


Figure 2.8: MoCo architecture. x_q is query sample, and x_i^k are key samples. The queue is constructed with encoded representations of queue samples. The gradient flows only through the query encoder. The weights of the momentum encoder are updated according to the moving average of the query encoder weights. The figure is adapted from [He et al., 2020].

Momentum Contrastive (MoCo) [He et al., 2020] is based on resemblance between contrastive learning and creating a dynamic dictionary as illustrated in Figure 2.8. According to this perspective, the sampled input images from the dataset, the key samples, are fed into an encoder network called a momentum encoder. The output representation of the momentum encoder is one of the keys in the dictionary. The dictionary is maintained as a queue that is updated after each iteration. The

momentum encoder learns dictionary look-up operation by unsupervised training. In a parallel branch, another version of the same minibatch of input data, the query samples, enters into another encoder network called a query encoder. The encoded representation, the query, is expected to be almost identical to the corresponding key, which forms positive pairs, and to be different from the others, negative pairs built by representations in the queue. The InfoNCE [Oord et al., 2018] loss is used, and the gradient flows only through the query encoder. The loss function for MoCo is

$$L_{MoCo} = -log\frac{\exp(q \cdot k_+/\tau)}{\sum_{i=1}^{K}\exp(q \cdot k_i/\tau)} \tag{2.16}$$

where k_+ is the positive key that matches query q. The denominator part in 2.16 uses this positive key and K negative keys from the queue. The temperature parameter τ controls the amount of loss for hard negative samples [Wang and Liu, 2021]. Hard negative samples are negative samples that have very similar representations to the query. The weights of the momentum encoder are updated according to the moving average of the query encoder weights. The output representations of the momentum encoder for the current batch are added to the queue, whereas the oldest ones are removed from the queue. This approach eliminates the need to use large batch sizes by storing representations in the queue.

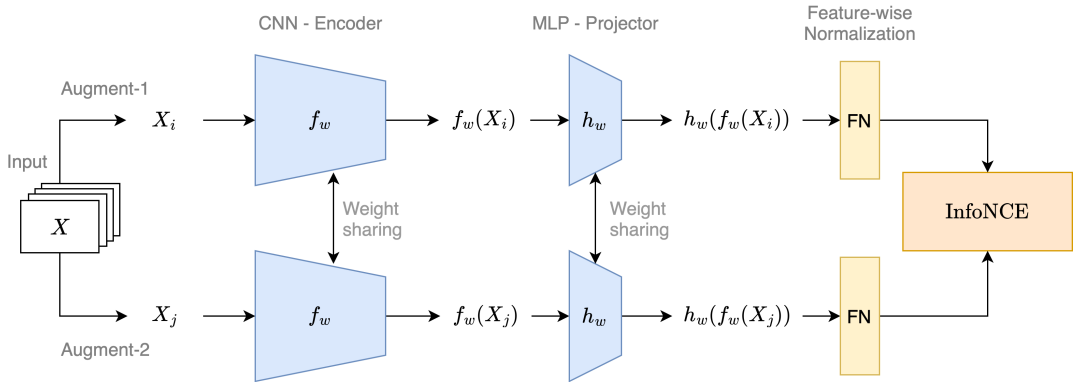


Figure 2.9: SimCLR architecture. X is a batch of input images. X_i and X_j are two different augmented versions of X. Encoder f_w and projector h_w share their weights between two branches. Projector output representations are ℓ_2-normalized before loss calculation.

SimCLR [Chen et al., 2020a] uses InfoNCE [Oord et al., 2018] with on a Siamese

network architecture as in Figure 2.9. X is a batch of input images, X_i and X_j are two different augmented views of this image batch, and they are fed into the same encoder f_w. Encoded representations enter the projector MLP h_w, and output representations are ℓ_2-normalized before loss calculation. Although loss calculations depend on projector output $h_w(.)$, outputs of encoder $f_w(.)$ are used in downstream tasks. Although SimCLR does not provide novel solutions or architecture, a significant performance gain comes from combination of effective random augmentations for the same input, contrastive loss calculation after the nonlinear projection head instead of the encoder, and larger batch sizes. Negative samples are taken from the running batch of data. Therefore, a larger batch size means better performance. The loss function for SimCLR is

$$l_{i,j}^{\text{SimCLR}} = -\log \frac{\exp(sim(z_i, z_j)/\tau)}{\sum_{k=1}^{2N} \mathbb{1}_{[k \neq i]} \exp(sim(z_i, z_k)/\tau)} \tag{2.17}$$

where τ is the temperature hyperparameter. Similarity $sim(\cdot)$ is defined as:

$$sim(z_i, z_j) = \frac{z_i^T z_j}{\|z_i\| \|z_j\|} \tag{2.18}$$

where $z_i = h_w(f_w(X_i))$, and $z_j = h_w(f_w(X_j))$. MoCo v2 [Chen et al., 2020b] is an improved version of MoCo by adding an MLP projection head for encoders and extra image augmentations after SimCLR [Chen et al., 2020a].

NNCLR [Dwibedi et al., 2021] generates two randomly transformed versions of the input sample X as X_i and X_j, and feeds them into two branches of the encoder networks f_w and h_w to obtain the encoded and then projected representations as in Figure 2.10. The representations from the second branch $h_w(f_w(X_j))$ are used directly in the contrastive loss calculation. The representations from the first branch $h_w(f_w(X_i))$ are used to find its nearest neighbor from the support set, and the representations of the nearest neighbor founded are used in the contrastive loss calculation. In both cases, the representations are ℓ_2-normalized prior to loss calculation. NNCLR uses InfoNCE, same loss as SimCLR. While MoCo [He et al., 2020] uses a queue to reach more negative samples, NNCLR [Dwibedi et al., 2021] uses a support set, a different type of queue, to find different positive samples by applying nearest neighbor algorithm.

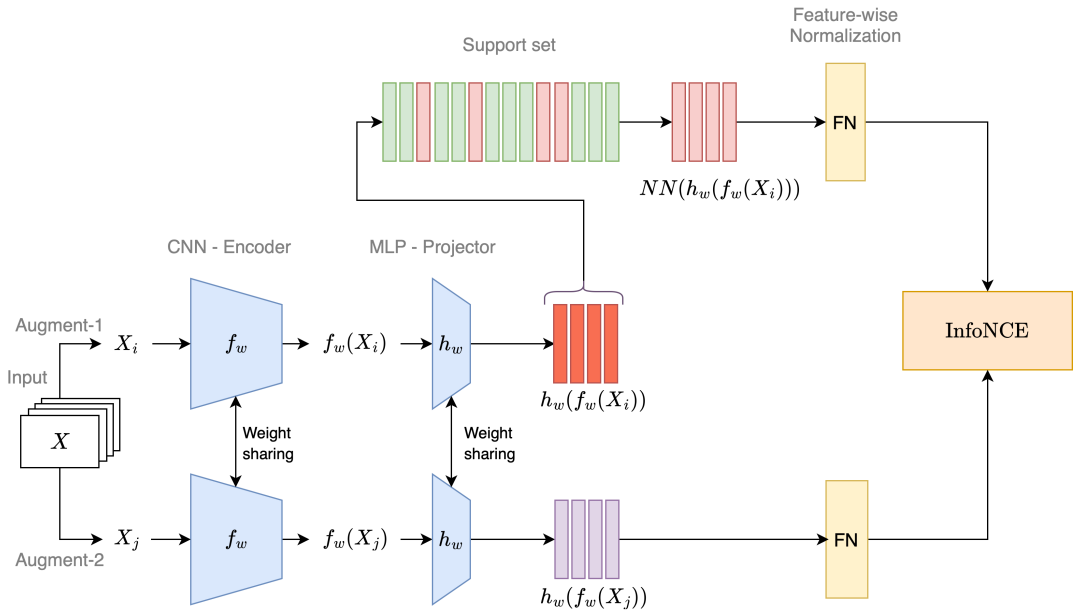


Figure 2.10: NNCLR architecture. X is an input image batch, X_i and X_j are two different augmented versions of X. Encoder f_w and projector h_w share their weights between two branches. $NN(h_w(f_w(X_i)))$ represents the nearest neighbors of $h_w(f_w(X_i))$ in the support set. The representations from two branches are ℓ_2-normalized prior to loss calculation. The figure is adapted from [Dwibedi et al., 2021].

In the spectral contrastive method [HaoChen et al., 2021], a simple form of contrastive loss is utilized based on spectral decomposition and graph representations. Augmented versions of input images are edges in the augmentation graph, each of dataset classes builds a subgraph as in Figure 2.11. Population data for the same classes are continuous. The distance between some samples for each class can be large, but they are actually connected with different augmentations of the same sample and different samples of the same class. The population augmentation graph includes vertices and edges. Vertices are formed by all augmentations, and edges are the connection point of two vertices if the augmented inputs come from the same input. The spectral decomposition is performed by decomposing eigenvectors of the adjacency matrix, which is obtained from the population graph. Parameterization of

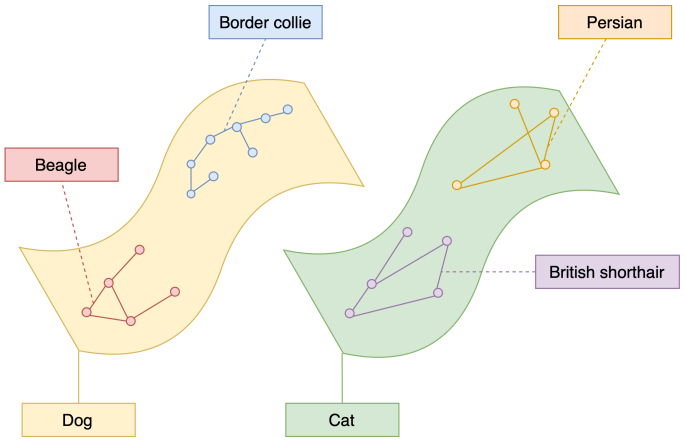


Figure 2.11: Augmentation graph representation for spectral loss. The classes are 'dog' and 'cat' for this example, and the subclasses are different breeds of these. Subgraphs are formed by subclasses. Augmented versions of input images and semantically similar images within subgraphs are linked. Images of different classes are expected to exist on different graphs. The figure is adapted from [HaoChen et al., 2021].

the decomposition provides the spectral loss:

$$L_{spec} = -\frac{2}{N}\sum_{i=1}^{N} z_i^T z_i' + \frac{1}{N(N-1)}\sum_{i\neq j}(z_i^T z_j')^2 \tag{2.19}$$

where $(z_i, z_i^{'})$ is the representation of positive pairs and $(z_i, z_j^{'})$ is the representation of negative pairs. The architecture of spectral contrastive learning is the same as that of SimCLR in Figure 2.9, only the loss part is changed with spectral contrastive loss.

Details related to CPC [Oord et al., 2018], DIM [Hjelm et al., 2018] and AMDIM [Bachman et al., 2019] are given in Section 2.2.

2.1.3 Distillation-based methods

The following two methods are used to learn competitive representations without contrastive learning or clustering. They are teacher-student models, and an asymmetry is created by either architecture or a learning rule. The collapse is avoided in these methods by using stop gradient operator for one branch.

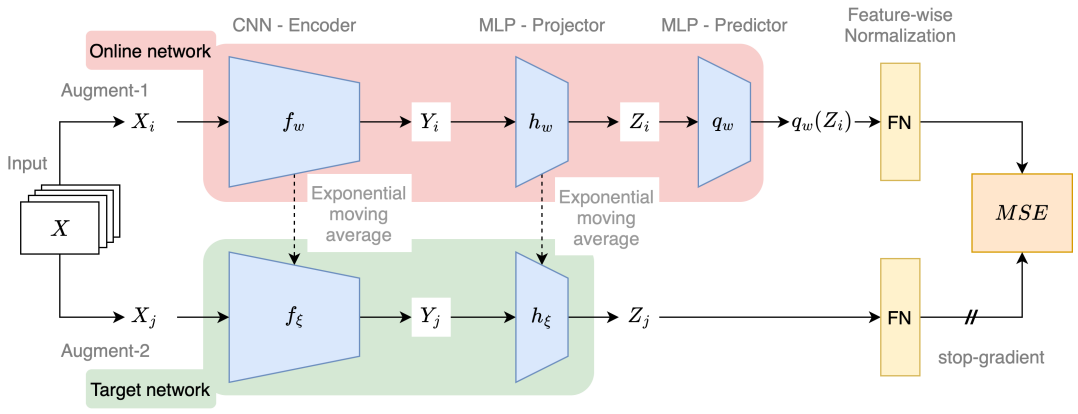


Figure 2.12: BYOL architecture. X is a batch of input images, X_i and X_j are two different augmented versions of X. In the figure, the red part includes the online network, and the green part contains the target network. Due to stop-gradient operation on the target network, only weights of the online network are updated with the loss gradient. The target network is updated with the exponential moving average of the online network weights. Representations from both networks are ℓ_2-normalized prior to loss calculation. The figure is adapted from [Grill et al., 2020].

In BYOL [Grill et al., 2020], two different transformed versions of the input are fed into two different neural networks as Figure 2.12. The first network, called as the online network, predicts the representation of another network, called as the target network. The weights of the online network are changed with gradients resulting from MSE loss between predicted representation and target representation. Stop-gradient operation inhibits backpropagation of gradients for the target network. The weights of the target network are updated according to the moving average of the online network and do not have gradients directly from loss. Therefore, the target network is considered the momentum encoder as in MoCo [He et al., 2020]. BYOL differs from contrastive methods [Chen et al., 2020a, He et al., 2020] because it does not have a repelling term for negative pairs. While MoCo [He et al., 2020] uses a momentum encoder to have consistent representations for negative pairs in the dictionary, BYOL uses it to stabilize target representations. Representations of negative pairs in the dictionary do not change instantly and dramatically via the smoothing effect of the momentum encoder in MoCo [He et al., 2020], the same effect provides stabilization of target representation in BYOL [Grill et al., 2020]. Performance degradation in

linear evaluation is shown in ablation studies [Grill et al., 2020] and [He et al., 2020] when using a decay rate different from the optimal decay rate for the moving average. Although momentum encoder was reported as the solution to avoid collapse for BYOL [Grill et al., 2020], it is later explained in [Chen et al., 2020a] that the key point for preventing collapse is the stop-gradient operation in the target network. The loss is represented in Figure 2.12 as:

$$L_{w,\xi} = \| \frac{q_w(Z_i)}{\|q_w(Z_i)\|_2} - \frac{Z_j}{\|Z_j\|_2} \|_2^2 = 2 - 2 \cdot \frac{\langle q_w(Z_i), Z_j \rangle}{\|q_w(Z_i)\|_2 \cdot \|Z_j\|_2} \tag{2.20}$$

where Z_i is the latent representation for the augmented input X_i, and Z_j is the latent representation for the augmented input X_j. The MLP predictor is denoted g_w. Then $\widetilde{L}_{w,\xi}$ is obtained by exchanging the transformed inputs X_i and X_j, which provides the symmetrized version of the loss:

$$L_{w,\xi}^{BYOL} = L_{w,\xi} + \widetilde{L}_{w,\xi} \tag{2.21}$$

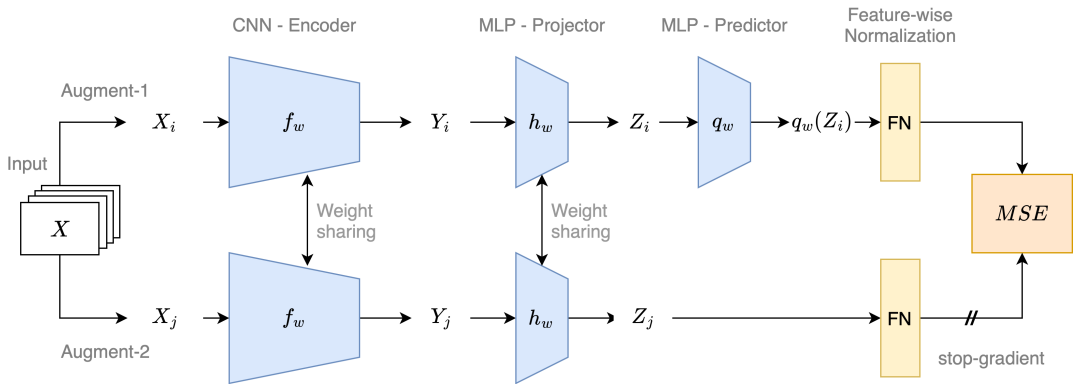


Figure 2.13: SimSiam architecture. X is an input image batch, X_i and X_j are two different augmented versions of X. The encoder f_w and the projector h_w share their weights between two branches, whereas stop-gradient exists in one branch. Representations from both branches are ℓ_2-normalized prior to loss calculation.

SimSiam [Chen and He, 2021] has a similar structure to BYOL [Grill et al., 2020], except that its encoder and projector networks share weights instead of utilizing moving-average. The remaining parts are the same as for BYOL as seen in Figure 2.13. Similarly to BYOL [Grill et al., 2020], the stop-gradient mechanism prevents collapse for SimSiam as well.

2.1.4 Clustering-based methods

These methods require the embeddings from different instances to be part of different groups in the representation space restricted as a unit sphere. The uniform distribution of cluster assignment for instances counteracts collapse.

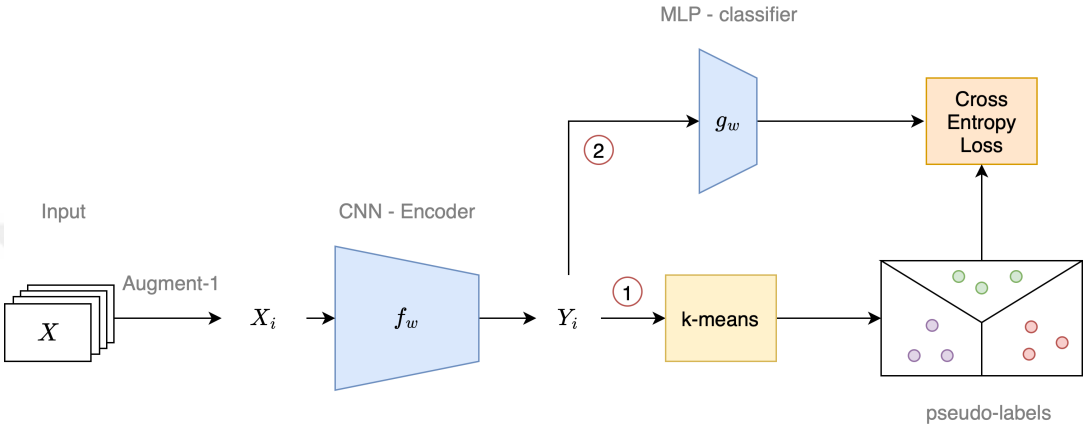


Figure 2.14: DeepCluster architecture. X is batch of input images, X_i is augmented version of X. First, encoded representations Y_i are clustered using the k-means algorithm. Second, the weights of f_w and g_w are updated by classification of pseudo-labels obtained from k-means. The figure is adapted from [Caron et al., 2018].

DeepCluster [Caron et al., 2018] uses k-means clustering on embeddings to create surrogate labels for representation learning. Iteratively, the encoder network produces latent representations, then produces surrogate labels by clustering these representations, and learns features by making classification. The architecture is illustrated in Figure 2.14. Pre-processing for clustering of k-means includes reducing the dimension with PCA, whitening, and $l2$-normalization. Collapse caused by empty clusters can be prevented by assigning a new centroid in the neighborhood of any non-empty cluster for the empty cluster. Collapse due to class imbalance is solved by sampling inputs based on a uniform distribution over surrogate labels.

Since clusters are refined in each epoch in DeepCluster, clustering for large datasets is not scalable. SwAV [Caron et al., 2020] performs online clustering with the Siamese network architecture for a scalable solution as illustrated in Figure 2.15. There are two stages in SwAV: cluster assignment and assignment prediction. SwAV

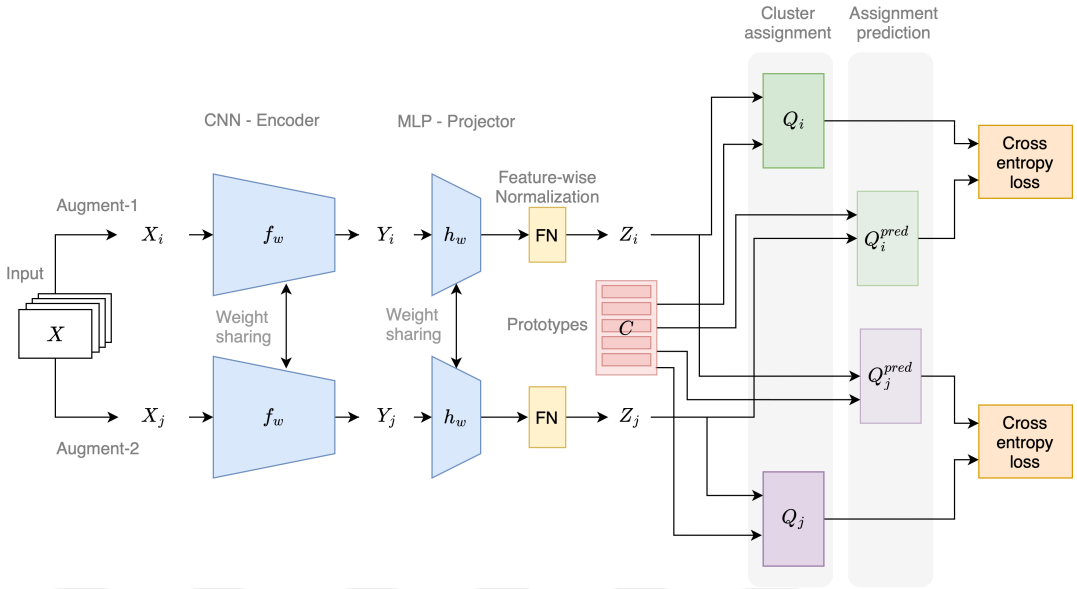


Figure 2.15: SwAV architecture. X is a batch of input images, X_i and X_j are two different augmented versions of X. C is a set of vectors, called prototypes. The code Q_i is obtained with the assignment of Z_i to the corresponding vectors in C. Q_i^{pred} is the prediction of Q_i. Q_j^{pred} is the prediction of Q_j. The figure is adapted from [Caron et al., 2020].

uses prototypes $C = [c_1, .., c_K]$ for clustering the feature representations Z. The code Q, the mapping between C and Z, is optimized so that prototypes C and feature representations Z are close to each other:

$$\max_{Q} \qquad \mathrm{tr}(Q^T C^T Z) + \epsilon H(Q) \tag{2.22}$$

where $H(Q) = -\sum_{ij} Q_{ij} \log Q_{ij}$. The code Q is obtained with the assignment of Z to the corresponding vectors in C:

$$Q^* = \mathrm{Diag}(u) \exp(\frac{C^T Z}{\epsilon}) \mathrm{Diag}(v) \tag{2.23}$$

where u and v are vectors for renormalization computed by Sinkhorn-Knopp algorithm [Cuturi, 2013], which is used to perform this assignment online. While code Q_i is obtained with the assignment of Z_i to the corresponding vectors in C, code Q_j is obtained with the assignment of Z_j to the corresponding vectors in C. Then the

prediction of q_j is made using z_i with loss function

$$l(z_i, q_j) = -\sum_k q^j \log \frac{\exp(z_i^T c_k/\tau)}{\sum_{k'} \exp(z_i^T c_{k'}/\tau)} \tag{2.24}$$

where τ is temperature coefficient.The prediction of q_i is also made using z_j. Since the loss is defined for one sample, it can be extended to all images in the batch. Prototypes and networks are updated with gradients of this loss.

2.1.5 Regularization-based methods

Another major strategy to prevent collapse is the decorrelation of pairs of hidden representations. This also indirectly brings about information maximization on latent representations. The whitening approaches used in these methods aim at isotropic spread of information within the feature space, which prevents dimensional collapse, illustrated in Figure 1.1b, as well as collapse, illustrated in Figure 1.1a.

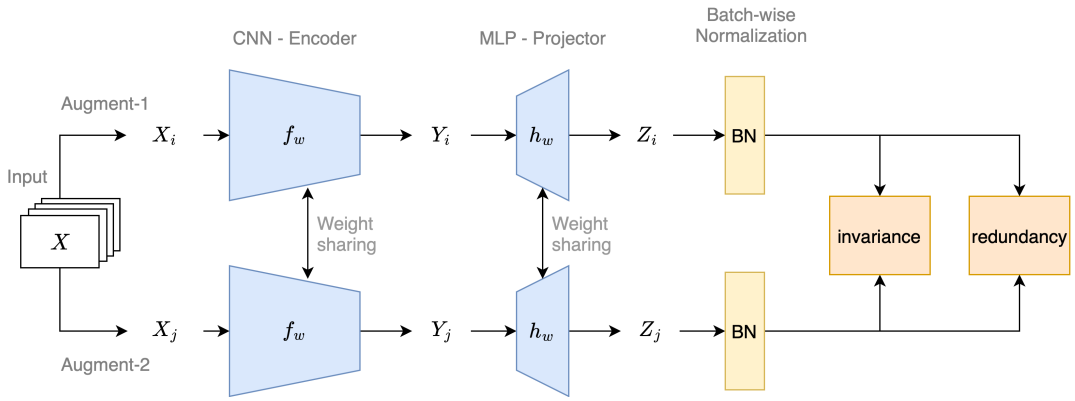


Figure 2.16: Barlow Twins architecture. X is a batch of input images, X_i and X_j are two different augmented versions of X. Y_i is output of the encoder f_w, and Z_i is output of the projector h_w. Projector network h_w is in expander form, increasing the dimension of latent representations. Invariance loss and redundancy loss are the first and second terms of L_{BT} in (2.25), respectively. The figure is adapted from [Zbontar et al., 2021].

In Barlow Twins [Zbontar et al., 2021], proposed approach creates a normalized cross-correlation matrix of the two output feature representations. The main objective is the convergence of this cross-correlation matrix with the identity matrix. The

overview of the architecture is illustrated in Figure 2.16. The loss function of Barlow Twins is the following:

$$L_{BT} = \sum_i (1 - C_{ii})^2 + \lambda \sum_i \sum_{j \neq i} C_{ij}^2 \tag{2.25}$$

where i, j are the indices of the cross-correlation matrix C. Due to the use of (i, j) for the indices, we change the notation for batch-normalized Z_i to Z, and batch-normalized Z_j to Z'. Embeddings $Z = [z_1, z_2, .., z_n]$ and $Z' = [z'_1, z'_2, .., z'_n]$ consist of n vectors with feature dimension d. The cross-correlation matrix C with dimensions $d \times d$ is obtained from

$$C_{ij} = \sum_k Z_{k,i} Z'_{k,j} \tag{2.26}$$

where k is the index for batch dimension, i, j are the indices for feature dimension of latent representation Z and Z'.

The first term of L_{BT} is 'invariance term', which provides being invariant to augmentations by imposing diagonal entries of C to be 1. The second term of L_{BT} is 'redundancy term' with coefficient λ, which provides decorrelation of different feature representations by enforcing off-diagonal entries of C to be 0.

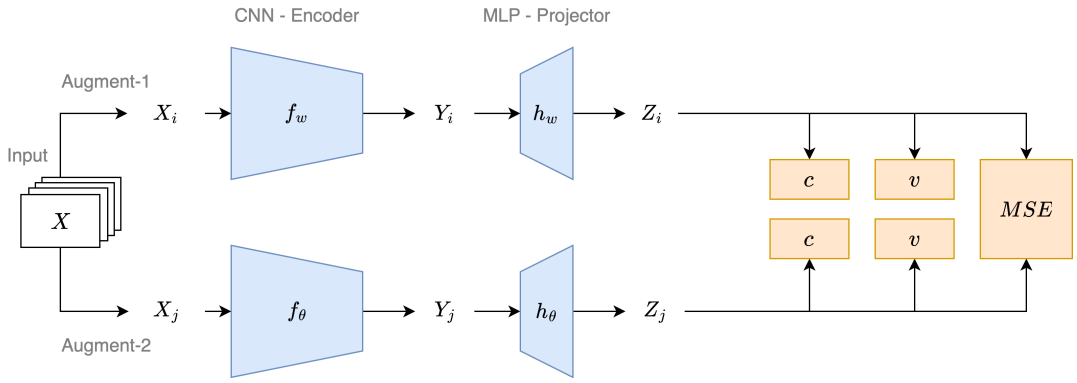


Figure 2.17: VICReg architecture. X is an input image batch, X_i and X_j are two different augmented versions of X. Y_i is output of the encoder f_w, and Z_i is output of the projector h_w. Projector network h_w is in expander form, increasing the dimension of latent representations. Variance term v, covariance term c, and invariance term s (MSE) of the loss function are presented in (2.27), (2.30), and (2.31), respectively. The figure is adapted from [Bardes et al., 2021].

VICReg [Bardes et al., 2021] is a follow-up work to Barlow Twins [Zbontar et al., 2021], based on similar ideas but different loss terms. Similarly as in Barlow Twins, we change the notation for Z_i to Z, and Z_j to $Z^{'}$ due to the use of (i, j) for the indices. Embeddings $Z = [z_1, z_2, .., z_N]$ and $Z^{'} = [z_1^{'}, z_2^{'}, .., z_n^{'}]$ consist of n vectors with feature dimension d. The variance term v is based on the standard deviation of z^j, which represents the vector formed by the values of the feature dimension j in Z

$$v(Z) = \frac{1}{d}\sum_{j=1}^{d} \max(0, \gamma - S(z^j, \epsilon)), \tag{2.27}$$

where $S(x, \epsilon) = \sqrt{\text{Var}(x) + \epsilon}$ is the standard deviation with ϵ regularized, and γ, target value for S, is 1. In this setting, the variance term encourages the standard deviation of the representations along the batch dimension to be 1, preventing the collapsed solution.

The covariance matrix C of the latent representation Z is

$$C(Z) = \frac{1}{n-1}\sum_{i=1}^{n}(z_i - \tilde{z})(z_i - \tilde{z})^T \tag{2.28}$$

where $\tilde{z}$ is the mean along the batch dimension:

$$\tilde{z} = \frac{1}{n}\sum_{i=1}^{n} z_i \tag{2.29}$$

The covariance term is obtained from the off-diagonal entries of $C(Z)$

$$c(Z) = \frac{1}{d}\sum_{i \neq j}[C(Z)]_{i,j}^2 \tag{2.30}$$

where d is the feature dimension of Z. The covariance term imposes off-diagonal entries of the covariance matrix C to approximate 0, which means decorrelation in latent representation.

The invariance term is the MSE loss between latent representations, which brings similar pairs closer together.

$$s(Z, Z^{'}) = \frac{1}{n}\sum_{i} \|z_i - z_i^{'}\|_2^2 \tag{2.31}$$

Whereas $s(.)$ is obtained using both branches, $c(.)$ and $v(.)$ are computed separately for each branch, as in Figure 2.17. All these terms with the weight factors λ,

μ, ν form the loss function of VICReg

$$l(Z, Z^{'}) = \lambda s(Z, Z^{'}) + \mu[v(Z) + v(Z^{'})] + \nu[c(Z) + c(Z^{'})]. \tag{2.32}$$

Although most of the VICReg experiments were carried out in a weight-sharing setting with Siamese network architecture in [Bardes et al., 2021], it is especially emphasized that the architecture also works without weight-sharing. This feature is important to be able to work with multi-modal data, although performance degradation is observed compared to weight-sharing setting in regular tasks [Bardes et al., 2021]. The projector head is an expander that increases the dimension of the latent representations.

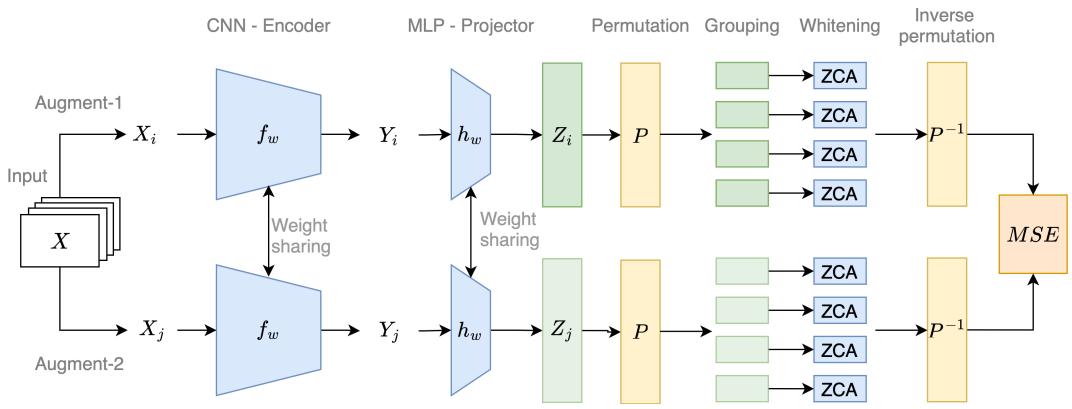


Figure 2.18: Shuffled-DBN architecture. X is a batch of input images, X_i and X_j are two different augmented versions of X. Y_i is output of the encoder f_w, and Z_i is output of the projector h_w in weight sharing setting. The feature indices of Z_i and Z_j are randomly permuted before each grouping operation for a batch. After DBN implementation (grouping and ZCA whitening), feature indices are reordered to the initial version before permutation.

Shuffled-DBN [Hua et al., 2021] is another method that deals with collapse by decorrelating representations. The main solution for collapse is based on Decorrelated Batch Normalization (DBN) [Huang et al., 2018], which has the following procedure: latent representations are split into groups along the feature dimension, then ZCA whitening [Bell and Sejnowski, 1997] is applied to each split. In this shuffled version as in Figure 2.18, feature indices of Z_i and Z_j are randomly permuted before

each grouping operation for a batch. After DBN implementation, feature indices are reordered to the initial version prior to permutation. With these additional permutation operations, Shuffled-DBN [Hua et al., 2021] also prevents dimensional collapse.

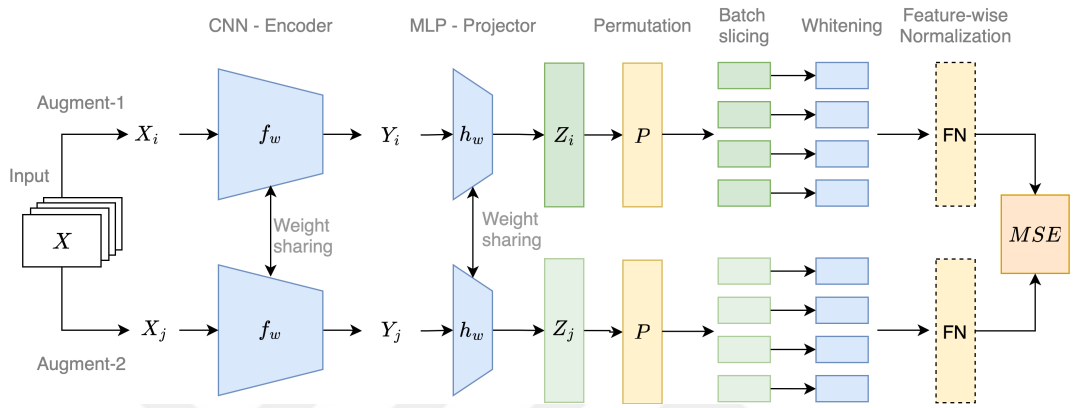


Figure 2.19: W-MSE architecture. X is a batch of input images, X_i and X_j are two different augmented versions of X. Y_i is output of the encoder f_w, and Z_i is output of the projector h_w in weight sharing setting. The feature indices of Z_i and Z_j are randomly permuted before each grouping operation for a batch. After batch slicing and whitening, the representations from both branches are optionally ℓ_2-normalized prior to loss calculation. The figure is adapted from [Ermolov et al., 2021].

Whitening MSE (W-MSE) [Ermolov et al., 2021] uses decorrelation to prevent collapse and learns useful representations as in Figure 2.19. Similarly to the grouping in Shuffled-DBN [Hua et al., 2021], W-MSE [Ermolov et al., 2021] uses batch slicing to split latent representations, but splits along their batch dimension rather than feature dimension. Permutation is also used before grouping. Each batch slice is whitened based on Cholesky decomposition, and the MSE loss is calculated with the embeddings after optional ℓ_2-normalization.

ARB [Zhang et al., 2022] is based on finding the closest orthonormal bases of latent representations and aligns these representations through these bases. The closest orthonormal basis $M(Z)$ to latent representation Z is defined as

$$M(Z) = \min_{B_0} \|Z - B_0\|_2^2 \qquad s.t. \qquad B_0^T B^0 = I \tag{2.33}$$

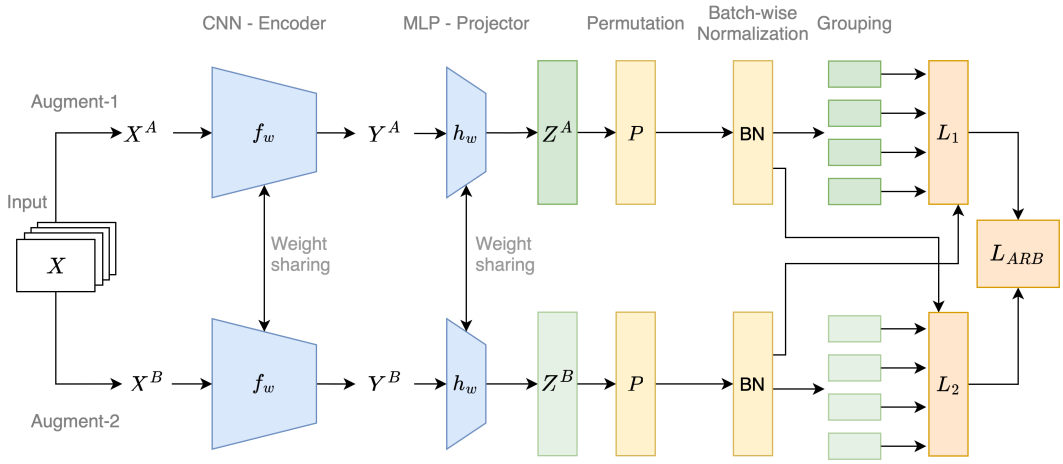


Figure 2.20: ARB architecture. X is an input image batch, X^A and X^B are two different augmented versions of X. Y^A is output of the encoder f_w, and Z^A is output of the projector h_w in a weight sharing setting. The feature indices of Z^A and Z^B are randomly permuted and then batch-normalized before grouping operation. Details of L_{ARB} exist in (2.34). The figure is adapted from [Zhang et al., 2022].

where B_0 is an orthonormal matrix. As in Figure 2.20, X^A and X^B are two transformed versions of input images $X = [x_1, .., x_n]$ as a batch. Z^A and Z^B are latent representations obtained after the encoder and projector networks. ARB aligns these representations to closest orthonormal bases with the loss function

$$L_{ARB} = \text{tr}(\|I - (Z^A)^T B^B)\|_2^2) + \text{tr}(\|I - (Z^B)^T B^A)\|_2^2), \tag{2.34}$$

where $B^A = M(Z^A)$ and $B^B = M(Z^B)$. I is an identity matrix. As in Figure 2.20, the indices of the feature dimension are permutated, then batch-normalized representation is split into groups along feature dimension before applying L_{ARB}.

CorInfoMax approach is related to the decorrelation-based methods discussed above because of correlation-based measures. However, unlike decorrelation-based methods, CorInfoMax does not constrain hidden vectors to be uncorrelated. Instead, it avoids covariance matrix degeneracy by using the log-determinant as a regularizer loss function. Furthermore, the principle of information maximization has been applied more directly and explicitly to the CorInfoMax algorithm.

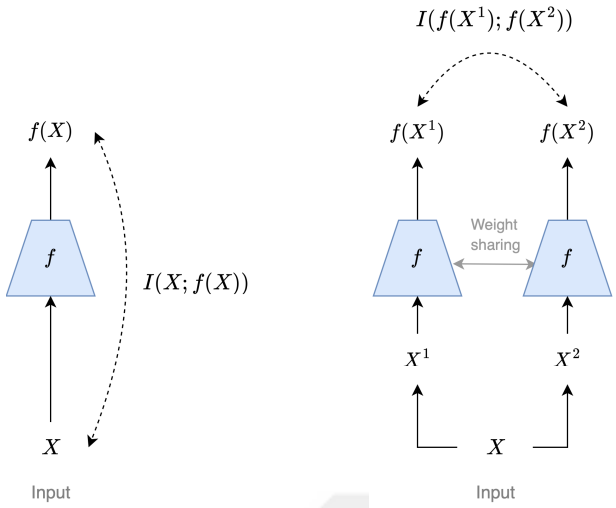


Figure 2.21: Two different approaches for information maximization: (a) maximizing mutual information between input X and the corresponding encoder output $f(X)$ as in Infomax [Linsker, 1988]. (b) maximizing mutual information between encoder output representations $f(X^{(1)})$ and $f(X^{(2)})$ as in [Becker and Hinton, 1992]. $X^{(1)}$ and $X^{(2)}$ are input views, f is an encoder in a weight-sharing setting.

2.2 Information maximization in self-supervised learning

The Infomax principle [Linsker, 1988] was introduced as the preservation of maximum information between the input features and the corresponding output features, based on neuroscience and information theory. As in Figure 2.21a, Infomax objective for input X with an encoder f is defined as

$$\underset{f \in F}{\text{maximize}} \quad I(X; f(X)). \tag{2.35a}$$

Another direction is introduced in [Becker and Hinton, 1992] by maximizing the information between the encoded outputs of different parts of the same input. As in Figure 2.21b, $X^{(1)}$ and $X^{(2)}$ are input views, f is an encoder in a weight-sharing setting (otherwise f_1 and f_2), proposed objective is

$$\underset{f \in F}{\text{maximize}} \quad I(f(X^{(1)}); f(X^{(2)})) \tag{2.36a}$$

For the second, mutual information estimation between encoded features has the advantage relative to (2.35a) that the calculation can be performed in a smaller

dimension than the input.

Calculating the precise mutual information is not feasible because the exact distributions are not accessible. Therefore, estimation of mutual information from the data is required, which is a difficult task in high dimensions [McAllester and Stratos, 2020]. A common approach is to use an estimator which maximizes a lower bound of mutual information, such as in Contrastive Predictive Coding (CPC) [Oord et al., 2018], Deep Infomax (DIM) [Hjelm et al., 2018], and Augmented Multiscale Deep InfoMax (AMDIM) [Bachman et al., 2019]. These are three successively published methods with a similar framework, improving the previous of each in a self-supervised learning setting. In all of them, the critical idea is to maximize mutual information between global and local representations. Local features are encoded representations of only part of the input. The global feature is a summary of local features, which is a representation of full input. InfoNCE [Oord et al., 2018] allows us to compare these representations to maximize mutual information in all these methods.

CPC [Oord et al., 2018] depends on the prediction task of future representations in the hidden space with autoregressive models. The global feature is aggregated from the current local representations iteratively. Future local representations are predicted from current global representations. Mutual information is maximized between the current global feature c_t and local predicted features x_{t+k} for model f_k with

$$\underset{f_k \in F}{\text{maximize}} \quad I(x_{t+k}, c_t), \tag{2.37a}$$

details related to the representations are illustrated in Figure 2.7.

DIM [Hjelm et al., 2018] maximizes mutual information between intermediate feature representations and output feature representation. It is expected that there is higher mutual information between the output representation and all intermediate features if noise-free information is shared between all receptive areas of the input. A feature map is constructed on the basis of local features obtained from the encoder's selected intermediate layer. These local features do not contain the entire area of the input; therefore, they are called local. When this feature map is fed into the

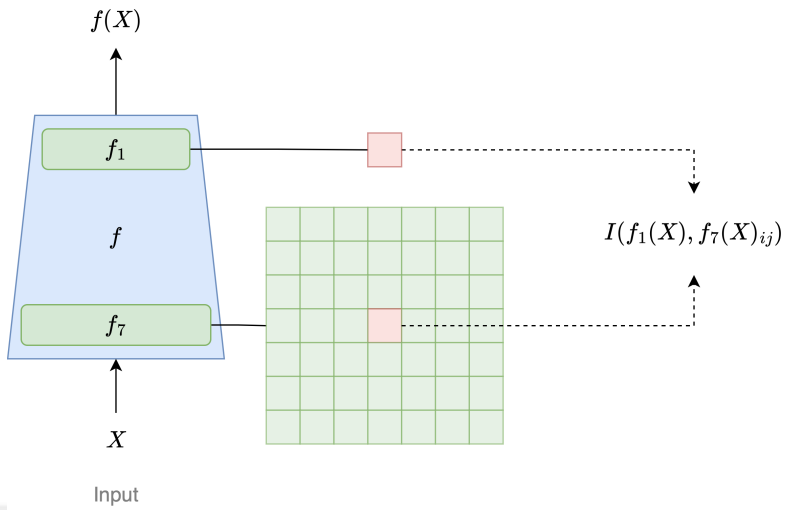


Figure 2.22: DIM architecture. X is input, and f is encoder. f_1 produces a global feature vector, and f_7 is an intermediate layer that produces a local feature map. Square boxes represent the corresponding features. The dashed line indicates mutual information as $I(\cdot, \cdot)$. The figure is adapted from [Hjelm et al., 2018].

remaining part of the encoder, a single output representation of the encoder, the global feature, is obtained as in Figure 2.22. InfoNCE maximizes mutual information between pairs of global and local features. X is input, and f is encoder. f_1 produces a global feature vector, and f_7 is an intermediate layer that produces a local feature map. The proposed objective can be written as follows.

$$\underset{f \in F}{\text{maximize}} \quad I(f_1(X), f_7(X)_{ij}) \tag{2.38a}$$

In AMDIM [Bachman et al., 2019], there are multilevel global features rather than a single global feature. That is why the definition was changed from global-local to antecedent-consequent. Sourcing features from different intermediate layers provides multi-level feature predictions. In addition, these feature predictions are performed across two different views on the same input. $X^{(1)}$ and $X^{(2)}$ are transformed versions of the input X, and f is encoder in a weight-sharing setting. While f_1 produces an antecedent feature, f_7 and f_5 can produce consequent features with indices (k, l). While f_5 produces an antecedent feature with indices (i, j), f_5 and f_7 can produce consequent features with indices (k, l). The proposed objective can be written as

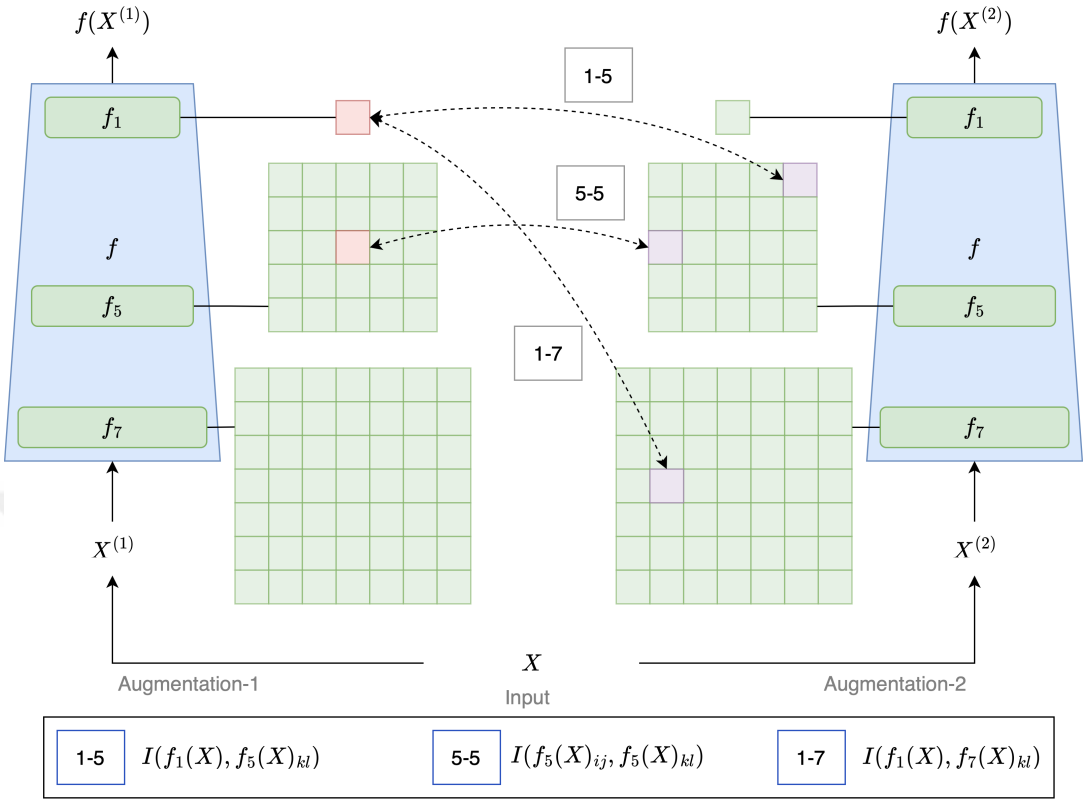


Figure 2.23: AMDIM architecture. $X^{(1)}$ and $X^{(2)}$ are transformed views of input X, and f is encoder in a weight-sharing setting. While f_1 produces an antecedent feature, f_7 and f_5 can produce consequent features with indices (k, l). While f_5 produces an antecedent feature with indices (i, j), f_5 and f_7 can produce consequent features with indices (k, l). Square boxes represent the corresponding features. The dashed line indicates mutual information as $I(\cdot, \cdot)$. The figure is adapted from [Bachman et al., 2019].

follows:

$$\underset{f \in F}{\text{maximize}} \quad I(f_n(X_{ij}^{(1)}), f_m(X^{(2)})_{kl}) \tag{2.39a}$$

where $(m, n) \in [(1, 5), (5, 5), (1, 7)]$. The antecedent feature formed by f_1 does not require indices such as (i, j) and (k, l). The overview of AMDIM architecture and information maximization is illustrated in Figure 2.23.

In all three consecutive methods, the receptive field should not consist of a full image region for the local or antecedent features. Therefore, these methods need layer modifications for standard deep convolutional networks, unlike our method,

CorInfoMax.

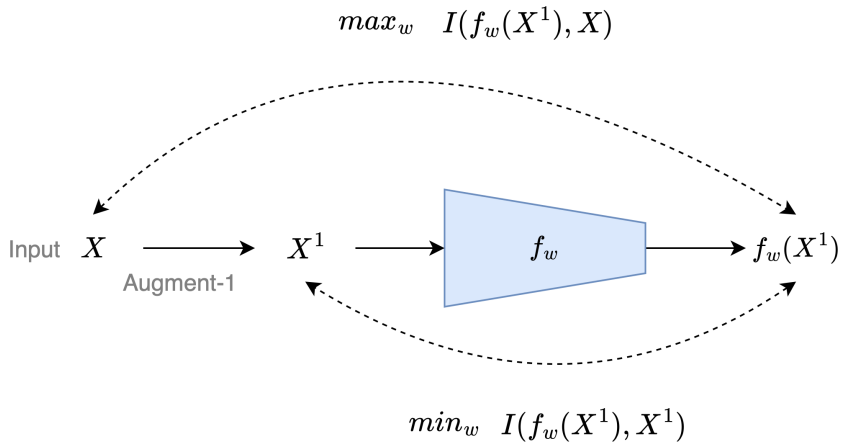


Figure 2.24: Barlow Twins from an information-theoretical perspective. For one branch of Barlow Twins, X^1 is augmented version of input X, $f_w(X^1)$ is output representation of encoder f_w. According to Information Bottleneck [Tishby et al., 2000], output representation $f_w(X^1)$ preserves maximum information about input X while minimizing the information about augmented version of input representation X^1.

.

In an information-theoretical perspective, Barlow Twins [Zbontar et al., 2021] poses its loss function also as an approximation of the Information Bottleneck [Tishby et al., 2000] objective function

$$\mathcal{IB}(w) \triangleq I(f_w(X^1), X^1) - \beta I(f_w(X^1), X), \tag{2.40}$$

where X is the original input, X^1 is the transformed version of the input, and $f_w(X^1)$ is the output of the encoder f_w for X^1. $I(\cdot, \cdot)$ is the SMI, and β is a weighting factor. As illustrated in Figure 2.24, the Barlow Twins approach tries to maximize mutual information between input and output, while reducing redundancy at output by eliminating the information content related to transformations [Zbontar et al., 2021]. In contrast, the CorInfoMax approach is based on maximization of the correlative information between the representations of random transformed versions of the same input.

SSL-HSIC [Li et al., 2021] constructs an SSL loss function using the Hilbert-Schmidt Independence Criterion (HSIC) [Gretton et al., 2005], a kernel-based dependence measure, to maximize the dependence between alternative representations of the same input and the identity representation of this input. Furthermore, this loss minimizes the variances between the kernel matrices of these alternative representations. If we define Z as the encoder-based representation of randomly transformed input images X, Y is the identity representation of these input images. The identity representation is one-hot encoded version of the indices that are distinct for each image in the dataset. f_w is the encoder and the loss function for SSL-HSIC is

$$L_{SSL-HSIC}(w) = -\operatorname{HSIC}(Z,Y) + \beta\sqrt{\operatorname{HSIC}(Z,Z)} \tag{2.41}$$

where β is the weighting factor and the exact calculation of HSIC is

$$\operatorname{HSIC}(Z,Y) = \|\mathbb{E}[k(Z)l(Y)^T] - \mathbb{E}[k(Z)]\mathbb{E}[l(Y)]^T\|_{HS}^2 \tag{2.42}$$

where k is a kernel applied to Z, l is a kernel applied to Y. The Hilbert-Schmidt norm $\|\cdot\|_{HS}$ is taken as the Frobenius norm for finite dimensions. While dependence between the identity representation of an input and alternative latent representations of the same input is maximized by SSL-HSIC [Li et al., 2021], the objective of CorInfoMax mainly differs with the maximization of the dependence between alternative latent representations of the same input.

2.3 Determinant maximization in 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) [Fu et al., 2019, Fu et al., 2016], simplex-structured matrix factorization (SSMF) [Chan et al., 2011], sparse component analysis (SCA) [Babatas and Erdogan, 2018], bounded component analysis (BCA) [Inan and Erdogan, 2014] and polytopic matrix factorization (PMF) [Tatli and Erdogan, 2021].

In the generative models in these frameworks, the input data is assumed to be linear transformations of several latent vectors. It is also assumed that these latent vectors are sufficiently scattered in their feature space. Maximizing the determinant

of the latent covariance matrix spreads the latent vectors to capture assumption on scattering of generative model samples. In the same way, the log-determinant of the latent vector covariance matrix in the CorInfoMax objective spreads latent vectors in their higher-dimensional space. This inherently avoids collapse. In case latent vectors exist in domain dependent to generative model, the determinant maximization criterion used in these matrix factorization frameworks is substantially equivalent to correlative information maximization between input and its latent vectors based on the LDMI objective [Erdogan, 2022].

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.

Chapter 3

LOG-DETERMINANT MUTUAL INFORMATION

In this chapter, we will introduce LDMI with some background on mutual information and entropy. In next, we will derive the objective function of CorInfoMax as a variation on the LDMI.

3.1 Background on information measures

The most natural and conventional means of measuring the uncertainty of a random vector with real values $\mathbf{x}$ is to use the joint Shannon differential entropy defined by [Cover and Thomas, 2006]

$$h(\mathbf{x}) = -\int\limits_{\mathrm{dom}(f_{\mathbf{x}})} \log(f_{\mathbf{x}(x)}) f_{\mathbf{x}(x)} \mathrm{d}x = -E(\log(f_{\mathbf{x}}(\mathbf{x}))). \tag{3.1}$$

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

$$I(\mathbf{x};\mathbf{y}) = h(\mathbf{x}) - h(\mathbf{x}\,|\,\mathbf{y}). \tag{3.2}$$

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 dependence between its arguments.

For a given r-dimensional random vector $\mathbf{x}$ with PDF $f_{\mathbf{x}}(\mathbf{x})$, log-determinant (LD) entropy is defined as [Zhouyin and Liu, 2021][Erdogan, 2022]

$$H_{LD}^{(\varepsilon)}(\mathbf{x}) = \frac{1}{2}\log\det(\mathbf{R}_{\mathbf{x}} + \varepsilon\mathbf{I}) + \frac{r}{2}\log(2\pi e). \tag{3.3}$$

where $\mathbf{R}_{\mathbf{x}}$ is the auto-covariance matrix of $\mathbf{x}$, and ε is a small nonnegative parameter defined for the diagonal perturbation in $\mathbf{R}_{\mathbf{x}}$. We note that $H_{LD}^{(\varepsilon)}(\mathbf{x})$ is equivalent to Shannon entropy when $\mathbf{x}$ is a Gaussian vector with covariance $\mathbf{R}_{\mathbf{x}}$, and ε is equal

to zero. However, we should underline that it is a standalone uncertainty measure solely based on 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 [Kailath et al., 2000], we can write

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

where we defined

$$H_{LD}^{(\varepsilon)}(\mathbf{x}|\mathbf{y}) = \frac{1}{2}\log\det(\mathbf{R}_{\mathbf{x}} - \mathbf{R}_{\mathbf{xy}}(\mathbf{R}_{\mathbf{y}} + \varepsilon\mathbf{I})^{-1}\mathbf{R}_{\mathbf{xy}}^T + \varepsilon\mathbf{I}) + \frac{r}{2}\log(2\pi e), \tag{3.4}$$

as the conditional LD-entropy.

We note that the argument of the log det function in (3.4) 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 [Kailath et al., 2000]). More precisely, since $\mu_{\mathbf{x}}$ 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 (3.4) 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. [Ozsoy et al., 2022]

The LD-mutual information (LDMI) measure is defined based on (3.3) and (3.4) as follows:

$$\begin{aligned} I_{LD}^{(\varepsilon)}(\mathbf{x};\mathbf{y}) &= H_{LD}^{(\varepsilon)}(\mathbf{x}) - H_{LD}^{(\varepsilon)}(\mathbf{x}\,|\,\mathbf{y}) \\ &= \frac{1}{2}\log\det(\mathbf{R}_{\mathbf{x}} + \varepsilon\mathbf{I}) \\ &\quad -\frac{1}{2}\log\det(\mathbf{R}_{\mathbf{x}} - \mathbf{R}_{\mathbf{xy}}(\mathbf{R}_{\mathbf{y}} + \varepsilon\mathbf{I})^{-1}\mathbf{R}_{\mathbf{xy}}^{T} + \varepsilon\mathbf{I}). \end{aligned} \tag{3.5}$$

Taking the average of (3.5) with its symmetric version obtained by exchanging $\mathbf{x}$ and $\mathbf{y}$, we can obtain an alternative but equivalent expression for LDMI:

$$\begin{aligned} I_{LD}^{(\varepsilon)}(\mathbf{x};\mathbf{y}) &= \frac{1}{4}\log\det(\mathbf{R}_{\mathbf{x}} + \varepsilon\mathbf{I}) + \frac{1}{4}\log\det(\mathbf{R}_{\mathbf{y}} + \varepsilon\mathbf{I}) \\ &\quad -\frac{1}{4}\log\det(\mathbf{R}_{\mathbf{x}} - \mathbf{R}_{\mathbf{xy}}(\mathbf{R}_{\mathbf{y}} + \varepsilon\mathbf{I})^{-1}\mathbf{R}_{\mathbf{xy}}^{T} + \varepsilon\mathbf{I}) \\ &\quad -\frac{1}{4}\log\det(\mathbf{R}_{\mathbf{y}} - \mathbf{R}_{\mathbf{yx}}(\mathbf{R}_{\mathbf{x}} + \varepsilon\mathbf{I})^{-1}\mathbf{R}_{\mathbf{yx}}^{T} + \varepsilon\mathbf{I}). \end{aligned} \tag{3.6}$$

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 [Erdogan, 2022]:

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.*

Three potential advantages of using LDMI in (3.6) over SMI in (3.2) for SSL are:

i. The first apparent advantage of LDMI is that it is based solely 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 a high sample complexity. In contrast, it is more practical to obtain estimates of the autocovariance and cross-covariance matrices, as discussed in Section 4.

ii. The second advantage is related to the use of information maximization principle in connection with self-supervised training. The maximization of SMI of two vectors induces general, potentially nonlinear dependence between them, whereas the maximization of LDMI increases correlation or linear dependence between them. Therefore, LDMI-based information maximization is expected to lead to the organization of the feature space, which is more favorable for data-efficient and low-complexity linear or shallow supervised classifiers, as targeted in self-supervised learning applications.

iii. The third advantage is related to the interpretability of the resulting LDMI-based maximization problems. The corresponding objective functions involve the log-determinant of the projector-space covariance, whose maximization clearly avoids feature collapse, a significant concern in non-contrastive self-supervised methods.

3.2 CorInfoMax as a criterion based on LDMI

This section derives the optimization objective for the proposed CorInfoMax framework as a variation on the LDMI measure. The primary motivation to obtain an LDMI variant is to increase the similarity between the alternative latent representations of the same input by enforcing the linear 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 (3.6). 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 expression of LDMI in (3.6) to impose this constraint. For this purpose, we apply the first-order Taylor series approximation, $\log\det(\mathbf{C}+\mathbf{D}) \approx \log\det(\mathbf{C})+\mathrm{Tr}(\mathbf{D}^T\mathbf{C}^{-1})$,

on the third term on the right side of (3.6), which provides the following.

$$\begin{aligned}
&\log\det(\mathbf{R}_{\mathbf{z}^{(1)}} - \mathbf{R}_{\mathbf{z}^{(1)}\mathbf{z}^{(2)}}(\mathbf{R}_{\mathbf{z}^{(2)}} + \varepsilon\mathbf{I})^{-1}\mathbf{R}^{T}_{\mathbf{z}^{(1)}\mathbf{z}^{(2)}} + \varepsilon\mathbf{I} \\
&\approx \frac{1}{\varepsilon}\mathrm{Tr}(\mathbf{R}_{\mathbf{z}^{(1)}} - \mathbf{R}_{\mathbf{z}^{(1)}\mathbf{z}^{(2)}}\mathbf{R}^{-1}_{\mathbf{z}^{(2)}}\mathbf{R}^{T}_{\mathbf{z}^{(1)}\mathbf{z}^{(2)}}) + \log\det(\varepsilon\mathbf{I}) \\
&= \frac{1}{\varepsilon}\min_{\mathbf{A}_1,\mathbf{b}_1} E(\|\mathbf{z}^{(1)} - (\mathbf{A}_1\mathbf{z}^{(2)} + \mathbf{b}_1)\|_2^2) + P\log(\varepsilon).
\end{aligned} \tag{3.7}$$

Therefore, the expression in (3.7) corresponds to the mean square error of the best linear (affine) MMSE estimator of $\mathbf{z}^{(1)}$ from $\mathbf{z}^{(2)}$, multiplied by ε^{-1} [Ozsoy et al., 2022]. Similarly, if we apply the same approximation to the fourth term on the right side of (3.6), we obtain the following.

$$\begin{aligned}
&\log\det(\mathbf{R}_{\mathbf{z}^{(2)}} - \mathbf{R}^{T}_{\mathbf{z}^{(1)}\mathbf{z}^{(2)}}(\mathbf{R}_{\mathbf{z}^{(1)}} + \varepsilon\mathbf{I})^{-1}\mathbf{R}_{\mathbf{z}^{(1)}\mathbf{z}^{(2)}} + \varepsilon\mathbf{I}) \\
&\approx \frac{1}{\varepsilon}\min_{\mathbf{A}_2,\mathbf{b}_2} E(\|\mathbf{z}^{(2)} - (\mathbf{A}_2\mathbf{z}^{(1)} + \mathbf{b}_2)\|_2^2) + P\log(\varepsilon).
\end{aligned} \tag{3.8}$$

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 (3.7) and ((3.8) into $\varepsilon^{-1}E(\|\mathbf{z}^{(1)} - \mathbf{z}^{(2)}\|_2^2)$ + const. Therefore, scaling the expression in (3.6) by 4 and using this modification, we obtain the following

$$J(\mathbf{z}^{(1)}, \mathbf{z}^{(2)}) = \log\det(\mathbf{R}_{\mathbf{z}^{(1)}} + \varepsilon\mathbf{I}) + \log\det(\mathbf{R}_{\mathbf{z}^{(2)}} + \varepsilon\mathbf{I}) - 2\varepsilon^{-1}E(\|\mathbf{z}^{(1)} - \mathbf{z}^{(2)}\|_2^2). \tag{3.9}$$

where we ignored the constant terms. We refer to (3.9) as the stochastic CorInfoMax objective function. In Chapter 4, we propose a self-supervised learning method based on the optimization of this objective function.

Chapter 4

LOG-DETERMINANT MUTUAL INFORMATION BASED SELF-SUPERVISED LEARNING

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

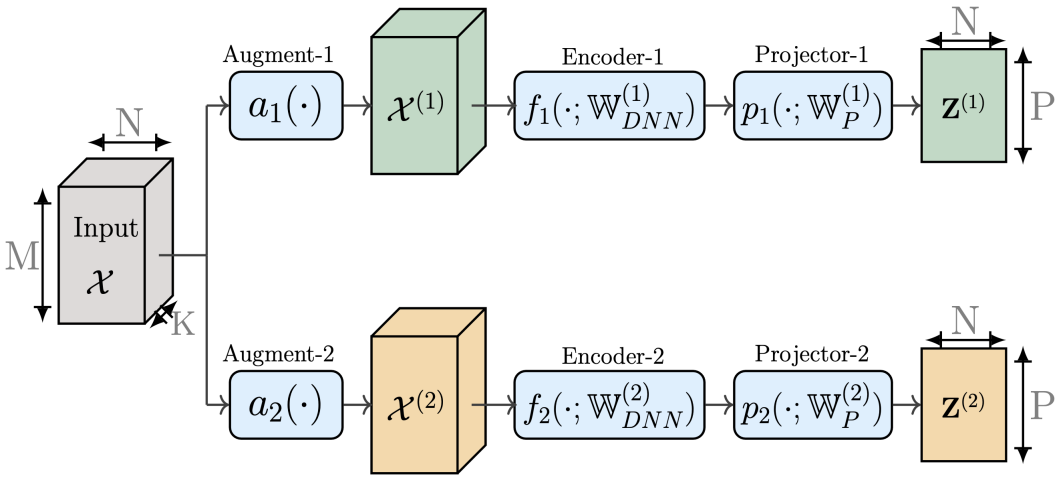


Figure 4.1: SSL setup: we consider two parallel encoder branches corresponding to two different augmentations of the same input X. Augmented views are fed into Siamese networks f followed by a projector p, basically a 3-layer MLP. N stands for batch size, $M \times K$ for image dimension, P for feature dimension of projector output.

4.1 Self-supervised learning setting

We start by describing the presumed self-supervised learning setup, illustrated in Figure 4.1:

- 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 different transformed versions of the same input. Therefore, we first apply the sequence of *augmentation functions* $a_1(\cdot)$ and $a_2(\cdot)$ to obtain transformed versions of the input sequence, i.e., $\{\mathcal{X}^{(q)}[l] = a_q(\mathcal{X}[l]) \in \mathbb{R}^{M\times K\times N}, l \in \mathbb{Z}\},\ q \in \{1,2\}$. We will illustrate particular augmentations choices in the examples provided in Chapter 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.

- In the siamese network structure, *encoder networks* are represented 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^{th} network.

- The *outputs of the encoder* for different augmented inputs are represented by $\mathbf{Y}^{(q)}[l] = f_q(\mathbb{X}^{(q)}[l]; \mathbb{W}_{DNN}^{(q)}) \in \mathbb{R}^{F\times N},\ q \in \{1,2\}, l \in \mathbb{Z}$, where F is the dimension of the output feature.

- The outputs of the enconders are fed into *projector networks* $p_q(\mathbf{Y}; \mathbb{W}_P^{(q)})$, $q = 1, 2$, where $\mathbf{Y}$ is its input, and $\mathbb{W}_P^{(q)}$ is the set of trainable parameters of the q^{th} projector.

- The *outputs of the projector network* for different augmentations are represented by $\mathbf{Z}^{(q)}[l] = p_q(\mathbf{Y}^{(q)}[l]; \mathbb{W}_P^{(q)}) \in \mathbb{R}^{P\times N},\ q \in \{1,2\}, l \in \mathbb{Z}$, where P is the projector head dimension.

For the numerical experiments in this thesis, we consider the weight-sharing setup, where both encoders and projectors use the same network weights. More specifically, the encoder networks f_1 and f_2 use the same weights, while the projector networks p_1 and p_2 use the same weights. More details are provided in the Appendix A.1.

4.2 Correlative mutual information maximization

We define the CorInfoMax approach for SSL through the optimization problem

$$\underset{\mathbb{W}_{DNN},\,\mathbb{W}_P}{\text{maximize}} \quad \hat{J}(\mathbf{z}^{(1)};\mathbf{z}^{(2)})[l], \tag{4.1a}$$

where $\hat{J}^{(\varepsilon)}$ is the sample-based estimate of (3.9) at the l^{th}-batch which can be written as

$$\begin{aligned}\hat{J}(\mathbf{z}^{(1)},\mathbf{z}^{(2)})[l])) \;&=\; \log\det(\hat{\mathbf{R}}_{\mathbf{z}^{(1)}}[l]+\varepsilon\mathbf{I})+\log\det(\hat{\mathbf{R}}_{\mathbf{z}^{(2)}}[l]+\varepsilon\mathbf{I})\\ &\quad -\frac{2}{\varepsilon N}\|\mathbf{Z}^{(1)}[l]-\mathbf{Z}^{(2)}[l]\|_F^2,\end{aligned} \tag{4.2}$$

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 (3.9), and $\hat{\mathbf{R}}_{\mathbf{z}^{(1)}}[l]$, $\hat{\mathbf{R}}_{\mathbf{z}^{(2)}}[l]$ and $\hat{\mathbf{R}}_{\mathbf{z}^{(1)}\mathbf{z}^{(2)}}[l]$ are the auto-covariance matrix and cross-covariance matrix estimates for the projector heads in the batch l^{th}. If the batch size is large enough, these estimates can be obtained from the current batch samples. However, due to hardware limitations and the fact that sufficiently small batch sizes offer better accuracy [McCandlish et al., 2018][Masters and Luschi, 2018], it may not be possible to use large batch sizes to obtain reliable covariance estimates. Therefore, we adopt recursive covariance matrix estimation across batches [Rosca et al., 2006]. The corresponding covariance update expressions take the form

$$\hat{\mathbf{R}}_{\mathbf{z}^{(q)}}[l]=\lambda\hat{\mathbf{R}}_{\mathbf{z}^{(q)}}[l-1]+(1-\lambda)\frac{1}{N}\tilde{\mathbf{Z}}^{(q)}[l]\tilde{\mathbf{Z}}^{(q)}[l]^T, \tag{4.3}$$

for $q\in\{1,2\}$ where $0\leq\lambda<1$ is the forgetting factor, and $\tilde{\mathbf{Z}}^{(q)}[l]$ represents the batch of mean-centralized projector output defined as

$$\tilde{\mathbf{Z}}^{(q)}[l]=\mathbf{Z}^{(q)}[l]-\boldsymbol{\mu}^{(q)}[l]\mathbf{1}_N^T, \tag{4.4}$$

where $\{\boldsymbol{\mu}^{(q)}[l], q \in \{1,2\}\}$ represents the mean estimates for the projector outputs, updated by

$$\boldsymbol{\mu}^{(q)}[l] = \lambda \boldsymbol{\mu}^{(q)}[l-1] + (1-\lambda)\frac{1}{N}\mathbf{Z}^{(q)}\mathbf{1}_N, \tag{4.5}$$

for $q \in \{1,2\}$. We can write the update expression for the cross-covariance matrix as follows.

$$\hat{\mathbf{R}}_{\mathbf{z}^{(1)}\mathbf{z}^{(2)}}[l] = \lambda \hat{\mathbf{R}}_{\mathbf{z}^{(1)}\mathbf{z}^{(2)}}[l-1] + (1-\lambda)\frac{1}{N}\tilde{\mathbf{Z}}^{(1)}[l]\tilde{\mathbf{Z}}^{(2)}[l]^T. \tag{4.6}$$

It is informative to inspect the terms in the objective function $\hat{J}(\mathbf{z}^{(1)}, \mathbf{z}^{(2)})$ in (4.2):

- 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 augmented versions of same input 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 the *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.

Figure 1.1.(c) illustrates the learning dynamics of the CorInfoMax approach based on (4.2) 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 Euclidian 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.

4.3 Computational complexity

In terms of complexity, the main difference between CorInfoMax and other related SSL methods is the extra log-determinant computation. The log determinant is typically computed using a decomposition method such as LU decomposition, and its gradient requires a matrix inversion. Both the determinant and the inversion have the same complexity as matrix multiplication [Bunch and Hopcroft, 1974], i.e. $O(n^{\alpha})$ with $2 < \alpha \leq 3$ to multiply two $n \times n$ matrices. Since the covariance matrices with dimension $P \times P$ are used in log-determinant computation, the complexity of loss function can be approximated to $\mathcal{O}(P^{2.xx})$.

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. Considering the projection dimensions used in the experiments, the extra cost of log-determinant and its gradients are negligible compared to the rest of the model computation, whose runtime is dominated by the encoder, as confirmed by the runtime experiments reported in Section 5.3.4.

Chapter 5

EXPERIMENTS AND RESULTS

This chapter starts with information on the used datasets. Then we share the implementation details such as the training procedure, computational resources, network architectures, and optimization procedure. In the last part, we report our experimental results with linear evaluation, semi-supervised learning, hyperparameter sensitivities, and visualizations which show effectiveness of our method, CorInfoMax.

5.1 Datasets

We perform experiments on:

- *CIFAR*-10 dataset [Krizhevsky et al., 2009] consists of 32×32 images with 10 classes. There are 5000 training images and 1000 validation images for each class.

- *CIFAR*-100 dataset [Krizhevsky et al., 2009] consists of 32×32 images with 100 classes. There are 500 training images and 100 validation images for each class.

- *Tiny ImageNet* dataset [Le and Yang, 2015] consists of 64×64 images with 200 classes. There are 500 training images and 50 validation images for each class.

- *ImageNet*-1K [Deng et al., 2009] has 1281167 different sizes of images from 1000 classes as training set. The set of 50000 validation images is treated as a test dataset for evaluation purposes.

- *ImageNet*-100 dataset [Tian et al., 2019, Kalantidis et al., 2020, Ge et al., 2021] contains 100 sub-classes, same subset as in the referenced works, of the

ImageNet dataset [Deng et al., 2009], which consists of images with variety of sizes. There are 1300 train images and 50 validation images for each class.

The images in all these datasets have three color channels.

5.2 Implementation details

5.2.1 Training procedure

The experiments consist of two consecutive stages: pretraining and linear evaluation. We first perform unsupervised pretraining of the encoder network f by applying the proposed CorInfoMax method described in Section 4.2 to the training dataset. After completion of the pre-training, we perform the linear evaluation, a standardized protocol to evaluate the quality of the learned representations [Kolesnikov et al., 2019, Chen et al., 2020a, Grill et al., 2020].

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

5.2.2 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 5.2.5.

5.2.3 Input augmentations

During *pretraining stage*, two augmented versions of each input image are generated, as shown in Figure 4.1. 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 [Grill et al., 2020] and VicReg [Bardes et al., 2021]: Each augmentation branch uses the same probability values for these randomized operations except Gaussian blurring and solarization, which use different probabilities.

During the linear evaluation training phase, a single augmentation of each input image is produced by random cropping and resizing followed by a random horizontal flip. During the linear evaluation test phase, we use resizing and center cropping augmentations, similar to [Zbontar et al., 2021],[Bardes et al., 2021]. We provide more details about the augmentations in Appendix B.1.

5.2.4 Network architecture

Encoder Network

For CIFAR datasets, we use a modified form of ResNet-18 architecture [He et al., 2016] similar to [Chen et al., 2020a, Chen and He, 2021, HaoChen et al., 2021].We use standard ResNet-50 [He et al., 2016] for Tiny ImageNet, ImageNet-100 and ImageNet-1K, and also standard ResNet-18 [He et al., 2016] 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.

Projector network

The output of the encoder network is fed into the projector network, as in Figure 4.1. 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 CIFAR-100, Tiny ImageNet and ImageNet-100, and 8192-8192-512 for ImageNet-1K. Finally, we perform the L_2-normalization on the projector output.

Linear classifier

During the linear evaluation phase, we employ a standard linear classifier whose input is the weight-frozen encoder network's output. Input dimension of linear classifier changes according to ResNet-18 or ResNet-50. The output dimension is the class number of the trained data set.

5.2.5 Optimization

For pretraining, we use 1000 epochs with a batch size of 512 for CIFAR, 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 $1e-4$. 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 (4.2) as our objective function, where we replace $\frac{2\varepsilon^{-1}}{N}$ with α, the attraction coefficient, which we consider as a separate hyperparameter. In our experiments, the diagonal perturbation is $\varepsilon = 1\text{e}{-8}$, 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 B.2.

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 $2\text{e}{-3}$ 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.

5.3 Experimental results

5.3.1 Linear evaluation

We evaluate the learned representations from the CorInfoMax pretraining by following the linear evaluation protocol explained in Sec. 5.2. As shown in Table 5.1, CorInfoMax achieves state-of-the-art performance in linear classification after pretraining. Due to the limited size of validation sets (5000-10000 samples), differences of less than 0.26 for CIFAR-10, 0.45 for CIFAR-100, 0.5 for Tiny-ImageNet, 0.56 ImageNet-100 ResNet-18, 0.54 for and ResNet-50, 0.21 for ImageNet-1K are not statistically significant for the best accuracy results in Table 5.1. The complete comparison is provided in Table B.2 in Appendix B.3. It is also interesting to observe the progress of the LDMI measure during the CorInfoMax training process, which is illustrated in Appendix 5.4.

We compare our linear evaluation results with following methods in Table 5.1; SimCLR [Chen et al., 2020a], SimSiam [Chen and He, 2021], Spectral Contrastive [HaoChen et al., 2021], BYOL [Grill et al., 2020], W-MSE 2 [Ermolov et al., 2021], MoCo-V2 [He et al., 2020], Barlow Twins [Zbontar et al., 2021] and VICReg [Bardes et al., 2021].

While competitor methods Barlow Twins [Zbontar et al., 2021] and VICReg [Bardes et al., 2021] provide results in Table 5.1 with projector output dimension 2048 for CIFAR-10 and CIFAR-100 datasets, projector output dimension 8192 for ImageNet-1K; CofInfoMax achieves its best scores with projector output dimension 64 for CIFAR-10, projector output dimension 128 for CIFAR-100, and projector output dimension 512 for ImageNet-1K.

Table 5.1: Top-1 accuracies (%) under linear evaluation on different datasets. Results are reported from [da Costa et al., 2022, Ermolov et al., 2021, HaoChen et al., 2021] for CIFAR-10 and CIFAR-100, [HaoChen et al., 2021] for Tiny ImageNet, [Ge et al., 2021, Lee et al., 2021] for ImageNet-100 (IN-100) in in ResNet-50 (Res50), [da Costa et al., 2022] for ImageNet-100 (IN-100) in ResNet-18 (Res18), [HaoChen et al., 2021, Chen and He, 2021] 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
	Res18	Res18	Res50	Res18	Res50	Res50
SimCLR	91.80	66.83	48.12	77.04	-	66.5
SimSiam	91.40	66.04	46.76	78.72	81.6	68.1
Spectral	92.07	66.18	49.86	-	-	66.97
BYOL	92.58	70.46	-	**80.32**	78.76	**69.3**
W-MSE 2	91.55	66.10	-	69.06	-	-
MoCo-V2	**92.94**	69.89	-	79.28	-	67.4
Barlow	92.10	70.90	-	**80.38**	-	68.7
VICReg	92.07	68.54	-	79.40	-	68.6
CorInfoMax	**93.18**	**71.61**	**54.86**	**80.48**	**82.64**	69.08

5.3.2 Semi-supervised learning

We compare the semi-supervised learning performance of CorInfoMax with VICReg [Bardes et al., 2021]. 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 5.2, while details are provided in Appendix B.2.5.

Table 5.2: 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 [Bardes et al., 2021].

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

Table 5.3: Number of classes, training samples, and validation samples for experimented datasets. Validation sets are used to report test accuracy.

Dataset	Class	Training	Validation
CIFAR-10	10	50000	10000
CIFAR-100	100	50000	10000
Tiny-ImageNet	200	100000	10000
ImageNet-100	100	130000	5000
ImageNet-1K	1000	1281167	50000

5.3.3 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: attraction coefficient in Table 5.4, batch size in Table 5.5, learning rate in Table 5.6, projector output dimension in Table 5.7. The results show that our approach is fairly robust to small changes in the hyperparameter.

Table 5.4: Top-1 accuracies (%) under linear evaluation for the different attraction coefficients (α) with the setup which provides best result for CIFAR-100 dataset.

Attraction coefficient (α)	125	250	500	1000	2000
Top-1 accuracy (%)	69.04	71.19	71.34	71.61	69.96

Table 5.5: Top-1 accuracies (%) under linear evaluation 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

Table 5.6: Top-1 accuracies (%) under linear evaluation 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

Table 5.7: Top-1 accuracies (%) under linear evaluation 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

5.3.4 Computational complexity

We selected VICReg [Bardes et al., 2021] for comparison and then integrated their loss function into our code to eliminate differences in the implementation of the methods. We ran 10 epochs with both loss functions for different projector output dimensions that changed between 64 and 1024, and the results are reported in Table 5.8.

Table 5.8: 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 [Bardes et al., 2021] 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

5.4 Visualization of LDMI 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 (4.3) during training) and the test accuracy values. Figure 5.1 shows the progress of the Training-LDMI and the test accuracy curves on the same graph for the CIFAR-10 dataset.

Similarly, Figure 5.2 shows the progress of the Training-LDMI and the test accuracy curves on the same graph for the CIFAR-100 dataset.

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

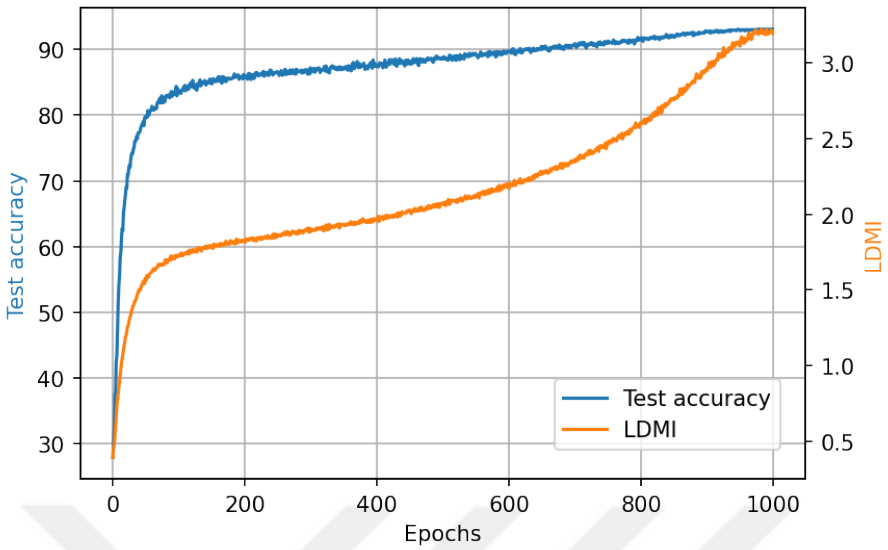


Figure 5.1: Evolution of the LDMI measure and the test accuracy for the CIFAR-10 dataset as a function of the CorInfoMax algorithm epochs.

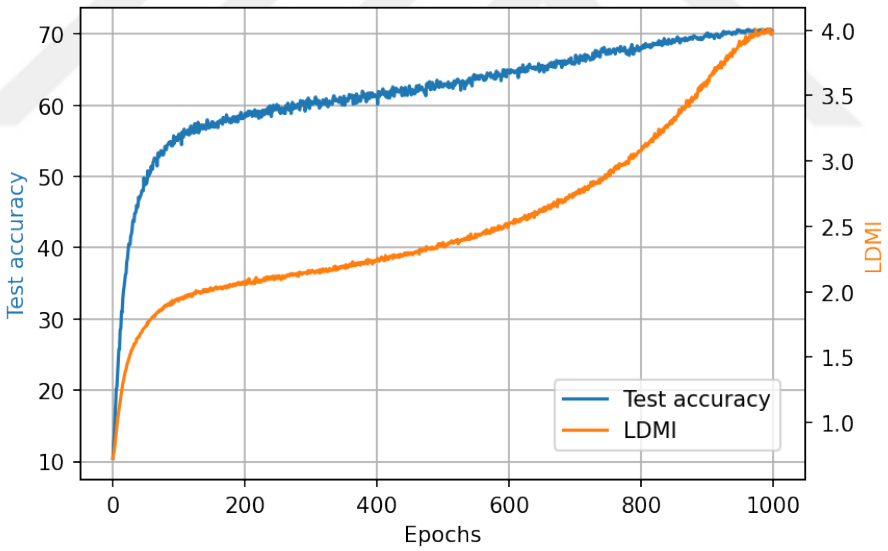


Figure 5.2: Evolution of the LDMI measure and the test accuracy for the CIFAR-100 dataset as a function of the CorInfoMax algorithm epochs.

5.5 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 (4.2), we expect that no dimensional collapse occurs (with very small ε), 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 5.3 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 the first 50 eigenvalues. Furthermore, the smallest 8-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.

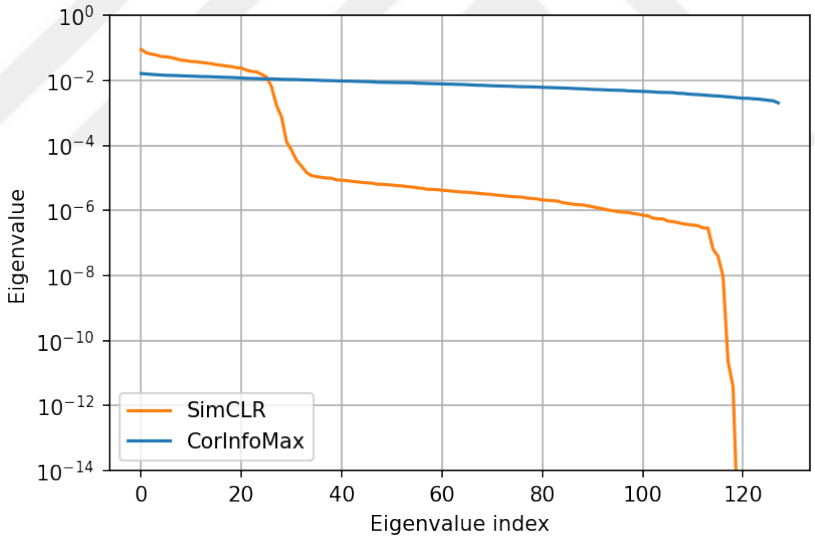


Figure 5.3: Comparison of the sorted eigenvalues of the projector vector covariance matrix $\hat{\mathbf{R}}_{\mathbf{z}^{(1)}}$ for CorInfoMax and SimCLR algorithms, for CIFAR-10 dataset and projector dimension of 128.

5.6 Embedding visualization after pretraining

In Figure 5.4, we use t-Distributed Stochastic Neighbor Embedding (t-SNE) to visualize where the test embeddings are located after pretraining. We get embeddings

from the output of the encoder network using our pretrained model, and 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.

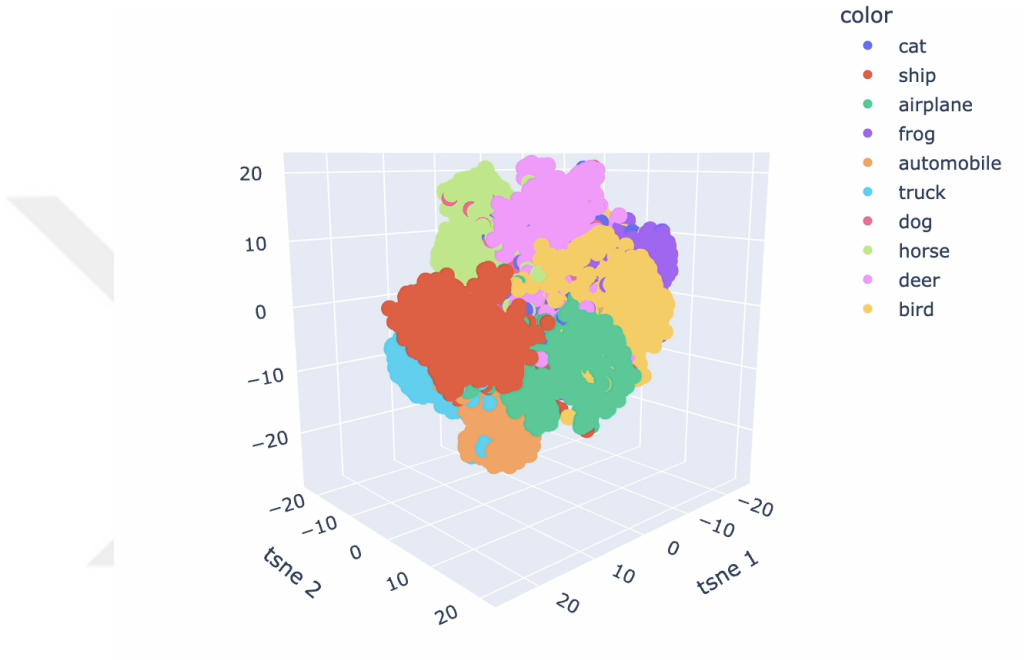


Figure 5.4: t-SNE visualization of the embeddings obtained from the CIFAR-10 test dataset from the output of the encoder network after 1000-epoch pretraining. Each color represents one class of CIFAR-10.

Chapter 6

CONCLUSION

We introduced CorInfoMax, a novel SSL framework based on the maximization of correlative information. CorInfoMax avoids both collapse and dimensional collapse to enforce latent representations scattering in the feature space. Our experiments demonstrate the state-of-the-art performance of CorInfoMax in numerous downstream tasks by using smaller projection output dimensions. We also performed run-time experiments to address potential concerns about computation complexity. According to the result, the CorInfoMax loss function does not have a significant impact on training time.

Although CorInfoMax is limitedly affected by hyperparameter changes, establishing a systematic hyperparameter selection method to achieve the best possible accuracy on a given task and dataset could be an important future work. In addition, the application range of CorInfoMax can be expanded by using different encoder types, using more than two input images, and using multimodal inputs such as audio-text, video-audio, video-subtitle, and image-title.

BIBLIOGRAPHY

[Agrawal et al., 2015] Agrawal, P., Carreira, J., and Malik, J. (2015). Learning to see by moving. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 37–45.

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

[Bachman et al., 2019] Bachman, P., Hjelm, R. D., and Buchwalter, W. (2019). Learning representations by maximizing mutual information across views. *Advances in neural information processing systems*, 32.

[Baevski et al., 2020] Baevski, A., Zhou, Y., Mohamed, A., and Auli, M. (2020). wav2vec 2.0: A framework for self-supervised learning of speech representations. *Advances in Neural Information Processing Systems*, 33:12449–12460.

[Bardes et al., 2021] Bardes, A., Ponce, J., and LeCun, Y. (2021). Vicreg: Variance-invariance-covariance regularization for self-supervised learning. *arXiv preprint arXiv:2105.04906*.

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

[Bell and Sejnowski, 1997] Bell, A. J. and Sejnowski, T. J. (1997). The "independent components" of natural scenes are edge filters. *Vision research*, 37(23):3327–3338.

[Bromley et al., 1993] Bromley, J., Bentz, J. W., Bottou, L., Guyon, I., LeCun, Y.,

Moore, C., Säckinger, E., and Shah, R. (1993). Signature verification using a "siamese" time delay neural network. In *Int. J. Pattern Recognit. Artif. Intell.*

[Brown et al., 2020] Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. (2020). Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

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

[Caron et al., 2018] Caron, M., Bojanowski, P., Joulin, A., and Douze, M. (2018). Deep clustering for unsupervised learning of visual features. In *Proceedings of the European conference on computer vision (ECCV)*, pages 132–149.

[Caron et al., 2020] Caron, M., Misra, I., Mairal, J., Goyal, P., Bojanowski, P., and Joulin, A. (2020). Unsupervised learning of visual features by contrasting cluster assignments. *Advances in Neural Information Processing Systems*, 33:9912–9924.

[Chan et al., 2011] Chan, T.-H., Ma, W.-K., Ambikapathi, A., and Chi, C.-Y. (2011). A simplex volume maximization framework for hyperspectral endmember extraction. *IEEE Transactions on Geoscience and Remote Sensing*, 49(11):4177–4193.

[Chen et al., 2020a] Chen, T., Kornblith, S., Norouzi, M., and Hinton, G. (2020a). A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR.

[Chen et al., 2020b] Chen, X., Fan, H., Girshick, R., and He, K. (2020b). Improved baselines with momentum contrastive learning. *arXiv preprint arXiv:2003.04297*.

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

[Chopra et al., 2005] Chopra, S., Hadsell, R., and LeCun, Y. (2005). Learning a similarity metric discriminatively, with application to face verification. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 1, pages 539–546 vol. 1.

[Cover and Thomas, 2006] Cover, T. M. and Thomas, J. A. (2006). *Elements of Information Theory (Wiley Series in Telecommunications and Signal Processing)*. Wiley-Interscience, USA.

[Cuturi, 2013] Cuturi, M. (2013). Sinkhorn distances: Lightspeed computation of optimal transport. *Advances in neural information processing systems*, 26.

[da Costa et al., 2022] da Costa, V. G. T., Fini, E., Nabi, M., Sebe, N., and Ricci, E. (2022). Solo-learn: A library of self-supervised methods for visual representation learning. *Journal of Machine Learning Research*, 23(56):1–6.

[Deng et al., 2009] Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255.

[Devlin et al., 2018] Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

[Doersch et al., 2015] Doersch, C., Gupta, A., and Efros, A. A. (2015). Unsupervised visual representation learning by context prediction. In *Proceedings of the IEEE international conference on computer vision*, pages 1422–1430.

[Dosovitskiy et al., 2014] Dosovitskiy, A., Springenberg, J. T., Riedmiller, M., and Brox, T. (2014). Discriminative unsupervised feature learning with convolutional neural networks. *Advances in neural information processing systems*, 27.

[Dwibedi et al., 2021] Dwibedi, D., Aytar, Y., Tompson, J., Sermanet, P., and Zisserman, A. (2021). With a little help from my friends: Nearest-neighbor

contrastive learning of visual representations. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9588–9597.

[Erdogan, 2022] Erdogan, A. T. (2022). 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.

[Ermolov et al., 2021] Ermolov, A., Siarohin, A., Sangineto, E., and Sebe, N. (2021). Whitening for self-supervised representation learning. In *International Conference on Machine Learning*, pages 3015–3024. PMLR.

[Fu et al., 2019] Fu, X., Huang, K., Sidiropoulos, N. D., and Ma, W.-K. (2019). Nonnegative matrix factorization for signal and data analytics: Identifiability, algorithms, and applications. *IEEE Signal Process. Mag.*, 36(2):59–80.

[Fu et al., 2016] Fu, X., Huang, K., Yang, B., Ma, W.-K., and Sidiropoulos, N. D. (2016). Robust volume minimization-based matrix factorization for remote sensing and document clustering. *IEEE Transactions on Signal Processing*, 64(23):6254–6268.

[Ge et al., 2021] Ge, S., Mishra, S., Li, C.-L., Wang, H., and Jacobs, D. (2021). Robust contrastive learning using negative samples with diminished semantics. *Advances in Neural Information Processing Systems*, 34.

[Gidaris et al., 2018] Gidaris, S., Singh, P., and Komodakis, N. (2018). Unsupervised representation learning by predicting image rotations. *arXiv preprint arXiv:1803.07728*.

[Gretton et al., 2005] Gretton, A., Bousquet, O., Smola, A., and Schölkopf, B. (2005). Measuring statistical dependence with hilbert-schmidt norms. In *International conference on algorithmic learning theory*, pages 63–77. Springer.

[Grill et al., 2020] Grill, J.-B., Strub, F., Altché, F., Tallec, C., Richemond, P., Buchatskaya, E., Doersch, C., Avila Pires, B., Guo, Z., Gheshlaghi Azar, M., et al. (2020). Bootstrap your own latent-a new approach to self-supervised learning. *Advances in Neural Information Processing Systems*, 33:21271–21284.

[Gutmann and Hyvärinen, 2010] Gutmann, M. and Hyvärinen, A. (2010). Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In Teh, Y. W. and Titterington, M., editors, *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, volume 9 of *Proceedings of Machine Learning Research*, pages 297–304, Chia Laguna Resort, Sardinia, Italy. PMLR.

[Gutmann and Hyvärinen, 2012] Gutmann, M. U. and Hyvärinen, A. (2012). Noise-contrastive estimation of unnormalized statistical models, with applications to natural image statistics. *Journal of machine learning research*, 13(2).

[Hadsell et al., 2006] Hadsell, R., Chopra, S., and LeCun, Y. (2006). Dimensionality reduction by learning an invariant mapping. *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, 2:1735–1742.

[HaoChen et al., 2021] HaoChen, J. Z., Wei, C., Gaidon, A., and Ma, T. (2021). Provable guarantees for self-supervised deep learning with spectral contrastive loss. *Advances in Neural Information Processing Systems*, 34.

[He et al., 2020] He, K., Fan, H., Wu, Y., Xie, S., and Girshick, R. (2020). Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9729–9738.

[He et al., 2016] He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

[Hjelm et al., 2018] Hjelm, R. D., Fedorov, A., Lavoie-Marchildon, S., Grewal, K., Bachman, P., Trischler, A., and Bengio, Y. (2018). Learning deep representations by mutual information estimation and maximization. *arXiv preprint arXiv:1808.06670*.

[Hua et al., 2021] Hua, T., Wang, W., Xue, Z., Ren, S., Wang, Y., and Zhao, H. (2021). On feature decorrelation in self-supervised learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9598–9608.

[Huang et al., 2018] Huang, L., Yang, D., Lang, B., and Deng, J. (2018). Decorrelated batch normalization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 791–800.

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

[Jing et al., 2021] Jing, L., Vincent, P., LeCun, Y., and Tian, Y. (2021). Understanding dimensional collapse in contrastive self-supervised learning. *arXiv preprint arXiv:2110.09348*.

[Kailath et al., 2000] Kailath, T., Sayed, A. H., and Hassibi, B. (2000). *Linear estimation*. Number BOOK. Prentice Hall.

[Kalantidis et al., 2020] Kalantidis, Y., Sariyildiz, M. B., Pion, N., Weinzaepfel, P., and Larlus, D. (2020). Hard negative mixing for contrastive learning. *Advances in Neural Information Processing Systems*, 33:21798–21809.

[Koch et al., 2015] Koch, G., Zemel, R., Salakhutdinov, R., et al. (2015). Siamese neural networks for one-shot image recognition. In *ICML deep learning workshop*, volume 2, page 0. Lille.

[Kolesnikov et al., 2019] Kolesnikov, A., Zhai, X., and Beyer, L. (2019). Revisiting

self-supervised visual representation learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 1920–1929.

[Krizhevsky et al., 2009] Krizhevsky, A., Hinton, G., et al. (2009). Learning multiple layers of features from tiny images.

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

[LeCun, 2022] LeCun, Y. (2022). A path towards autonomous machine intelligence version 0.9.2, 2022-06-27,. `https://openreview.net/pdf?id=BZ5a1r-kVsf`. Accessed: 2022-07-17.

[Lee et al., 2021] Lee, H., Lee, K., Lee, K., Lee, H., and Shin, J. (2021). Improving transferability of representations via augmentation-aware self-supervision. *Advances in Neural Information Processing Systems*, 34.

[Li et al., 2021] Li, Y., Pogodin, R., Sutherland, D. J., and Gretton, A. (2021). Self-supervised learning with kernel dependence maximization. *Advances in Neural Information Processing Systems*, 34:15543–15556.

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

[Masters and Luschi, 2018] Masters, D. and Luschi, C. (2018). Revisiting small batch training for deep neural networks. *arXiv preprint arXiv:1804.07612*.

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

[McCandlish et al., 2018] McCandlish, S., Kaplan, J., Amodei, D., and Team, O. D. (2018). An empirical model of large-batch training. *arXiv preprint arXiv:1812.06162*.

[Mikolov et al., 2013] Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013). Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

[Mnih and Kavukcuoglu, 2013] Mnih, A. and Kavukcuoglu, K. (2013). Learning word embeddings efficiently with noise-contrastive estimation. *Advances in neural information processing systems*, 26.

[Noroozi and Favaro, 2016] Noroozi, M. and Favaro, P. (2016). Unsupervised learning of visual representations by solving jigsaw puzzles. In *European conference on computer vision*, pages 69–84. Springer.

[Oord et al., 2018] Oord, A. v. d., Li, Y., and Vinyals, O. (2018). Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*.

[Ozsoy et al., 2022] Ozsoy, S., Hamdan, S., Arik, S. O., Yuret, D., and Erdogan, A. T. (2022). Self-supervised learning with an information maximization criterion. Manuscript submitted for Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022.

[Pathak et al., 2016] Pathak, D., Krahenbuhl, P., Donahue, J., Darrell, T., and Efros, A. A. (2016). Context encoders: Feature learning by inpainting. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2536–2544.

[Poole et al., 2019] Poole, B., Ozair, S., Van Den Oord, A., Alemi, A., and Tucker, G. (2019). On variational bounds of mutual information. In *International Conference on Machine Learning*, pages 5171–5180. PMLR.

[Radford and Narasimhan, 2018] Radford, A. and Narasimhan, K. (2018). Improving language understanding by generative pre-training.

[Radford et al., 2019] Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., and Sutskever, I. (2019). Language models are unsupervised multitask learners.

[Raffel et al., 2020] Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., Liu, P. J., et al. (2020). Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21(140):1–67.

[Rosca et al., 2006] Rosca, J., Erdogmus, D., Príncipe, J. C., and Haykin, S. (2006). *Independent Component Analysis and Blind Signal Separation: 6th International Conference, ICA 2006, Charleston, SC, USA, March 5-8, 2006, Proceedings*, volume 3889. Springer.

[Schneider et al., 2019] Schneider, S., Baevski, A., Collobert, R., and Auli, M. (2019). wav2vec: Unsupervised pre-training for speech recognition. *arXiv preprint arXiv:1904.05862*.

[Schroff et al., 2015] Schroff, F., Kalenichenko, D., and Philbin, J. (2015). Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 815–823.

[Sohn, 2016] Sohn, K. (2016). Improved deep metric learning with multi-class n-pair loss objective. In *NIPS*.

[Tatli and Erdogan, 2021] Tatli, G. and Erdogan, A. T. (2021). Polytopic matrix factorization: Determinant maximization based criterion and identifiability. *IEEE Transactions on Signal Processing*, 69:5431–5447.

[Tian et al., 2019] Tian, Y., Krishnan, D., and Isola, P. (2019). Contrastive multiview coding. *arXiv preprint arXiv:1906.05849*.

[Tishby et al., 2000] Tishby, N., Pereira, F. C., and Bialek, W. (2000). The information bottleneck method. *arXiv preprint physics/0004057*.

[Tschannen et al., 2019] Tschannen, M., Djolonga, J., Rubenstein, P. K., Gelly, S., and Lucic, M. (2019). On mutual information maximization for representation learning. In *International Conference on Learning Representations*.

[Wang and Liu, 2021] Wang, F. and Liu, H. (2021). Understanding the behaviour of contrastive loss. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2495–2504.

[Wu et al., 2018] Wu, Z., Xiong, Y., Yu, S. X., and Lin, D. (2018). Unsupervised feature learning via non-parametric instance discrimination. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3733–3742.

[Zbontar et al., 2021] Zbontar, J., Jing, L., Misra, I., LeCun, Y., and Deny, S. (2021). Barlow twins: Self-supervised learning via redundancy reduction. In *International Conference on Machine Learning*, pages 12310–12320. PMLR.

[Zhang et al., 2016] Zhang, R., Isola, P., and Efros, A. A. (2016). Colorful image colorization. In *European conference on computer vision*, pages 649–666. Springer.

[Zhang et al., 2022] Zhang, S., Qiu, L., Zhu, F., Yan, J., Zhang, H., Zhao, R., Li, H., and Yang, X. (2022). Align representations with base: A new approach to self-supervised learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16600–16609.

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

[Zhouyin and Liu, 2021] Zhouyin, Z. and Liu, D. (2021). Understanding neural networks with logarithm determinant entropy estimator. *arXiv preprint arXiv:2105.03705*.

Appendix A

SSL SETUP DETAILS

A.1 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 4.1 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 encoder, i.e., $f(\cdot;\mathbb{W}_{DNN})$ whose output is input to the projection network $p(\cdot;\mathbb{W}_P)$.

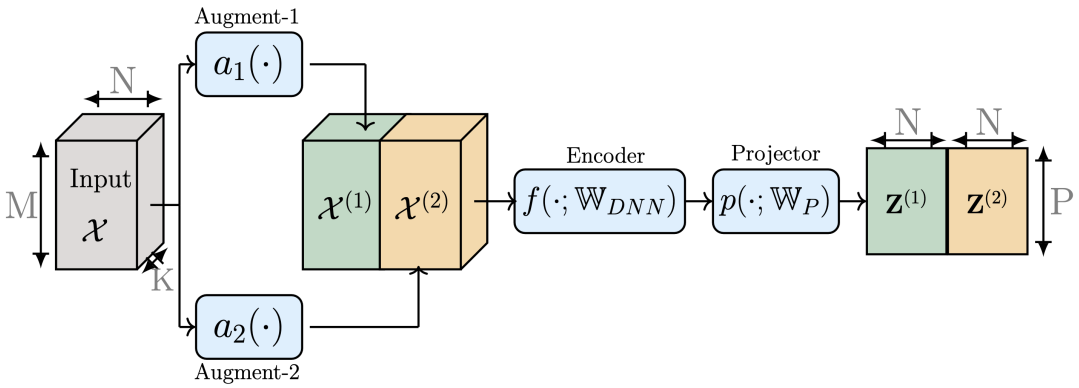


Figure A.1: Self-supervised learning set-up with weight sharing.

A.2 Pseudocode of CorInfoMax

Algorithm 1 describes the main steps of the CorInfoMax approach in a PyTorch-style pseudocode format on the next page.

Algorithm 1: PyTorch-style pseudocode for CorInfoMax

```
# f:  encoder with projector network
# N: batch size, D: projector output dimension
# R1, R2:  covariance matrices - initialized as identity (DXD)
# mu1, mu2:  means vectors - 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
   x1, x2 = augmentation(x) # random augmentations
   z1 = f(x1) # projector output-1
   z2 = f(x2) # projector output-2
   # 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
   sim_loss = mse_loss(z1, z2) # attraction
   loss = cor_loss + alpha * sim_loss
   # optimization
   loss.backward()
   optimizer.step()
```

Appendix B

EXPERIMENT DETAILS

B.1 Image augmentations

B.1.1 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 the cropped area to 32×32 for CIFAR datasets, 64×64 for Tiny ImageNet, and 224×224 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 randomly selected (uniform) 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 B.1.

- 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 randomly selected (uniform) between 0.1 and 2.0 pixels.

- Solarization: inverting the values of the image pixels by subtracting the maximum value. We keep the result if it is above the threshold value; otherwise, we replace it with the original value. The default threshold value is 128.

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

Table B.1: 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

B.1.2 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 [Grill et al., 2020], [Zbontar et al., 2021], [Bardes et al., 2021], [Chen and He, 2021]. For the random-resized-crop operation, the target sizes are 32×32 for CIFAR datasets, 64×64 for Tiny ImageNet, and 224×224 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 [Grill et al., 2020]: input images are resized to 256×256 then center cropped to 224×224. For the other datasets, we apply a similar preprocessing by preserving the resize-crop ratio, as in [HaoChen et al., 2021].

B.1.3 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.

B.2 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 obtain our reported result with $8192 - 8192 - 512$ with a batch size of 1536.

For the covariance update expression in (4.3), we use the initialization $\hat{\mathbf{R}}_{\mathbf{z}^{(q)}}[0] = \mathbf{I}$ for $q = 1, 2$. We initialize the cross-covariance matrix with $\hat{\mathbf{R}}_{\mathbf{z}^{(1)}\mathbf{z}^{(2)}}[0] = \mathbf{0}$. 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 $\varepsilon = 1\text{e}{-8}$.

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.

In the following, we summarize the experimental details for each dataset.

B.2.1 CIFAR-10 experiment

For CIFAR-10, the encoder network is ResNet-18 modified for CIFAR based on the small (32×32) input image size. The modifications are: using a smaller kernel size, 3×3 instead of 7×7, 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 $1\text{e}{-6}$. The weight decay parameter for the optimizer is 0.0001.

B.2.2 CIFAR-100 experiment

For CIFAR-100, the encoder network is modified ResNet-18, as explained in the previous 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 1e−6 until the end of the training. The weight decay parameter for the optimizer is 0.0001.

B.2.3 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 5.1, 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 1e−6 until end of the training. The weight decay parameter for the optimizer is 0.0001.

B.2.4 ImageNet-100 experiments

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 our model for 400 epochs to make it more comparable with other methods in Table 5.1, 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 decays to 0.005 until the end of the training. The weight decay parameter for the optimizer is 0.0001.

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 our model for 200 epochs to make CorInfoMax more comparable to other methods in Table 5.1, 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 start value of the learning rate 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.

B.2.5 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 start value of the learning rate is 0.003 for the warm-up period. After reaching 0.2, the learning rate is decayed to $1e{-}6$ until end of training. The weight decay parameter for the optimizer is 0.0001.

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 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 [Bardes et al., 2021], then used published parameters and VICReg codes[Bardes et al., 2021] to evaluate the fine-tuning results.

B.3 Full comparison with solo-learn

Some of the results in Table 5.1 are from the solo-learn [da Costa et al., 2022]. Table B.2 shows the full results of [da Costa et al., 2022] compared to CorInfoMax.

Table B.2: 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**

Appendix C

SUPPLEMENTARY NOTES ON CORINFOMAX CRITERION

C.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 (4.2) with respect to the projector outputs, which are backpropagated to the train the encoder networks. [Ozsoy et al., 2022]

For the first term on the right side of (4.2), we can write

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

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 (4.2), 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 (4.2), which is the Euclidian distance based loss, we can write

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

and

$$\nabla_{\mathbf{z}_n^{(2)}} \left(-\frac{2}{\varepsilon N} \|\mathbf{Z}^{(1)}[l] - \mathbf{Z}^{(2)}[l]\|_F^2 \right) = \frac{4}{\varepsilon N}(\mathbf{z}_n^{(1)} - \mathbf{z}_n^{(2)}). \quad \text{(A.3)}$$

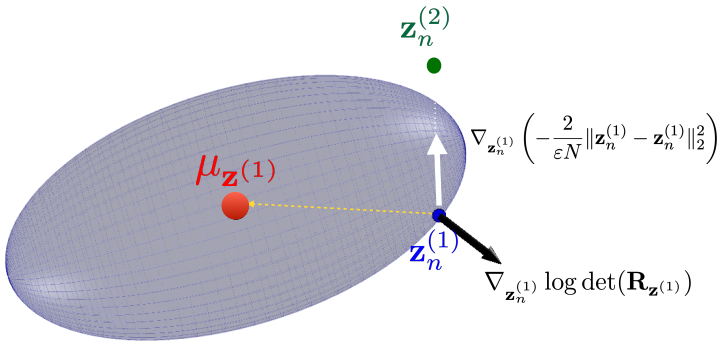


Figure C.1: Gradients of the objective function for CorInfoMax: The ellipsoid is the level surface of the quadratic function $q(\mathbf{z})$ in (A.4) containing $\mathbf{z}_n^{(1)}$, one of the projector-1 output samples, the black arrow represents the gradient of $\log\det(\mathbf{R}_{\mathbf{z}^{(1)}})$ 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 $\mathbf{z}_n^{(1)}$.

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

$$q(\mathbf{z})=\frac{(1-\lambda)}{N}(\mathbf{z}-\boldsymbol{\mu}_{\mathbf{z}^{(1)}}[l])^T(\hat{\mathbf{R}}_{\mathbf{z}^{(1)}}[l]+\varepsilon\mathbf{I})^{-1}(\mathbf{z}-\boldsymbol{\mu}_{\mathbf{z}^{(1)}}[l]), \tag{A.4}$$

at $\mathbf{z}=\mathbf{z}_n^{(1)}$, which is illustrated by the black vector in Figure C.1. 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 C.1, 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.1) is pointing away from the center of the ellipsoid, encouraging the expansion.

On the other hand, the gradient expressions in (A.2) and (A.3) 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.