



A survey on self-supervised methods for visual representation learning

Tobias Uelwer¹ · Jan Robine¹ · Stefan Sylvius Wagner² · Marc Höftmann¹ · Eric Upschulte^{3,4} · Sebastian Konietzny¹ · Maike Behrendt² · Stefan Harmeling¹

Received: 9 January 2024 / Revised: 9 August 2024 / Accepted: 7 November 2024 /
Published online: 4 March 2025
© The Author(s) 2025

Abstract

Learning meaningful representations is at the heart of many tasks in the field of modern machine learning. Recently, a lot of methods were introduced that allow learning of image representations without supervision. These representations can then be used in downstream tasks like classification or object detection. The quality of these representations is close to supervised learning, while no labeled images are needed. This survey paper provides a comprehensive review of these methods in a unified notation, points out similarities and differences of these methods, and proposes a taxonomy which sets these methods in relation to each other. Furthermore, our survey summarizes the most recent experimental results reported in the literature in form of a meta-study. Our survey is intended as a starting point for researchers and practitioners who want to dive into the field of representation learning.

Keywords Representation learning · Self-supervised learning · Meta-study · Survey

1 Introduction

Images often contain a lot of information which is irrelevant for further downstream tasks. A *representation* of an image ideally extracts the relevant parts of it. The goal of *representation learning* is to learn an *encoder network* f_θ with learnable parameters θ that maps an input image x to a lower-dimensional representation (embedding) $y = f_\theta(x)$. The main purpose of this paper is to present and discuss the different approaches and ideas for finding useful encoders. Note that in the machine learning literature this setup is also called feature learning or feature extraction.

Editor: Enver Sangineto.

Extended author information available on the last page of the article

1.1 Supervised and unsupervised learning

Before categorizing the various methods of representation learning, we begin by distinguishing the two basic settings of machine learning: (1) *supervised learning* learns a function given labeled data points, interpreted as input and output examples, (2) whereas in *unsupervised learning* we learn something given only data points (unlabeled). It is possible to take the view that also for unsupervised learning the goal is to learn a function, e.g., in clustering we learn a classification function and invent labels at the same time, in dimensionality reduction (such as PCA) we learn a regression function and invent lower-dimensional embeddings simultaneously. In light of this, representation learning is an instance of unsupervised learning, since we are only given a set of unlabeled data points and our goal is to learn an encoder, that maps the data onto some other representation that has favorable properties.

1.2 Self-supervised learning

Recently, new machine learning methods emerged, that have been labeled *self-supervised learning*. In a nutshell, such methods create an encoder by performing the following two separate steps:

- (i) Invent a supervised learning task by creating a output t for each given image x .
- (ii) Apply supervised learning to learn a function from inputs x to targets t .

The mechanism that generates the targets can be manually designed or can include learned neural networks. Note that the targets are not necessary stationary, but could change during training. Even though self-supervised learning applies classical supervised learning as its second step, it is overall best viewed as an unsupervised method, since it only takes unlabeled images as its starting point.

1.3 Outline

This paper strives to give an overview over recent advances in representation learning. Starting from the autoencoder, we will discuss different methods which we group in the following way:

- *Pretext task methods* solve an auxiliary task, e.g., predicting the angle by which an input image was rotated. The idea is that the representations learned along the way are also useful for solving other tasks. To create a pretext task, one usually needs to select image transformations and define a new loss function. The encoder will only extract information that is relevant for solving this task, so extra care must be taken. We will further discuss these methods in Sect. 2.
- *Information maximization methods* allow networks to learn representations that are invariant to various image transformations and at the same time avoid trivial solutions by maximizing information content. These representations prove useful for other downstream tasks, such as classification. Much like with pretext task methods, the transformations need to be chosen with care, as they influence which information

is targeted to be preserved or suppressed. Section 3 presents some of these methods in detail.

- *Teacher-student methods* consist of two networks where one extracts knowledge from the other. Trivial solutions are avoided by using asymmetric architectures where the teacher provides stable representations for the student to learn, while invariances to various image transformations are usually distilled onto the teacher. We will take a closer look at these methods in Sect. 4.1.1.
- *Contrastive methods* discriminate between positive and negative examples that are defined by the method on-the-fly. Maximizing the similarity between positive examples while repelling negatives offers a powerful way to extract rich and context-specific features. However, selecting large numbers of appropriate negative examples is a challenging task. In Sect. 5 we review these contrastive methods in detail.
- *Clustering-based methods* invent multiple class labels by clustering the representations and then train a classifier on those labels. Intuitively, semantically similar images should lead to similar representations that should also be assigned to the same clusters by some clustering approach. Cluster assignments can replace memory inefficient feature comparisons of negative samples from memory banks and also provide some interpretability of the image representations. However, the lack of labels in the unsupervised environment require more sophisticated training in this setup. Section 6 summarizes the most relevant representation learning methods that use clustering.

Note that the order in which we present the methods in this paper is not the same as the order in which they have been introduced in the literature. There also exist approaches which are specialized for graphs (Grover & Leskovec, 2016; Perozzi et al., 2014; Kipf & Welling, 2016), time-series (Eldele et al., 2021) or text (Mikolov et al., 2013b, a; Kenton & Toutanova, 2019). However, we omit those in this work and focus on approaches for visual representation learning.

In Sect. 7 we further put the discussed methods in relation to each other and in Sect. 9 we summarize experimental results that were reported in the literature in a meta study.

1.4 Related works

There exist also other works which are related to our survey. Ozbulak et al. (2023) give an overview of existing self-supervised learning methods and explain important concepts and loss functions. Geiping et al. (2023) provide an overview of foundational techniques and practical tips for successfully training self-supervised methods. Shwartz-Ziv and LeCun (2024) review self-supervised learning methods from an information theory perspective. Representation learning techniques for videos are discussed by Ravanbakhsh et al. (2023). Nozawa and Sato (2022) discuss evaluation procedures for self-supervised learning methods. Representation learning for reinforcement learning methods are reviewed by Botteghi et al. (2022).

1.5 Notation

Before describing the specifics of the different representation learning methods, we start by defining the notation used in this paper. Given a dataset of images, we write

$$X = [x_1, \dots, x_n] \quad (1)$$

for a randomly sampled batch of images. Every representation learning method trains an encoder network f_θ , where θ are the learnable parameters. This encoder network computes a representation

$$Y = [y_1, \dots, y_n] = [f_\theta(x_1), \dots, f_\theta(x_n)] = f_\theta(X) \quad (2)$$

of the images in X . Some methods additionally train a projection network g_ϕ , with parameters ϕ , that computes projections

$$Z = [z_1, \dots, z_n] = [g_\phi(y_1), \dots, g_\phi(y_n)] = g_\phi(Y) \quad (3)$$

of the representations in Y . There are methods that also train a prediction network q_ψ , with parameters ψ , that computes a prediction based on z or y . Both, projections and predictions are only used to train the network and after training the projection and prediction networks are discarded, and only the encoder f_θ is used for downstream tasks.

Most methods apply a transformation to the input image to obtain *views* of the original image. We write $x_i^{(j)} = t(x_i)$ for the j -th view of the original image x_i , which was obtained by applying the transformation t . Usually, these methods randomly sample transformations from a given set of transformations $\mathcal{T}$ and can differ for each image in the batch. This is why we treat t as a random variable that is sampled for each image of the batch. In contrast to that, other methods use predefined transformations $t^{(1)}, \dots, t^{(m)}$ that are fixed and do not change. In Sect. 3 we give more details and some examples of these transformations. We write

$$X^{(j)} = [x_1^{(j)}, \dots, x_n^{(j)}] = t([x_1, \dots, x_n]) = t(X) \quad (4)$$

for the batch of the j -th view. We use a similar notation for the resulting representations $Y^{(j)} = [y_1^{(j)}, \dots, y_n^{(j)}]$ and projections $Z^{(j)} = [z_1^{(j)}, \dots, z_n^{(j)}]$. Some methods split the input image into patches, which we also consider as a special case of transformation, where each patch is a view of the image. In that case we write $X_i = [x_i^{(1)}, \dots, x_i^{(m)}]$, which contains all m views of the image x_i and denote the corresponding representations and projections as $Y_i = [y_i^{(1)}, \dots, y_i^{(m)}]$ and $Z_i = [z_i^{(1)}, \dots, z_i^{(m)}]$.

Several approaches rely on a Siamese network architecture (Bromley et al., 1993), where the networks share the same weights across all views. In this case, the views are processed independently, i.e., $Y_i = [f_\theta(x_i^{(1)}), \dots, f_\theta(x_i^{(m)})] = f_\theta(X_i)$, instead of processing them simultaneously. To highlight this special case, we call the network f_θ a *Siamese encoder*. The same distinction can be made for the other networks. A *Siamese projector* computes projections $Z_i = [g_\phi(y_i^{(1)}), \dots, g_\phi(y_i^{(m)})] = g_\phi(Y_i)$ individually.

We use $\mathcal{L}$ to refer to the loss function that is used to train the parameters using stochastic gradient descent (SGD). Sometimes the total loss consists of multiple components which we denote using the letter ℓ .

We use square brackets to access elements of vectors and matrices, e.g., the j -th element of vector v is written as $v[j]$ and the entry at column j and row k of matrix C is written as $C[j, k]$. Furthermore, we define the softmax function that normalizes a vector $v \in \mathbb{R}^d$ to a probability distribution as

$$(\text{softmax}_\tau(v))[j] = \frac{\exp(v[j]/\tau)}{\sum_{k=1}^d \exp(v[k]/\tau)} \text{ for } j = 1, \dots, d, \quad (5)$$

where $\tau > 0$ is a temperature parameter which controls the entropy of that distribution (Wu et al., 2018). The higher the value of τ the closer the normalized distribution is to a uniform distribution. We write $\text{softmax}(v)$ when no temperature is used, i.e., when $\tau = 1$.

1.5.1 Distance metrics and similarity measures

Throughout the paper, we use various concepts to compare two vectors. We introduce these concepts in the following. First, we define the squared error between two vectors $v, w \in \mathbb{R}^d$ as

$$d_{\text{se}}(v, w) = \|v - w\|_2^2 = (v - w)^\top (v - w) = \sum_{j=1}^d (v[j] - w[j])^2. \quad (6)$$

Note that the squared error is the same as the squared Euclidean norm of the residuals. Sometimes, the vectors v and w are normalized before the squared error is calculated. We denote the resulting distance metric as normalized squared error

$$d_{\text{nse}}(v, w) = \left\| \frac{v}{\|v\|_2} - \frac{w}{\|w\|_2} \right\|_2^2. \quad (7)$$

We define the cosine similarity as

$$s_{\text{cos}}(v, w) = \frac{v^\top w}{\|v\|_2 \|w\|_2}. \quad (8)$$

Note that d_{nse} and s_{cos} are linearly related, i.e.,

$$d_{\text{nse}}(v, w) = 2 - 2 \frac{v^\top w}{\|v\|_2 \|w\|_2} = 2 - 2 s_{\text{cos}}(v, w) \quad (9)$$

$$\Leftrightarrow s_{\text{cos}}(v, w) = 1 - \frac{1}{2} d_{\text{nse}}(v, w). \quad (10)$$

To measure the distance between two discrete probability distributions described by the entries of two vectors v and w , the cross-entropy can be used which is given as

$$d_{\text{ce}}(v, w) = - \sum_{j=1}^d v[j] \log w[j]. \quad (11)$$

Note that the cross-entropy does not fulfill the triangle inequality and is therefore no distance metric in a mathematical sense. A common loss function for multiclass classification is the cross-entropy between scores $v \in \mathbb{R}^d$ that are normalized using softmax and a one-hot distribution of the true class label $c \in \mathbb{N}$, which we denote by

$$d_{\text{classification}}(v, c) = d_{\text{ce}}(\text{onehot}(c), \text{softmax}(v)) \quad (12)$$

$$= -v[c] + \log \left(\sum_{j=1}^d \exp v[j] \right), \quad (13)$$

where $\text{onehot}(c)$ is a vector with 1 in the c -th component and 0 everywhere else.

2 Pretext task methods

In the introduction we have defined the concept of self-supervised learning which relies on defining a mechanism that creates targets for a supervised learning task. There are many possibilities to invent such supervised learning problems. These learning problems are usually called *pretext* tasks. The idea is that the features learned by solving the pretext task are also helpful for solving other problems. In the following we present works, that creatively came up with such tasks.

2.1 Autoencoders

Autoencoders (Le Cun, 1987) have been part of machine learning for a very long time and in the light of the previous section, they can be seen as early instances of self-supervised learning: (1) the invented targets are the inputs themselves and (2) the learned function is a bottleneck neural network consisting of an encoder f_θ that maps an image x_i to a low-dimensional representation $y_i = f_\theta(x_i)$ and a predictor q_ψ that reconstructs the input image $\hat{x}_i = q_\psi(y_i)$ from its representation. Figure 1 shows the encoder and the predictor network.

Given a batch of images X , the encoder and predictor networks are jointly trained to minimize the reconstruction error

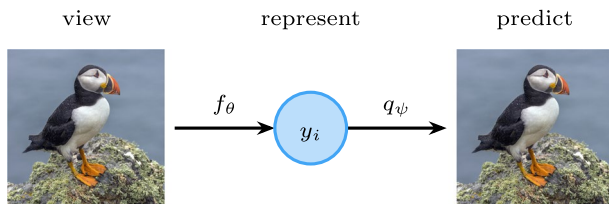
$$\mathcal{L}_{\theta, \psi}^{\text{AE}} = \frac{1}{n} \sum_{i=1}^n d_{\text{sc}}(\hat{x}_i, x_i). \quad (14)$$

There are many variants of the autoencoder: denoising autoencoder (Le Cun, 1987), stacked denoising autoencoder (Vincent et al., 2010), contractive autoencoder (Rifai et al., 2011) or variational autoencoder (VAE, Kingma & Welling, 2013).

2.2 Predicting image rotations (RotNet)

The idea of the RotNet which was introduced by Gidaris et al. (2018) is to learn a representation that is useful to predict the angle of a random rotation applied to the input image. The assumption is that a representation that can predict the rotation is also valuable for other tasks. The authors show that even a small number of rotations is sufficient to learn

Fig. 1 An autoencoder consists of two networks: an encoder f_θ that maps the input image to a representation and a predictor q_ψ that is trained to reconstruct the original image from the representation



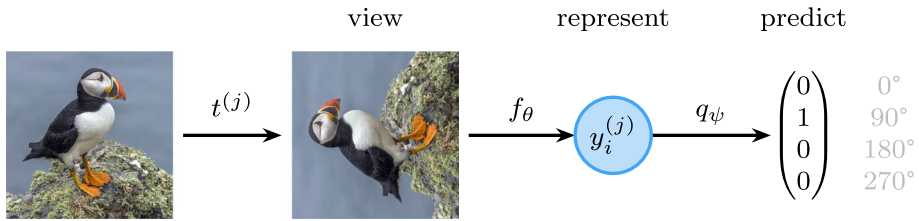


Fig. 2 RotNet solves the task of predicting image rotations to obtain representations for downstream tasks

a good representation. Best results are obtained when the number of rotations is four (0° , 90° , 180° , 270°). In that case the rotation augmentation can be efficiently implemented using flips and transposes and no interpolation is needed.

Given a batch of images X , we consider a single image x_i . Four views $x_i^{(j)} = t^{(j)}(x_i)$ are created using the rotation transformations $t^{(1)}$, $t^{(2)}$, $t^{(3)}$, $t^{(4)}$. The Siamese encoder f_θ converts each view into a representation $y_i^{(j)} = f_\theta(x_i^{(j)})$. The Siamese predictor q_ψ is then used to predict the index of the rotation that was applied to the original image. Both networks are trained by minimizing the classification loss

$$\mathcal{L}_{\theta, \psi}^{\text{RotNet}} = \frac{1}{n} \sum_{i=1}^n \sum_{c=1}^4 d_{\text{classification}}(q_\psi(y_i^{(c)}), c) \quad (15)$$

for each of the four rotations. After training, the classification head q_ψ is discarded and only f_θ is used to calculate representations (of images that were not rotated). Figure 2 shows the architecture of a RotNet. The authors use a Network-in-Network architecture (Lin et al., 2013) for their experiments on CIFAR-10 and the AlexNet architecture (Krizhevsky et al., 2017) for experiments on ImageNet.

2.3 Solving Jigsaw puzzles

Jigsaw (Noroozi & Favaro, 2016) is similar to RotNet in the sense that it also solves a classification task. However, instead of rotating the image, the transformation consists of randomly permuting several patches of the image like a jigsaw puzzle. The pretext task of the model is then to predict the class of the permutation that was used to shuffle the patches. To facilitate the task, it is necessary to restrict the used permutations to a subset of all permutations. In their work the authors use 1000 predefined permutations (instead of $9! = 362880$ for a 3×3 grid of tiles).

From an input image x_i nine non-overlapping randomly permuted patches $[x_i^{(1)}, \dots, x_i^{(9)}]$ are extracted, where the order of the patches follows one of the predefined permutations. After that, the Siamese encoder f_θ converts each patch into a representation $y_i^{(j)} = f_\theta(x_i^{(j)})$. The predictor q_ψ is used to predict the index c_i of the permutation that was applied to the original image, given all patch representations $Y_i = [y_i^{(1)}, \dots, y_i^{(9)}]$ at once. The networks are trained by minimizing the loss

$$\mathcal{L}_{\theta, \psi}^{\text{Jigsaw}} = \frac{1}{n} \sum_{i=1}^n d_{\text{classification}}(q_\psi(Y_i), c_i) \quad (16)$$

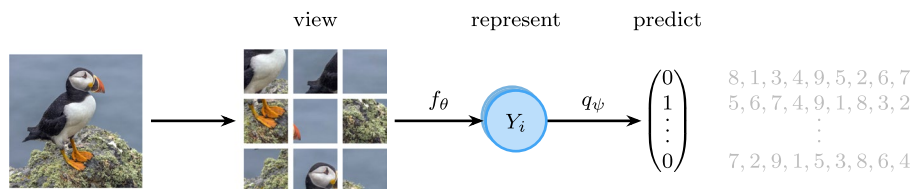


Fig. 3 Jigsaw extracts patches from an image that are then permuted. The pretext task is to find the permutation that was used to permute the images. The context-free network f_θ processes each patch separately and the representations are only joined in the latter layers

between the class scores and the index of the used permutation c_i . For an overview of the Jigsaw architecture we refer to Fig. 3. The encoder f_θ is implemented as a truncated AlexNet. The representations Y_i are concatenated to form the input of the classification head q_ψ , which is implemented as a multi-layer perceptron (MLP). After training, the classification head is discarded and the encoder is used to obtain image representations for other downstream task.

2.4 Masked autoencoders (MAE)

Masked autoencoders (He et al., 2022) are similar to denoising autoencoders, where the input images are corrupted and the autoencoder tries to reconstruct the original image. More specifically, the input image is split into smaller non-overlapping patches, from which a random subset is masked out.

Given a batch of images X , we consider a single image x_i . The image is split up into m patches $X_i = [x_i^{(1)}, \dots, x_i^{(m)}]$, some of which will be randomly chosen to be masked. We call the set of indices of the masked patches M_i^{mask} and the set of unmasked indices M_i^{keep} . The encoder f_θ converts the *unmasked* patches into patch representations $Y_i = [y_i^{(j)} : j \in M_i^{\text{keep}}] = f_\theta([x_i^{(j)} : j \in M_i^{\text{keep}}])$, implemented as a vision transformer (Dosovitskiy et al., 2021). The predictor q_ψ is another vision transformer that predicts the *masked* patches $\hat{X}_i = [\hat{x}_i^{(j)} : j \in M_i^{\text{mask}}] = q_\psi(Y_i, M_i^{\text{mask}})$ from Y_i with a mask token for each index in M_i^{mask} . See Fig. 4 for an illustration. The loss is the mean squared error between the pixels of the predicted patches and the masked patches

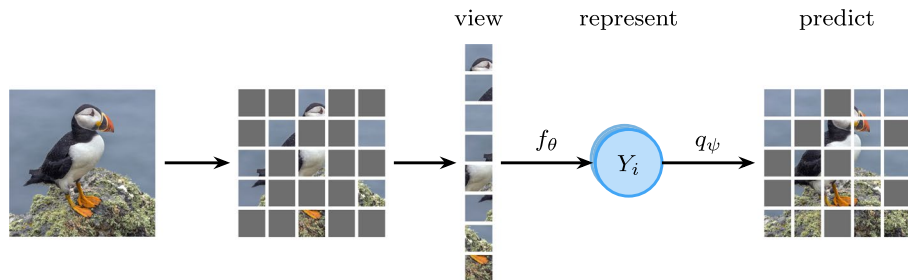


Fig. 4 The masked autoencoder predicts masked patches using vision transformers for the encoder and predictor

$$\mathcal{L}_{\theta, \psi}^{\text{MAE}} = \frac{1}{n} \sum_{i=1}^n \sum_{j \in M_i^{\text{mask}}} d_{\text{sc}}(\hat{x}_i^{(j)}, x_i^{(j)}). \quad (17)$$

Without precaution the model could “cheat” by predicting image patches from neighboring pixels, since the information in natural images is usually very spatially redundant (He et al., 2022). To learn good representations, it is crucial that the masking ratio is very high, e.g., 75%, to encourage the encoder to extract more high-level features (He et al., 2022).

2.5 Advantages and disadvantages

Most pretext-task methods are easy to understand and very intuitive. However it is somewhat unclear how the features learned while solving the pretext task actually transfer to other applications. It is possible that the method overfits to the pretext task and does not generalize to other tasks like image classification. This can for example be the case for the Jigsaw method. Noroozi and Favaro (2016) show that in some situations their method can take shortcuts by using edge statistics as a proxy to solve the problem instead of focusing on relevant image features. Furthermore, Jigsaw puzzles have a high complexity especially when considering puzzles with more tiles.

Another disadvantage is that the performance of the RotNet and the Jigsaw are not on the same level as newer methods that will be discussed in the remainder of the paper. The autoencoder also is not comparable in performance with state-of-the-art methods and is only included in this survey for completeness.

2.6 Relations to other methods

The autoencoder is the earliest representation learning method and highly influenced other representation learning methods: many of the methods that we are discussing in this work are maximizing a cosine similarity measure with is directly related with the mean-squared error that is minimized by training an autoencoder. While we are not aware of methods that also use image rotations as they are used by the RotNet, many many use transformations like horizontal flipping, blurring or random cropping, as we will see in the next section. The idea of jigsaw puzzles has been picked up by Misra and Maaten (2020) in their pretext-invariant representation learning (PIRL) method. We will discuss this method in detail in Sect. 5.3.6.

3 Information maximization methods

Many self-supervised representation learning methods make use of image transformations. Jigsaw and Rotation Networks, two approaches from the preceding section, apply selected transformations to image examples with the aim to predict the transformation’s parametrization. In contrast, the following methods focus on learning representations that are invariant to certain transformations. Such a task typically entails a popular failure mode, called *representation collapse*. It commonly describes trivial solutions, e.g., constant representations, that satisfy the invariance objective, but provide little to no informational value for actual downstream tasks. Instead of learning a rich and diverse feature presentation of the data, the representations *collapse* to a less informative state, often by mapping different or

unrelated data points to the same outputs. With that, representation collapse can also be viewed as information collapse in the context of information theory, concentrating a majority of the probability mass of the embedding in a single point, which leads to a decrease of information content. There are several factors that facilitate the occurrence of information collapse, including insufficient regularization and penalization of uninformative representations in the loss function.

To avoid the representation collapse the so-called *information maximization* methods have been developed. They form a class of representation techniques that focus on the information content of the embeddings (Zbontar et al., 2021; Bardes et al., 2021; Ermolov et al., 2021). Due to the employed techniques that aim to maximize information, some of these methods may also be referred to as *feature decorrelation* methods (Tao et al., 2022), as they explicitly decorrelate all elements of embedding vectors. This effectively avoids collapse and results in an indirect maximization of information content. In the following, we present methods that implement this technique using the normalized cross-correlation matrix of embeddings across views (Zbontar et al., 2021), the covariance matrix for single views (Bardes et al., 2021), as well as a whitening operation (Ermolov et al., 2021).

3.1 Transformations

The main idea of information maximization methods is that the learned representations should be invariant with respect to certain transformations, i.e., the original image and the transformed images should yield the same representations. We have already encountered two transformations in the previous section, the rotation and jigsaw transformation. There are many more transformations possible: the following transformations have been proven useful for the next methods to be described.

1. Horizontal flipping: Most natural images can be flipped horizontally without changing the semantics, e.g., an image of a car still shows a car after being flipped. Vertical flipping can cause problems when for example the sky is suddenly at the bottom of the image.
2. Blurring: Convolving an image with a Gaussian filter is another way to transform an image.
3. Adding Gaussian noise: Learned representations should also be (to some extent) invariant to the application of noise.
4. Sobel filter: Applying a Sobel filter to an image highlights the edges of an image. These edges usually still contain a lot of relevant information about the image.
5. Cropping and resizing: Scaling the image to a different size should also keep the semantic information.
6. Color jittering: Changing the contrast, brightness and hue of an image yields another instance of an image that shows the same content.
7. Grayscale: Converting color-images to grayscale images is closely related to color jittering.

Figure 5 demonstrates the effect of the presented transformations on an image of a puffin. Note that these image transformations are closely related to dataset augmentation techniques used in supervised learning (Yang et al., 2022; Shorten & Khoshgoftaar, 2019).

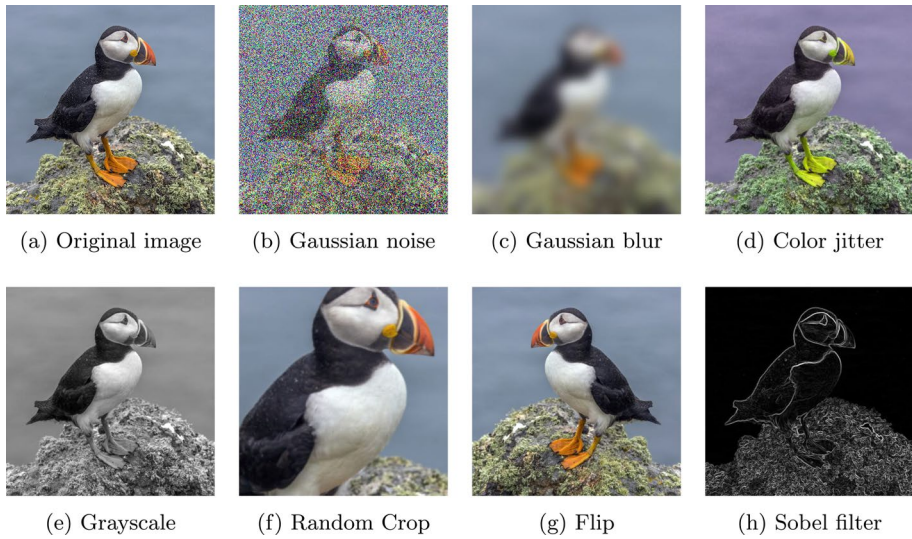


Fig. 5 Example transformations applied to an image of a puffin

3.2 Barlow twins

The central idea behind this framework is the principle of redundancy reduction. This principle was introduced by the neuroscientist Barlow (1961) and states that the reduction of redundancy is important for the organization of sensory messages in the brain.

To implement this redundancy reduction principle, the Barlow Twins approach takes a batch of images X and creates two views $X^{(1)} = t^{(1)}(X)$ and $X^{(2)} = t^{(2)}(X)$ of these images, where $t^{(1)}, t^{(2)} \sim \mathcal{T}$ are transformations that is randomly sampled from $\mathcal{T}$ for every image of the batch. A Siamese encoder f_θ computes representations $Y^{(1)} = f_\theta(X^{(1)})$ and $Y^{(2)} = f_\theta(X^{(2)})$, which are fed into a Siamese projector g_ϕ to compute projections $Z^{(1)} = [z_1^{(1)}, \dots, z_n^{(1)}] = g_\phi(Y^{(1)})$ and $Z^{(2)} = [z_1^{(2)}, \dots, z_n^{(2)}] = g_\phi(Y^{(2)})$ for both views. Figure 6 shows the Barlow twins architecture.

The idea of Barlow Twins is to regularize the cross-correlation matrix between the projections of both views. The cross-correlation matrix is calculated as

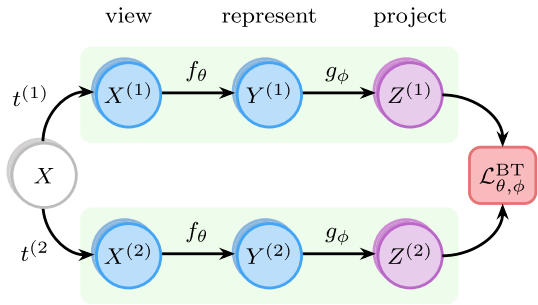
$$C = \frac{1}{n} \sum_{i=1}^n \left((z_i^{(1)} - \mu^{(1)}) / \sigma^{(1)} \right) \left((z_i^{(2)} - \mu^{(2)}) / \sigma^{(2)} \right)^\top, \quad (18)$$

where $\mu^{(j)}$ and $\sigma^{(j)}$ are the mean and standard deviation over the batch of projections of the j -th view, calculated as

$$\mu^{(j)} = \frac{1}{n} \sum_{i=1}^n z_i^{(j)}, \quad (19)$$

$$\sigma^{(j)} = \sqrt{\frac{1}{n-1} \sum_{i=1}^n (z_i^{(j)} - \mu^{(j)})^2}. \quad (20)$$

Fig. 6 Barlow twins takes two views of the input batch and minimizes the correlation of the projected representations



The loss function is then defined as

$$\mathcal{L}_{\theta, \phi}^{\text{BT}} = \sum_{k=1}^d (1 - C[k, k])^2 + \lambda \sum_{k=1}^d \sum_{k' \neq k} C[k, k']^2, \quad (21)$$

where d is the number of dimensions of the projection and $\lambda > 0$ is a hyperparameter. The first term promotes invariance with regard to the applied transformations and the second term decorrelates the learned embeddings, i.e., reduces redundancy. By using this loss, the encoder f_θ is encouraged to predict embeddings that are decorrelated and thereby non-redundant. The Barlow Twins are trained using the LARS optimizer (You et al., 2017). Note that this loss function is related to the VICReg method, which we are going to introduce next: the first term is called *variance* term and the second *covariance* term.

3.3 Variance-invariance-covariance regularization (VICReg)

VICReg was introduced by Bardes et al. (2021) and is a joint-embedding architecture that falls into the category of information maximization methods. Figure 7 gives an overview of the architecture, which is identical to Barlow twins, but uses a different loss function. It aims to maximize agreement between representations of different views of an input, while preventing *informational collapse* using two additional regularization terms. Specifically, VICReg defines regularization terms for variance, invariance and covariance.

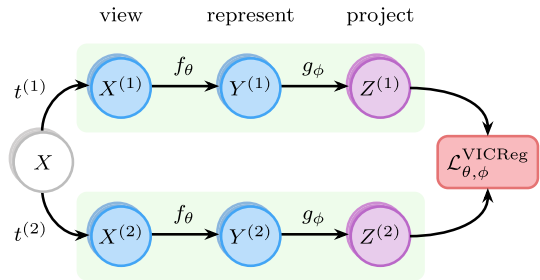
Given a batch of images X , two views $X^{(1)} = t^{(1)}(X)$ and $X^{(2)} = t^{(2)}(X)$ are defined, where $t^{(1)}, t^{(2)} \sim \mathcal{T}$ are, again, randomly sampled from $\mathcal{T}$ for every image of the batch. A Siamese encoder f_θ computes representations $Y^{(1)} = f_\theta(X^{(1)})$ and $Y^{(2)} = f_\theta(X^{(2)})$, which are fed into a Siamese projector g_ϕ to compute projections $Z^{(1)} = [z_1^{(1)}, \dots, z_n^{(1)}] = g_\phi(Y^{(1)})$ and $Z^{(2)} = [z_1^{(2)}, \dots, z_n^{(2)}] = g_\phi(Y^{(2)})$. Each projection has d dimensions. For each view, the covariance matrix of the projections is computed, which is defined as

$$C^{(j)} = \frac{1}{n-1} \sum_{i=1}^n (z_i^{(j)} - \mu^{(j)}) (z_i^{(j)} - \mu^{(j)})^\top, \quad (22)$$

where $\mu^{(j)}$ is the mean over the batch of projections of the j -th view, i.e.,

$$\mu^{(j)} = \frac{1}{n} \sum_{i=1}^n z_i^{(j)}. \quad (23)$$

Fig. 7 VICReg takes two views of the input batch and minimizes the mean squared error between the projected representations, while regularizing the covariance matrices of representations from individual views to avoid representation collapse



The *variance* term aims to keep the standard deviation of each element of the embedding across the batch dimension above a margin b . Practically, this prevents embedding vectors to be the same across the batch and thus is one of the two mechanisms that intent to prevent collapse. It can be implemented using a hinge loss

$$\ell_v(Z^{(j)}) = \frac{1}{d} \sum_{k=1}^d \max\left(0, b - \sqrt{C^{(j)}[k, k] + \varepsilon}\right), \quad (24)$$

where $\varepsilon > 0$ is a small hyperparameter for numerical stability. Bardes et al. (2021) used $b = 1$. On that note, the variance term is closely related to the invariance term of Barlow Twins (Zbontar et al., 2021), but applied with a different intention. While Barlow Twins practically maximizes the squared diagonals of the normalized cross-correlation matrix to encourage correlation of embedding elements across views, VICReg maximizes the square root of the diagonals of the covariance matrix of single views in order to prevent collapse. Note, that the maximization in Barlow Twins is restricted by the preceding normalization of the embeddings. As VICReg does not apply a normalization, the margin loss is used to restrict this optimization.

The *covariance* term decorrelates elements of embedding vectors for single views in order to reduce redundancy and prevent collapse. This is achieved by minimizing the squared off-diagonal elements of the covariance matrix $C^{(j)}$ towards 0, i.e.,

$$\ell_c(Z^{(j)}) = \frac{1}{d} \sum_{k=1}^d \sum_{k' \neq k} (C^{(j)}[k, k'])^2. \quad (25)$$

Note, that this is similar to the redundancy reduction term used in Barlow Twins (Eq. 21, right summand), the main difference being again that Barlow Twins applies it across views, but with a similar intention.

Finally, the *invariance* term is used to maximize the agreement between two projections $z_i^{(1)}$ and $z_i^{(2)}$ of the same image, thus inducing invariance to the transformations that were applied to x_i . For this, Bardes et al. (2021) apply the mean squared error between the projections

$$\ell_1(Z^{(1)}, Z^{(2)}) = \frac{1}{n} \sum_{i=1}^n d_{\text{se}}\left(z_i^{(1)}, z_i^{(2)}\right). \quad (26)$$

Notably, it is the only loss term in VICReg operating across different views.

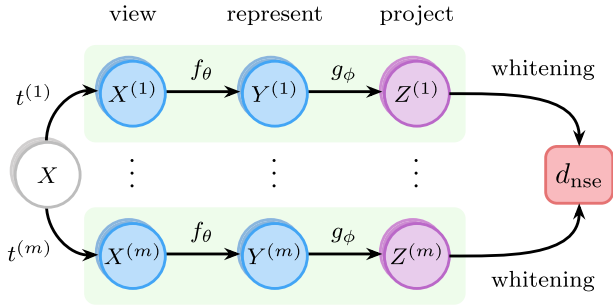


Fig. 8 WMSE training: The batch of input images X are randomly transformed and fed into the encoder network f_θ . The representations are then projected using the projection head g_ϕ . Next, whitening is applied to the projections. The networks are trained by minimizing the normalized MSE between the projections

Overall, the loss of VICReg can be defined as the weighted sum of all three regularizations for the given views

$$\mathcal{L}_{\theta, \phi}^{\text{VICReg}}(X) = \lambda_V[\ell_V(Z^{(1)}) + \ell_V(Z^{(2)})] + \lambda_C[\ell_C(Z^{(1)}) + \ell_C(Z^{(2)})] + \lambda_I \ell_I(Z^{(1)}, Z^{(2)}), \quad (27)$$

where $\lambda_V, \lambda_I, \lambda_C > 0$ are hyperparameters that balance the individual losses.

3.4 Whitening (WMSE)

Whitening linearly transforms a set of data points, such that the resulting data points are decorrelated and have unit variance, i.e., the covariance matrix becomes the identity matrix. The method WMSE (Ermolov et al., 2021) applies this idea to the embeddings of images to prevent the representation collapse.

Given a batch of images X , random transformations $t^{(1)}, \dots, t^{(m)}$ are applied to obtain m views $X^{(j)}$ for all $j \in \{1, \dots, m\}$. A Siamese encoder f_θ maps the views onto representations $Y^{(j)} = f_\theta(X^{(j)})$, which are then fed into a Siamese projector g_ϕ to compute projections $Z^{(j)} = [z_1^{(j)}, \dots, z_n^{(j)}] = g_\phi(Y^{(j)})$. All projections are then concatenated into a single matrix $Z = [z_1^{(1)}, \dots, z_n^{(1)}, \dots, z_1^{(m)}, \dots, z_n^{(m)}]$. This matrix is *whitened* to obtain $\tilde{Z}$ by removing the mean and decorrelating it using the Cholesky decomposition of the inverse covariance matrix, i.e.,

$$\tilde{Z} = [z_1^{(1)}, \dots, z_n^{(1)}, \dots, z_1^{(m)}, \dots, z_n^{(m)}] = W_Z \left(Z - \frac{1}{nm} \sum_{i=1}^n \sum_{j=1}^m z_i^{(j)} \mathbf{1}_{nm} \right), \quad (28)$$

where $\mathbf{1}_{nm}$ is an all-ones vector with nm entries and W_Z is the Cholesky factor of the inverse covariance matrix P^{ZZ} (also known as the precision matrix), i.e., $W_Z W_Z^\top = P^{ZZ}$. Note that the Cholesky decomposition is differentiable which allows to backpropagate through it during training. The architecture is shown in Fig. 8.

To train the models, the normalized squared error between all pairs of the whitened projections is minimized, i.e., the loss function is defined as

$$\mathcal{L}_{\theta, \phi}^{\text{WMSE}} = \frac{1}{n} \sum_{i=1}^n \frac{2}{m(m-1)} \sum_{j=1}^m \sum_{k=j+1}^m d_{\text{nsc}}\left(\tilde{z}_i^{(j)}, \tilde{z}_i^{(k)}\right). \quad (29)$$

The constant $2/(m(m-1))$ is the number of comparison per image. The whitening step is essential to prevent the representations from collapsing. The objective maximizes the similarity between all augmented pairs while also preventing representation collapse by enforcing unit covariance on the projections.

The proposed method introduces an alternative to prevent representation collapse using a whitening transform and ensures fast training without asymmetric networks (Ermolov et al., 2021). By allowing multiple positive pairs and using simple network designs, it achieves competitive performance across various datasets and is applied to model-based reinforcement learning (Ermolov & Sebe, 2020). However, during training occurs an optimization instability which needs to be addressed.

3.4.1 Batch slicing

One problem of the original method is that the loss has a large variance over consecutive training batches. To counteract this issue, Ermolov et al. (2021) employ so-called batch slicing: the idea of batch slicing is that different views of the same image $z_i^{(1)}, z_i^{(2)}, \dots, z_i^{(m)}$ should not be in the same batch when the whitening matrix is calculated. For this, Z is partitioned into m parts. The elements of each part are then permuted using the same permutation for each of the m parts. Finally, each of the parts is further subdivided into d subsets which are then used to calculate the whitening matrix for that specific subset. In that way, the loss minimization is dependent on $m \cdot d$ covariance matrices that need to satisfy the identity, leading empirically to lower variance.

3.5 Advantages and disadvantages

Information maximization methods effectively mitigate representation collapse in self-supervised learning by maximizing the information content of embeddings and reducing feature redundancy. Techniques such as feature decorrelation or regularization allow to directly supervise desired properties of the representation learning system, yielding robust learned embedding spaces that are suitable for various downstream tasks, such as image classification, object detection and instance segmentation. Notably, regularization can also be applied complementarily to other self-supervised methods, as demonstrated by Bardes et al. (2021). While it is generally important to perform hyperparameter tuning, both VICReg and Barlow Twins report relative robustness to smaller batch sizes (Zbontar et al., 2021; Bardes et al., 2021), which is especially beneficial for scenarios with limited computational resources. This is in contrast to other self-supervised learning methods, like SimCLR (Chen et al., 2020a), that often require larger batch sizes to stabilize learning and prevent collapse, making VICReg and Barlow Twins more adaptable to diverse training environments. Like other methods that rely on the similarity relations between different views of the same data, information maximization methods heavily depend on the use of adequate data augmentations used to create such varied views. The choice and quality of data augmentation strategies are crucial in promoting learned invariance, thereby playing a significant role in influencing the performance of models on downstream tasks.

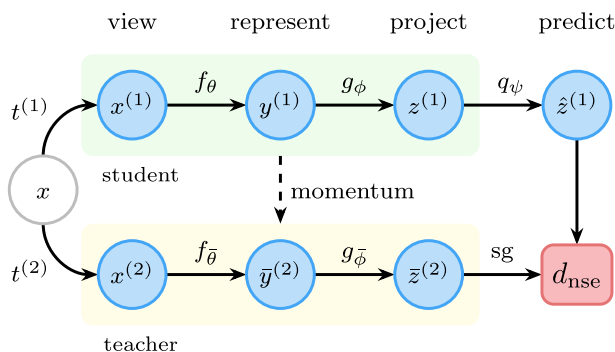


Fig. 9 BYOL consists of a student and teacher network. The teacher network is not updated via gradient descent (stop-gradient) and thus provides stable representations for the student network to learn. The teacher network is updated iteratively via an exponential moving average of the student. The student branch has a predictor which is trained to match the fixed representations of the teacher

3.6 Relations to other methods

Pretext task methods and information maximization methods both provide training procedures that allow models to learn meaningful and useful features from input data without the need for explicit labels or other additional information. Pretext task methods, like autoencoders, describe mechanisms that already entail relations between data points that qualify for meaningful supervision. In that context, information maximization methods also explore specific mechanisms that leverage the similarity relation between multiple views of the same example by maximizing mutual information.

4 Teacher–student methods

Methods based on teacher-student learning are closely related to information maximization methods. Similar to information maximization methods, the student learns to predict the teacher's representations across different image transformations. This allows the student to learn invariant representations that are robust to different transformations of the same image. These methods consist of two branches, where one is considered the student and the other the teacher. To prevent representational collapse as defined in Sect. 3, the teacher provides stable target representations for the student to predict. To provide stable targets, the teacher is not updated using gradient descent and its parameters are fixed when updating the student. Sometimes a momentum encoder is used between the teacher and student to update the fixed targets. That is, the weights from the student are slowly copied to the teacher to provide more recent targets. The teacher usually has the same architecture as the student, but does not necessarily have the same parameters. The teacher can be a running average of the student's representations, where, e.g., a momentum encoder is used to update the teacher network with the student's weights. For some teacher-student methods the teacher shares the student's weights and an additional predictor network has to predict the teacher's representation.

4.1 Bootstrap your own latent (BYOL)

BYOL (Grill et al., 2020) is inspired by the observation that learning representations by predicting fixed representations from a randomly initialized target network avoids representational collapse albeit with subpar performance. This naturally entails a teacher-student architecture where the teacher (target network) provides stable representations for the student (online network) to learn on.

BYOL defines two different networks: a student network and a teacher network. The architecture is shown in Fig. 9, the student network and teacher network consist of the following parts:

- Student network: encoder f_θ , projector g_ϕ , predictor q_ψ
- Teacher network: encoder $f_{\bar{\theta}}$, projector $g_{\bar{\phi}}$

The encoder f and projector g are present in both the student and teacher networks, whereas the predictor q is only part of the student network.

4.1.1 Learning augmentation-invariant features from the teacher

Like information-maximization methods, teacher-student methods learn representations by applying different transformations to the images (see Sect. 3). Given an image x_i , BYOL applies randomly-sampled transformations $t^{(1)}, t^{(2)} \sim \mathcal{T}$ to obtain two different views $x_i^{(1)} = t^{(1)}(x_i)$ and $x_i^{(2)} = t^{(2)}(x_i)$. The student network computes representations $y_i^{(j)} = f_\theta(x_i^{(j)})$, projections $z_i^{(j)} = g_\phi(y_i^{(j)})$, and predictions $\hat{z}_i^{(j)} = q_\psi(z_i^{(j)})$ for both views $j \in \{1, 2\}$. The views are also fed to the teacher network to obtain target projections $\bar{z}_i^{(1)} = g_{\bar{\phi}}(f_{\bar{\theta}}(x_i^{(1)}))$ and $\bar{z}_i^{(2)} = g_{\bar{\phi}}(f_{\bar{\theta}}(x_i^{(2)}))$.

BYOL minimizes two normalized squared errors: (1) between the prediction of the first view and the target projection of the second view, (2) between the prediction of the second view and the target projection of the first view. The final loss function is

$$\mathcal{L}_{\theta, \phi, \psi}^{\text{BYOL}} = \frac{1}{n} \sum_{i=1}^n \left[d_{\text{nse}}(\hat{z}_i^{(1)}, \bar{z}_i^{(2)}) + d_{\text{nse}}(\hat{z}_i^{(2)}, \bar{z}_i^{(1)}) \right]. \quad (30)$$

Note that the loss is minimal when the cosine similarity between the vectors is 1. Thus, representations are learned that are similar for two different transformations. In other words, the information content in the learned representations is maximized.

4.1.2 Teacher-student momentum encoder

At each training step, the loss is minimized with respect to θ , ϕ , and ψ . That is, only the weights of the student are updated by the gradient of the loss function using the LARS optimizer (You et al., 2017). The weights of the teacher are updated by the exponential moving average (Lillicrap et al., 2019), i.e.,

$$\bar{\theta} \leftarrow \tau \bar{\theta} + (1 - \tau) \theta, \quad (31)$$

$$\bar{\phi} \leftarrow \tau \bar{\phi} + (1 - \tau) \phi, \quad (32)$$

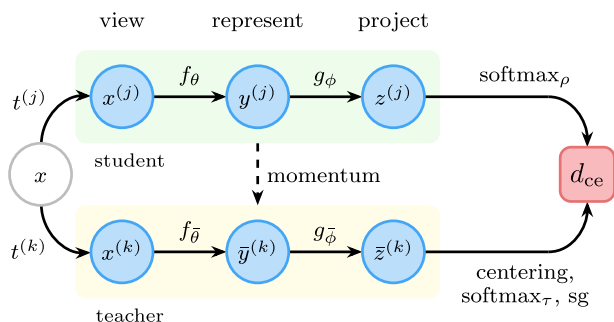


Fig. 10 DINO consists of two ViTs, one acting as a student and the other as teacher. The embeddings of the ViT's are transformed into a Softmax distribution which produces soft labels for both the student and the teacher. The knowledge of the student is then iteratively distilled onto the teacher, which provides stable targets for the student

where $\tau \in [0, 1]$ controls the rate at which the weights of the teacher network are updated with the weights of the student network.

The authors show that BYOL's success relies on two key components: (1) keeping the predictor q_ψ near optimal at all times by predicting the stable target representations, and (2) updating the parameters in the direction of $\nabla_{\theta, \phi, \psi} \mathcal{L}^{\text{BYOL}}$ and not in the direction of $\nabla_{\bar{\theta}, \bar{\phi}} \mathcal{L}^{\text{BYOL}}$. In other words, the proposed loss and update do not jointly optimize the loss over θ, ϕ and $\bar{\theta}, \bar{\phi}$, which would lead to a representation collapse. Chen and He (2021) provide further insight on how this is related to the predictor: regarding (1) it is observed that keeping the learning rate of the predictor fixed instead of decaying it during training improves performance, which supports the fact that the predictor should learn the latest representations. Regarding (2) Chen and He (2021) use a predictor that maps the projection to the identity. They argue that the gradient of the symmetrized loss with an identity predictor, i.e., between the two projections cancels out the stop-gradient operator. In this case the gradient of the symmetrized loss between the two projections is in the same direction, hence leading to a collapsed representation. Note that this analysis was performed for SimSiam (Chen & He, 2021), which does not use a momentum encoder. It is feasible however that the predictor plays the same role for BYOL. Grill et al. (2020) show that one of the key advantages of BYOL is that no negative samples are required in the training process, allowing BYOL to be more robust against the choice of image augmentations and to work with smaller batch sizes. This however comes at the cost of a complex asymmetric teacher-student architecture using momentum encoding and a separate predictor, which is complex and thus is prone to mode collapse as shown by Chen and He (2021) and Pham et al. (2022). Overall, the effect of each of these components is currently not well understood and still an active area of research.

4.2 Self-distillation with no labels (DINO)

One of the main contributions of DINO (Caron et al., 2021) is adapting the teacher-student architecture closer to the knowledge distillation framework (Gou et al., 2021), where instead of matching the output embeddings directly, the teacher provides soft labels by applying a softmax operation on its output. The authors show that this facilitates preventing a representation collapse.



Fig. 11 Two sets of augmented crops for the teacher and student as defined by the multi-crop augmentation strategy. The teacher set contains two global views which cover at least 50% of the augmented image. The student set is made up of k patches that cover less than 50% of the other augmented version of the input image

DINO defines a student and a teacher network (see Fig. 10). The student consists of an encoder f_θ and a projector g_ϕ with parameters θ and ϕ . The encoder is implemented as a vision transformer (ViT, Dosovitskiy et al., 2021) and the projector as an MLP. The teacher consists of an encoder $f_{\bar{\theta}}$ and a projector $g_{\bar{\phi}}$ with the same architecture as the student, but a separate set of parameters $\bar{\theta}$ and $\bar{\phi}$.

DINO uses a multi-crop strategy first proposed by Caron et al. (2020) to create a batch of m views $X_i = [x_i^{(1)}, \dots, x_i^{(m)}]$ of an image x_i . Each view is a random crop of x_i followed by more transformations. Most crops cover a small region of the image, but some crops are of high resolution, which we refer to as *local* and *global* views, respectively. Let M_i be the set of indices of the global views. The idea is that the student has access to all views, while the teacher only has access to the global views, which creates “local-to-global” correspondences (Caron et al., 2021). See Fig. 11 for an illustration.

The student computes representations $y_i^{(j)} = f_\theta(x_i^{(j)})$ and projections $z_i^{(j)} = g_\phi(y_i^{(j)})$ for each view. The teacher computes target projections $\bar{z}_i^{(j)} = g_{\bar{\phi}}(f_{\bar{\theta}}(x_i^{(j)}))$ for the global views $j \in M_i$.

4.2.1 Preventing collapse

Caron et al. (2021) experimentally discover two forms of collapse: Either the computed probability distribution is uniform, or one dimension dominates, regardless of the input. This motivates two countermeasures:

1. To prevent collapse to a uniform distribution, the target distribution of the teacher is *sharpened* by setting the temperature parameter τ to a small value.
2. To prevent one dimension from dominating, the output of the teacher is centered to make it more uniform. This is accomplished by adding a centering vector c as a bias to the teacher, which is computed with an exponentially moving average

$$c \leftarrow \beta c + (1 - \beta) \bar{z}, \quad (33)$$

where $\beta \in [0, 1]$ is a decay hyperparameter determining to what extent the centering vector is updated and

$$\bar{z} = \frac{1}{n} \sum_{i=1}^n \frac{1}{|M_i|} \sum_{j \in M_i} \bar{z}_i^{(j)} \quad (34)$$

is the mean of all projections of the teacher in the current batch.

4.2.2 Learning invariant features via soft labels

DINO formulates the task of predicting the target projections as a knowledge distillation task. The projections of the teacher and the student are converted to probability distributions by applying the softmax function over all components. Hereby, the cross-entropy loss can be applied, where the teacher computes soft labels for the student. The total loss function matches every view of the student to every global view of the teacher (except the same global views), i.e.,

$$\mathcal{L}_{\theta, \phi}^{\text{DINO}} = \frac{1}{n} \sum_{i=1}^n \sum_{j \in M_i} \sum_{k \neq j} d_{\text{ce}}(\text{softmax}_{\tau}(\bar{z}_i^{(j)} - c), \text{softmax}_{\rho}(z_i^{(k)})), \quad (35)$$

where $\tau, \rho > 0$ are hyperparameters controlling the temperature of the distributions (see Sect. 1.5) for the teacher and the student, respectively. Overall, the parameter updates are very similar to BYOL's, since the student network is updated by minimizing the loss $\mathcal{L}_{\theta, \phi}^{\text{DINO}}$ using the AdamW optimizer, and the teacher network is updated by an exponential moving average of the student, i.e.,

$$\bar{\theta} \leftarrow \alpha \bar{\theta} + (1 - \alpha) \theta, \quad (36)$$

$$\bar{\phi} \leftarrow \alpha \bar{\phi} + (1 - \alpha) \phi, \quad (37)$$

where $\alpha \in [0, 1]$ controls the rate at which the weights of the teacher network are updated with the weights of the student network.

A key advantage of DINO as shown by Caron et al. (2021) is that the trained attention masks are able to detect complex features within a scene. These embeddings can then be used directly, e.g., as segmentation masks, that allow DINO to perform well on downstream tasks simply by using a k-nearest-neighbor classifier on its representations. An architectural simplification by Caron et al. (2021) is to remove the predictor present in BYOL and SimSiam, while relying on the centering and sharpening of the teacher output to prevent representational collapse. This is arguably easier to interpret than the predictor. The main disadvantage of DINO is that it requires a large amount of computational resources and data, making it challenging for researchers to train their own models. Newer iterations such as DINO v2 (Oquab et al., 2024) attempt to reduce the amount of pre-training by simply mapping embeddings of new images to existing embeddings. Other approaches such as PAWS (Assran et al., 2021) further exploit learning with soft labels for improved efficiency.

4.3 Efficient self-supervised vision transformers (EsViT)

Li et al. (2021) keep the same teacher-student architecture as Caron et al. (2021), but replace the ViTs with multi-stage transformers. As an optimization to the transformer architecture, a multi-stage transformer subsequently merges image patches together

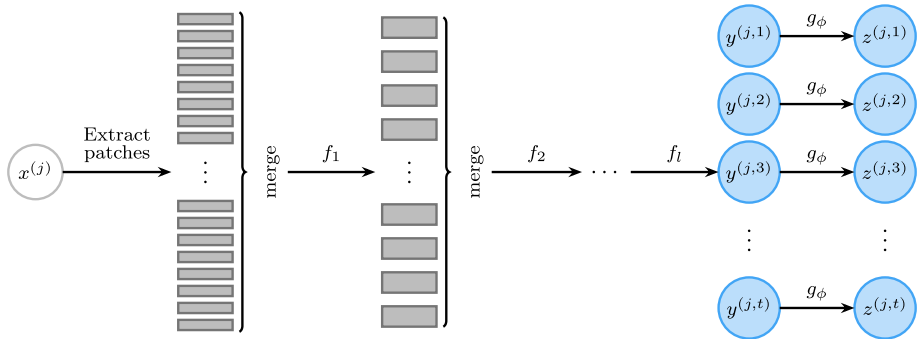


Fig. 12 Multi-stage transformer: an image is decomposed into image patches, which are merged and applied self-attention to subsequently. This results in a smaller number of patches being processed simultaneously, while also learning hierarchical embeddings. At the end, either average pooling is performed over the outputs or the output sequence is used as it is

across every layer to reduce the number of image patches that have to be processed. The authors show that the merging process destroys important local to global correspondences which are learned in common transformers. Therefore, an additional region matching loss is proposed that mitigates the lost semantic correspondences during the merging process in multi-stage transformer architectures.

4.3.1 Multi-stage vision transformers

Vaswani et al. (2021) reduce computational complexity of a standard transformer by reducing the number of patches that go through the transformer at each layer. For this, a special image patch merging module merges the patches at each layer and attention is calculated between them via sparse self-attention modules. This process is repeated multiple times. An illustration of the procedure is provided in Fig. 12. Overall, the number of tokens, i.e., feature maps that have to be evaluated by one self-attention module is reduced through each subsequent layer, while allowing for more diverse feature learning due to the different self-attention heads processing different merged patches, allowing for hierarchical correspondences to be learned.

4.3.2 Extending the loss function for region level feature matching

Li et al. (2021) propose to extend DINO's loss for multi-stage transformers in order to learn the local-to-global correspondences lost during the merging process. As EsViT is an extension of DINO, the teacher network and student network are defined as in Sect. 4.2. Li et al. (2021) propose a loss function that consists of a view-level loss ℓ_{view} and a region-level loss ℓ_{region} . The view-level loss is the same loss used to train DINO, i.e.,

$$\ell_{\text{view}} = \frac{1}{n} \sum_{i=1}^n \sum_{j \in M_i} \sum_{k \neq j} d_{\text{ce}}(\text{softmax}_{\tau}(\bar{z}_i^{(j)}) - c_{\text{view}}), \text{softmax}_{\rho}(z_i^{(k)})), \quad (38)$$

where c_{view} is the centering vector for the view-level loss.

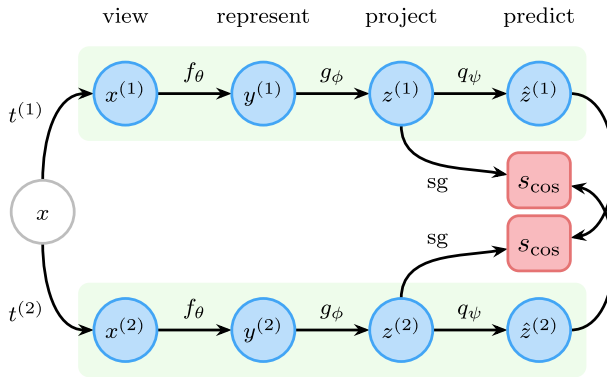


Fig. 13 SimSiam maximizes the cosine similarity between the projected representations. Both networks use the same parameters. The stop gradient operator interrupts the backward-flow of the gradients and thereby preventing the representational collapse

The region-level loss of EsViT is computed from the encoder outputs for each image patch $Y_i = [y_i^{(1,1)}, \dots, y_i^{(m,T)}]$ directly (see Fig. 12), where T is the sequence length, i.e., the number of patches for a given view j . Then the region-level loss is defined as

$$\ell_{\text{region}} = \frac{1}{n} \sum_{i=1}^n \sum_{j \in M_i} \sum_{k \neq j} \sum_{t=1}^T d_{\text{cc}}(\text{softmax}_{\tau}(\bar{z}_i^{(j,s^*)}) - c_{\text{region}}, \text{softmax}_{\rho}(z_i^{(k,t)})). \quad (39)$$

where $s^* = \underset{s}{\operatorname{argmax}} s_{\text{cos}}(\bar{z}_i^{(j,s)}, z_i^{(k,t)})$, T is the number of image patches, and c_{region} is the centering vector for the region-level loss. The idea is to match every image patch projection of the student $z_i^{(k,t)}$ to the best image patch projection of the teacher $\bar{z}_i^{(j,s^*)}$. That is, for every projection as defined by the multi-crop strategy in Fig. 11, the region-level loss matches the most concurring image patches of the student and teacher. The final EsViT loss combines the view-level and region-level loss, i.e.,

$$\mathcal{L}_{\theta, \phi}^{\text{EsViT}} = \ell_{\text{view}} + \ell_{\text{region}} \quad (40)$$

The authors show that when only training with the view-level loss on a multi-stage transformer, the model fails to capture meaningful correspondences, such as the background being matched for two augmentations of the same image. Adding the region-level loss mitigates the issue of lost region-level correspondences in multi-stage transformer architectures and recovers some of the correspondences lent inherently by monolithic transformer architectures. The main advantage of EsViT's hierarchical architecture as shown by Li et al. (2021) is that it allows for an equally well performing model with a smaller parameter count compared to other self-supervised methods such as DINO, where the smaller parameter count increases inference performance for downstream tasks. Although EsViT is more efficient than DINO due to its similarity to DINO, it still requires a large amount of computational resources and data to train.

4.4 Simple siamese representation learning (SimSiam)

SimSiam was introduced by Chen and He (2021) and uses a similar architecture and loss function as BYOL. However, teacher and student share the same parameters and hence a momentum encoder is not used as in previously presented teacher-student methods. Zhang et al. (2022) show that the predictor on one path and the stop-gradient on the other are sufficient to induce an asymmetric architecture, which only allows the path with the predictor to be optimized. Hence, the path with the predictor becomes the student, while the path with stop-gradient provides the encoder outputs for the predictor to learn, thus acting as teacher.

Given a batch of images X , for each image x_i two views $x_i^{(1)} = t^{(1)}(x_i)$, $x_i^{(2)} = t^{(2)}(x_i)$ are created using random transformations $t^{(1)}, t^{(2)} \sim \mathcal{T}$ that are sampled for each image of the batch. For each of these views, a Siamese encoder f_θ computes a representation $y_i^{(j)} = f(x_i^{(j)})$ and a Siamese projector g_ϕ computes a projection $z_i^{(j)} = g_\phi(y_i^{(j)})$. Finally, the projection is fed through a predictor q_ψ to obtain a prediction $\hat{z}_i^{(j)} = q_\psi(z_i^{(j)})$. See Fig. 13 for an overview of SimSiam.

The goal of the predictor is to predict the projection of the other view. Therefore, the loss computes the negative cosine similarity between the prediction of the first view and the projection of the second view, and vice versa, i.e.,

$$\mathcal{L}_{\theta, \phi, \psi}^{\text{SimSiam}} = -\frac{1}{n} \sum_{i=1}^n \frac{1}{2} \left[s_{\cos}(\hat{z}_i^{(1)}, \text{sg}(z_i^{(2)})) + s_{\cos}(\hat{z}_i^{(2)}, \text{sg}(z_i^{(1)})) \right], \quad (41)$$

where $\text{sg}(\cdot)$ is the stop gradient operator that prevents gradients from being backpropagated through this branch of the computational graph.

The encoder f_θ is implemented as a ResNet (He et al., 2016). The projector g_ϕ and the predictor q_ψ are MLPs. The authors show empirically that a predictor is crucial to avoid collapse. Chen and He (2021) argue that the gradient of the symmetrized loss with a predictor that is the identity is in the same direction as the gradient of the symmetrized loss between the two projections, such that the stop-gradient operation is cancelled out, thus leading to representation collapse. Using a random predictor does not work either and (Chen & He, 2021) argue that the predictor should always learn the latest representations. This argument is similar to the one made by Grill et al. (2020) in Sect. 4.2, where the predictor should be kept near optimal at all times.

The authors experiment with the training objective by replacing it with the cross-entropy loss. Their experiments show that this also works, however the performance is worse.

The key advantage of SimSiam is that training does not require very large batch sizes allowing the use of SGD instead of LARS. A disadvantage of SimSiam is that it heavily relies on batch normalization (Ioffe & Szegedy, 2015), which is used for both f_θ and g_ϕ . SimSiam is also very sensitive to the size of the model as Li et al. (2022) discuss in their work.

4.5 Advantages and disadvantages

One of the key advantages of teacher-student methods is the ability to learn from any batch of augmented image samples due to the teacher network providing stable targets for learning. This is an advantage to contrastive methods which require additional (negative) samples to learn representations. We will take a closer look at these methods in Sect. 5.

Grill et al. (2020) show that this allows BYOL for instance to be more robust against the choice of image augmentations and to work with smaller batch sizes. Another advantage of teacher-student methods is that they learn good representations with emergent features (e.g., object boundaries and correspondences between objects). Caron et al. (2021) show for DINO that the trained attention masks are able to detect complex features within a scene. These embeddings can then be used directly, e.g., as segmentation masks, that allow DINO to perform well on downstream tasks simply by using a k -nearest-neighbor classifier on its representations. Both of these key advantages have made methods such as BYOL and DINO very useful in downstream applications as shown in Sect. 8. This however also comes at the cost of an often more complex architecture using momentum encoding and a separate predictor layer, which is difficult to interpret and is also prone to mode collapse as shown by Chen and He (2021), Pham et al. (2022) and Ioffe and Szegedy (2015). An improvement to this is removing the predictor present in BYOL and SimSiam as done by Caron et al. (2021), while relying on the centering and sharpening of the teacher output to prevent representational collapse, which is arguably easier to interpret than the predictor. Overall, the effect of several components in teacher-student methods is currently not well understood and an active area of research. Another drawback of teacher-student methods similar to larger contrastive learning methods is that they require a large amount of computational resources and data, making it challenging for researchers to train their own models.

4.6 Relations to other methods

4.6.1 Information maximization methods

Recent work has shown that BYOL and SimSiam perform feature decorrelation like information maximization methods such as Barlow Twins and VICReg. To show this, Liu et al. (2022) first replace the prediction MLP q_ψ in BYOL and SimSiam by a linear mapping $q_W(z) = Wz$. They then derive an equation for the weight dynamics of a mini-batch

$$W(t)^T W(t) = Z(t)Z(t)^T, \quad (42)$$

where $W(t)$ is the weight matrix of q_W , $Z(t)$ is the mini-batch of projections for the predictor at time-step t . Note that $Z(t)Z(t)^T = \frac{1}{n} \sum_{i=1}^n z_i z_i^T$ is the uncentered covariance matrix of the mini-batch Z . Liu et al. (2022) show empirically that during training $W(t)^T W(t)$ becomes diagonal, hence the covariance matrix of the projections $Z(t)Z(t)^T$ becomes diagonal. This means that as training progresses with BYOL and SimSiam, the projections of the examples in the mini-batch become increasingly uncorrelated. Incidentally, this is the same as in the presented feature decorrelation methods. VICReg's main objective is identical, to approximate an identity covariance matrix between samples in the mini-batch, while Barlow Twins similarly approximates an identity cross-correlation matrix between the mini-batches of each image transformation branch.

4.6.2 Contrastive methods

Recent work has shown that teacher-student methods seem to perform implicit contrastive learning. We expand on this in Sect. 5.10.

5 Contrastive methods

Contrastive methods prevent representation collapse by *decreasing* the similarity between representations of unrelated data points. Given a data point x^* called the *anchor*, one defines mechanisms to generate *positive* samples and *negative* samples for the anchor. The positives should retain the relevant information of the anchor, while the negatives should contain information different from the anchor. For vision tasks, the positives could, e.g., be random transformations of the same image, while the negatives are (transformations of) other images. The goal of contrastive methods is to move representations of positives closer to the representation of the anchor while moving representations of negatives away from the anchor.

More formally, given an anchor x^* we define the conditional distributions of positives $p_{\text{pos}}(x^+|x^*)$ and negatives $p_{\text{neg}}(x^-|x^*)$. These distributions are induced by the mechanisms that generate the positives and negatives and are not explicitly known. Let $y^* = f_{\theta}(x^*)$, $y^+ = f_{\theta}(x^+)$, and $y^- = f_{\theta}(x^-)$ be the corresponding representations, calculated with an encoder f_{θ} parameterized by θ . The task of contrastive methods is to maximize the likelihood of positive representations $p_{\text{pos}}(y^+|y^*)$ while minimizing the likelihood of negative representations $p_{\text{neg}}(y^-|y^*)$. Note that we continue to use representations y in this introduction, but the same methods can be applied to projections z equivalently.

5.1 Noise contrastive estimation (NCE)

The idea of noise contrastive estimation (Gutmann & Hyvärinen, 2010) is to formulate the task of contrastive representation learning as a supervised classification problem. One assumption of NCE is that the negatives are independent from the anchor, i.e., $p_{\text{neg}}(x^-|x^*) = p_{\text{neg}}(x^-)$. In this context, the negatives are often called *noise*. There are two widely used approaches, the original NCE and InfoNCE (van den Oord et al., 2018). Roughly speaking, NCE performs binary classification to decide whether an individual sample is a positive or negative, whereas InfoNCE performs multiclass classification on a set of samples to decide which one is the positive. In the following we will explain InfoNCE in more detail.

5.2 InfoNCE

For each anchor x^* , InfoNCE generates one positive sample from $p_{\text{pos}}(x^+|x^*)$ and $n - 1$ negative samples from $p_{\text{neg}}(x^-)$. Let $X = [x_1, \dots, x_n]$ be the set of those samples, where x_c is the positive with index $c \in \{1, \dots, n\}$. In the context of representation learning we further compute representations using an encoder f_{θ} and obtain the set $Y = [y_1, \dots, y_n]$.

InfoNCE now defines a supervised classification task, where the input is (y^*, Y) and the class label is the index of the positive c . A classifier $p_{\psi}(c|Y, y^*)$ with parameters ψ is trained to match the true data distribution of the labels $p_{\text{data}}(c|Y, y^*)$. A common supervised learning objective is to minimize the cross-entropy between the data distribution and the model distribution, i.e.,

$$\min_{\psi, \theta} \mathbb{E}_{Y, y^*} [H(p_{\text{data}}(c|Y, y^*), p_{\psi}(c|Y, y^*))] \quad (43)$$

$$= \min_{\psi, \theta} \mathbb{E}_{Y, y^*} [\mathbb{E}_{c|Y, y^*} [-\log p_{\psi}(c|Y, y^*)]] . \quad (44)$$

Note that this is an anticausal prediction problem, where the underlying cause (label) is predicted from its effect (input) (Schölkopf et al., 2012). In InfoNCE we know the underlying mechanism (since we generate the labels artificially), so we can derive the optimal classifier using Bayes' theorem.

First, we write down the data distribution of a set Y given a label and an anchor, i.e.,

$$p_{\text{data}}(Y|c, y^*) = \prod_{i=1}^n p_{\text{data}}(y_i|c, y^*) = \prod_{i=1}^n \begin{cases} p_{\text{pos}}(y_i|y^*), & \text{if } i = c, \\ p_{\text{neg}}(y_i), & \text{if } i \neq c, \end{cases} \quad (45)$$

$$= p_{\text{pos}}(y_c|y^*) \prod_{i \neq c} p_{\text{neg}}(y_i) = \frac{p_{\text{pos}}(y_c|y^*)}{p_{\text{neg}}(y_c)} \prod_{i=1}^n p_{\text{neg}}(y_i), \quad (46)$$

where we assume conditional independence among the samples in Y . InfoNCE further assumes that the labels are sampled uniformly, i.e., $p_{\text{data}}(c) = \frac{1}{n}$. Now we apply Bayes' theorem:

$$p_{\text{data}}(c|Y, y^*) = \frac{p_{\text{data}}(Y|c, y^*) p_{\text{data}}(c)}{\sum_{c'=1}^n p_{\text{data}}(Y|c', y^*) p_{\text{data}}(c')} \quad (47)$$

$$= \frac{\frac{p_{\text{pos}}(y_c|y^*)}{p_{\text{neg}}(y_c)} \prod_{i=1}^n p_{\text{neg}}(y_i) \frac{1}{n}}{\sum_{c'=1}^n \frac{p_{\text{pos}}(y_{c'}|y^*)}{p_{\text{neg}}(y_{c'})} \prod_{i=1}^n p_{\text{neg}}(y_i) \frac{1}{n}} \quad (48)$$

$$= \frac{\frac{p_{\text{pos}}(y_c|y^*)}{p_{\text{neg}}(y_c)}}{\sum_{c'=1}^n \frac{p_{\text{pos}}(y_{c'}|y^*)}{p_{\text{neg}}(y_{c'})}} . \quad (49)$$

An optimal classifier with zero cross-entropy would match this distribution. We can see that the optimal probability of a class is the density ratio $\frac{p_{\text{pos}}(y_c|y^*)}{p_{\text{neg}}(y_c)}$, normalized across all classes. It describes the likelihood of y_c being a positive sample for y^* versus being a negative sample. This motivates the choice of the classifier of InfoNCE, which is defined similar to Eq. 49 as

$$p_{\psi}(c|Y, y^*) = \frac{s_{\psi}(y^*, y_c)}{\sum_{c'=1}^n s_{\psi}(y^*, y_{c'})}, \quad (50)$$

where $s_{\psi}(y^*, y)$ is a predictor that computes a real-valued positive score. Minimizing the cross-entropy from Eq. 43 brings the model distribution $p_{\psi}(c|Y, y^*)$ closer to the data distribution $p_{\text{data}}(c|Y, y^*)$, which ensures that s_{ψ} approaches the density ratio of the data, i.e., $s_{\psi}(y^*, y) \approx \frac{p_{\text{pos}}(y|y^*)}{p_{\text{neg}}(y)}$ (in fact, it only needs to be proportional to the density ratio). The density ratio is high for positive samples and close to zero for negative samples, which means that $s_{\psi}(y^*, y)$ learns some similarity measure between the representations. Since ψ and θ (i.e., predictor and encoder) are optimized jointly, the encoder is encouraged to learn

similar embeddings for an anchor and its positive, and to learn dissimilar embeddings for an anchor and its negative samples (as long as the predictor is not too expressive). In other words, the encoder is encouraged to extract information that is “unique” to the anchor and the positive sample.

By combining the negative logarithm and the classifier (Eqs. 44 and 50) the general InfoNCE loss for (y^*, Y, c) is defined as

$$\text{InfoNCE}_{s_\psi}(y^*, Y, c) = -\log \left(\frac{s_\psi(y^*, y_c)}{\sum_{c'=1}^n s_\psi(y^*, y_{c'})} \right). \quad (51)$$

For the sake of notation in the following sections, we slightly adjust this definition. All considered methods compute the exponential of some score to obtain positive values for s_ψ , which is why we include it directly in the loss function. In this case the InfoNCE loss computes the commonly used softmax cross-entropy. Instead of specifying the class label we denote the positive by y^+ and the set of negatives by $\bar{Y}$. Thus, our final definition of the InfoNCE loss for a score function $s_\psi(y^*, y)$ is

$$\text{InfoNCE}_{s_\psi}(y^*, y^+, \bar{Y}) = -\log \left(\frac{\exp(s_\psi(y^*, y^+))}{\exp(s_\psi(y^*, y^+)) + \sum_{\bar{y} \in \bar{Y}} \exp(s_\psi(y^*, \bar{y}))} \right). \quad (52)$$

5.3 Contrastive predictive coding (CPC)

CPC (van den Oord et al., 2018) is an influential self-supervised representation learning technique which is applicable to a wide variety of input modalities such as text, speech and images. It is based on the theory of predictive coding, which originated in the neuroscience literature by observing the learning behavior of biological neural circuits (Huang & Rao, 2011; Bastos et al., 2012). In short, a model tries to predict future outcomes given the past or *context*. Thus, the learned representation of the context should incorporate all information necessary for prediction while removing unimportant noise. This predictive coding task is solved by formulating it as a contrastive learning problem. CPC operates on sequential data, which is a natural choice for audio data, but can also be applied to vision tasks by splitting images into sequences of patches.

Given a batch of images X , we consider a single image x_i . The image is split into m patches $[x_i^{(1)}, \dots, x_i^{(m)}]$. Note that the patches are overlapping and that an additional image augmentation is applied to each patch. A Siamese encoder f_θ converts each patch into a representation $y_i^{(j)} = f_\theta(x_i^{(j)})$. The *context* of a patch are the patches in the same row and all patches in the rows above. Let $C_i^{(j)} \subset \{1, \dots, m\}$ be the set of indices of those patches. A projector g_ϕ computes a context representation $z_i^{(j)} = g_\phi([y_i^{(k)} : k \in C_i^{(j)}])$ for each patch. The network g_ϕ is implemented by a masked CNN (PixelCNN, (van den Oord et al., 2016)) that restricts its receptive field to the patches in the context.

Given a context representation $z_i^{(j)}$, the predictive coding task of CPC is to predict representations of future patches, which are the patches in the column below the patch. Let $F_i^{(j)} = [k_{i,1}^{(j)}, \dots, k_{i,K}^{(j)}] \subset \{1, \dots, m\}$ be the set of indices of those patches. The prediction task is solved by minimizing an InfoNCE loss for each future representation. For the l -th future representation with patch index $k = k_{i,l}^{(j)}$, a separate predictor q_{ψ_l} computes the anchor $\hat{y}_i^{(k)} = q_{\psi_l}(z_i^{(j)})$ from the context representation, and the positive is the future representation

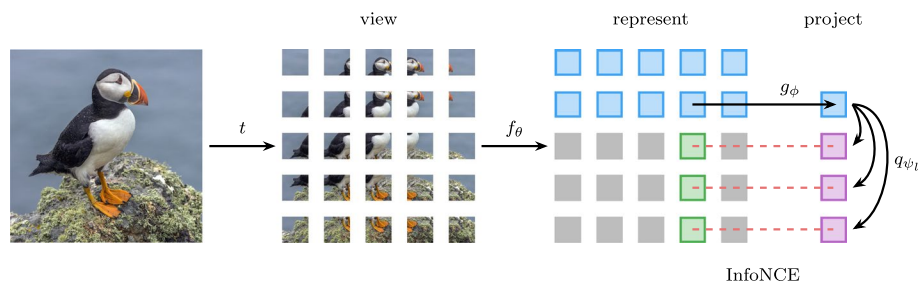


Fig. 14 In CPC training an image is transformed and split into overlapping patch encodings, where the top rows (blue) act as the anchor. The positive samples (purple) lie in a column below the exact anchor position and every other patch of the whole dataset is a negative. InfoNCE is applied to distinguish between both distributions

$y_i^{(k)}$. The negatives can be any unrelated representations, e.g., the representations of all patches outside the context and the future, as well as all representations from other images in X . We denote the set of negatives by $\tilde{Y}_i^{(j)}$. The loss function accumulates the InfoNCE losses over all contexts and futures across the batch, i.e.,

$$\mathcal{L}_{\theta, \phi, \psi}^{\text{CPC}} = \frac{1}{n} \sum_{i=1}^n \frac{1}{m} \sum_{j=1}^m \sum_{k \in F_i^{(j)}} \text{InfoNCE}_s(\hat{y}_i^{(k)}, y_i^{(k)}, \tilde{Y}_i^{(j)}), \quad (53)$$

where the similarities are calculated using the dot product $s(\hat{y}, y) = \hat{y}^\top y$, and the parameters of all predictors are combined in $\psi = [\psi_1, \dots, \psi_K]$. See Fig. 14 for an illustration of the method.

5.3.1 CPC v2

The second version of CPC (Henaff, 2020) targets the problem of sample efficiency. So far, contrastive methods require a large amount of data, especially the need to find hard negatives, to perform well on common benchmarks such as ImageNet (Bardes et al., 2021). This work focuses on the improvement of training routines rather than modifying the general idea behind CPC. The most notable changes are improved image augmentations, larger network sizes, using layer normalization (Ba et al., 2016) instead of batch normalization (Ioffe & Szegedy, 2015), which creates an unintentional co-dependency between the patches, and extending the prediction task to all four directions, instead of predicting bottom from top patches only.

5.4 Contrastive multiview coding (CMC)

CMC (Tian et al., 2020a) considers multiple views of the same scene and tries to maximize the mutual information between those views. For each view, a view-specific encoder extracts view-invariant features. A contrastive learning objective forces the encoders to be as informative as possible about the other views. In general, the views could be different sensory inputs of the same scene (e.g., color, depth, surface normals). For vision tasks that only have access to RGB images, the different views could be individual color channels. In

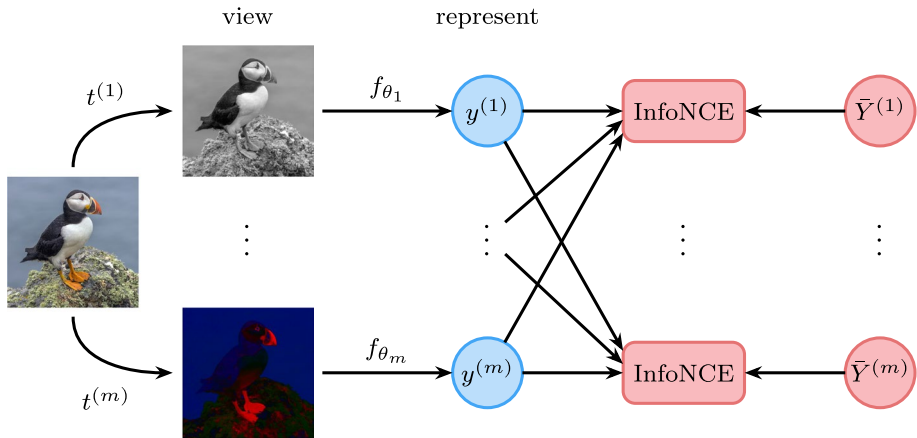


Fig. 15 CMC assumes there are different interfaces to the world, that collect information about a shared source. The goal is to find a consensus about the perceived information coming from the different modalities. This is done by extracting the view-invariant features in order to learn a shared representation

this paper, the authors consider the L and ab channels of an image after converting it to the Lab color space. Note that the views can be interpreted as image augmentations, however, each view uses the *same* image augmentation for all images, which is in contrast to other methods.

To apply CMC to images, we define m fixed image transformations $[t^{(1)}, \dots, t^{(m)}]$. Let X be a batch of images that gets transformed into $X^{(j)} = t^{(j)}(X)$ for each view $j \in \{1, \dots, m\}$. The encoders $f_{\theta_1}, \dots, f_{\theta_m}$ compute view-specific representations $y_i^{(j)} = f_{\theta_j}(x_i^{(j)})$ for each $j \in \{1, \dots, m\}$. The representation used for downstream tasks can either be the representation of a specific view or the concatenation of representations across multiple or all views.

The idea of CMC is to apply the InfoNCE loss to pairs of views. Specifically, the anchor is the representation $y_i^{(j)}$ of the j -th view, the positive is the representation $y_i^{(k)}$ of the same image but from the k -th view, where $k \neq j$, and the negatives are representations from other images but also from the k -th view. We denote the set of those negative images by $\bar{Y}_i^{(k)}$ which are obtained using a memory bank (Wu et al., 2018). With memory banks, large batches of negative samples can be obtained efficiently, at the cost of slightly outdated representations.

The loss for a single image x_i accumulates the InfoNCE losses of all ordered pairs of views, so the total loss function across the batch is given as

$$\mathcal{L}_{\theta}^{\text{CMC}} = \frac{1}{n} \sum_{i=1}^n \sum_{j=1}^m \sum_{k=1; k \neq j}^m \text{InfoNCE}_{s_{\tau}}(y_i^{(j)}, y_i^{(k)}, \bar{Y}_i^{(k)}), \quad (54)$$

where the similarities are calculated as $s_{\tau}(y, y') = s_{\cos}(y, y')/\tau$, i.e., as cosine similarity divided by a temperature hyperparameter $\tau > 0$, and where $\theta = [\theta_1, \dots, \theta_m]$ combines the parameters of all encoders. See Fig. 15 for an illustration of the method.

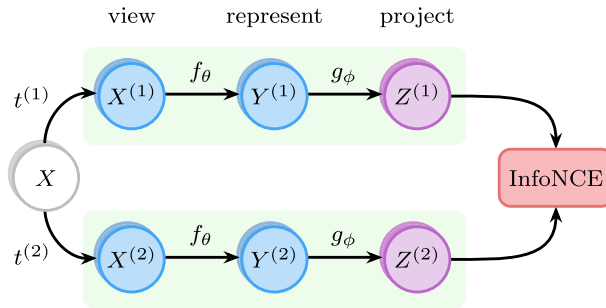


Fig. 16 SimCLR defines the views in the batch that were constructed from different images as negative examples

5.5 Simple contrastive learning of representations (SimCLR)

The architecture used for SimCLR (Chen et al., 2020a) is similar to previous methods like VICReg or Barlow Twins. Given a batch of images X , two views $X^{(1)} = t^{(1)}(X)$ and $X^{(2)} = t^{(2)}(X)$ are created using random transformations $t^{(1)}, t^{(2)} \sim \mathcal{T}$. A Siamese encoder f_θ calculates representations $Y^{(1)} = f_\theta(X^{(1)})$ and $Y^{(2)} = f_\theta(X^{(2)})$ which are then fed into a Siamese projector g_ϕ to obtain projections $Z^{(1)} = [z_1^{(1)}, \dots, z_n^{(1)}] = g_\phi(Y^{(1)})$ and $Z^{(2)} = [z_1^{(2)}, \dots, z_n^{(2)}] = g_\phi(Y^{(2)})$. Figure 16 gives an overview over this process.

SimCLR uses a contrastive loss to maximize the similarity between the two projections of the same image while minimizing the similarity to projections of other images. Specifically, for an image x_i two InfoNCE losses are applied. The first one uses the anchor $z_i^{(1)}$, the positive $z_i^{(2)}$, and the negatives $\bar{Z}_i = [z_1^{(1)}, z_1^{(2)}, \dots, z_n^{(1)}, z_n^{(2)}] \setminus \{z_i^{(1)}, z_i^{(2)}\}$, which are all projections from other images in the batch. The second InfoNCE loss swaps the roles of anchor and positive but uses the same set of negatives. Therefore, the loss function is defined as

$$\mathcal{L}_{\theta, \phi}^{\text{SimCLR}} = \frac{1}{n} \sum_{i=1}^n \frac{1}{2} \left[\text{InfoNCE}_{s_\tau}(z_i^{(1)}, z_i^{(2)}, \bar{Z}_i) + \text{InfoNCE}_{s_\tau}(z_i^{(2)}, z_i^{(1)}, \bar{Z}_i) \right], \quad (55)$$

where the similarities are calculated as $s_\tau(z, z') = s_{\cos}(z, z')/\tau$, i.e., the cosine similarity divided by a temperature hyperparameter $\tau > 0$.

The transformations consist of a random cropping followed by a resize back to the original size, a random color distortion, and a random Gaussian blur. A ResNet is used as encoder f_θ and the projector g_ϕ is implemented as an MLP with one hidden layer. To train SimCLR large batch sizes are used in combination with the LARS optimizer (You et al., 2017). The authors note that their method does not need memory banks (Wu et al., 2018) as it is the case for other contrastive methods and is thus easier to implement.

5.6 Momentum contrast (MoCo)

Momentum Contrast (He et al., 2020) is a contrastive learning approach that uses a momentum encoder with an encoding queue to bridge the gap between contrastive *end-to-end* and *memory bank* methods (Wu et al., 2018). In essence, it allows the optimization of a

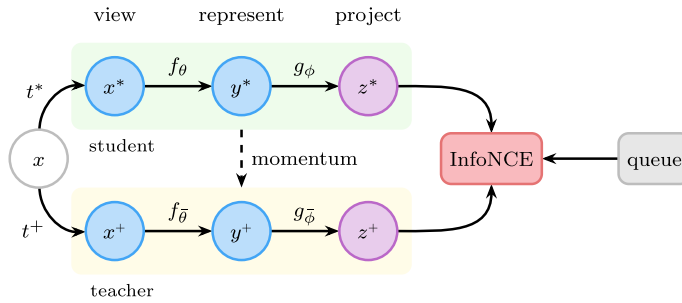


Fig. 17 MoCo computes projections of different views with a student and a teacher network, and minimizes the contrastive InfoNCE loss. The projections z^* and z^+ are computed on-the-fly, whereas negative projections are cached in a queue of the most recent versions of z^+ from previous iterations, significantly improving computational efficiency

contrastive objective with significantly reduced computational costs, both in terms of time and GPU memory (Chen et al., 2020b). Figure 17 gives an overview of the architecture.

Similar to teacher-student methods (Sect. 4.1.1), MoCo defines a student network consisting of an encoder f_θ and a projector g_ϕ with parameters θ and ϕ , and a teacher network consisting of an encoder $f_{\bar{\theta}}$ and a projector $g_{\bar{\phi}}$ with parameters $\bar{\theta}$ and $\bar{\phi}$. Given an image x_i , two views $x_i^* = t^*(x_i)$ and $x_i^+ = t^+(x_i)$ are created using random transformations $t^*, t^+ \sim \mathcal{T}$. The student computes the representation $y_i^* = f_\theta(x_i^*)$ and projection $z_i^* = g_\phi(y_i^*)$, while the teacher computes the representation $y_i^+ = f_{\bar{\theta}}(x_i^+)$ and projection $z_i^+ = g_{\bar{\phi}}(y_i^+)$.

MoCo minimizes the InfoNCE loss to learn projections that are similar for two views of the same image and dissimilar to projections of views of other images. In our notation, the student computes the anchor z_i^* , the teacher computes the positive z_i^+ , and the selection of the negatives $\bar{Z}_i$ is described below. The loss of MoCo is then defined as

$$\mathcal{L}_{\theta, \phi}^{\text{MoCo}} = \frac{1}{n} \sum_{i=1}^n \text{InfoNCE}_{s_i}(z_i^*, z_i^+, \bar{Z}_i), \quad (56)$$

where the similarities are calculated using the dot product $s_i(z^*, z) = z^\top z^* / \tau$ divided by a temperature hyperparameter $\tau > 0$. The teacher is updated by an exponential moving average of the student, i.e.,

$$\bar{\theta} \leftarrow \alpha \bar{\theta} + (1 - \alpha) \theta, \quad (57)$$

$$\bar{\phi} \leftarrow \alpha \bar{\phi} + (1 - \alpha) \phi \quad (58)$$

where $\alpha \in [0, 1]$ controls the rate at which the weights of the teacher network are updated with the weights of the student network.

In an *end-to-end* setting, the negatives are computed on-the-fly in one batch (see SimCLR, Sect. 5.3.2.1), resulting in relatively large resource consumption. In contrast, memory banks (Wu et al., 2018) describe the concept of saving projections for all items in the dataset, drastically reducing resource consumption but introducing potential negative effects from inconsistent or outdated projections. MoCo aims to combine the benefits of both end-to-end training and memory banks. Similar to memory banks, MoCo only computes projections of the positives and saves them for reuse in later iterations. Instead of saving projections for all images in the dataset, MoCo uses a queue to cache only the last K computed

projections, thus avoiding outdated projections. Since older projections are removed from the queue, saved projections no longer require momentum updates. The teacher provides the projections that are to be cached, while the student is updated via backpropagation using the contrastive loss.

5.6.1 MoCo v2

The second version of MoCo (Chen et al., 2020b) introduces several smaller changes to further improve downstream performance and outperform SimCLR. The most notable changes include the replacement of the linear projection layer of MoCo with an MLP, as well as the application of a cosine learning rate scheduler (Loshchilov & Hutter, 2017) and additional augmentations. The new 2-layer MLP head was adopted following SimCLR. Note, that the MLP is only used during unsupervised training and is not intended for downstream tasks. In terms of additional augmentations, MoCo v2 also adopts the blur operation used in SimCLR.

5.7 Pretext-invariant representation learning (PIRL)

In the previously introduced pretext tasks we compute the representations of transformed images to predict properties from specific transformations, i.e., rotation angles (Noroozi and Favaro, 2016) or patch permutations (Gidaris et al., 2018). In this way, representations are encouraged to be covariant to the specific transformation, but are not guaranteed to capture the same underlying semantic information regardless of the transformation used. Although such covariance is advantageous in certain cases, we are more interested in representations that are semantically meaningful, so it is desirable to learn representations that are invariant to the transformation. In order to achieve this, Misra and Maaten (2020) refined the pretext task loss formulation and developed an approach called Pretext-Invariant Representation Learning (PIRL) which also makes use of memory banks (Wu et al., 2018).

The goal of PIRL is to train an encoder network f_θ that maps images $x_i^{(1)} = x_i$ and transformed images $x_i^{(2)} = t_\pi(x_i)$ to representations $y_i^{(1)}$ and $y_i^{(2)}$, respectively, which are invariant to the transformations used. Analogous to Sect. 2.3, t_π denotes a jigsaw transformation consisting of a random permutation of image patches, where π is the corresponding permutation. The loss formulation of pretext tasks as defined in Sect. 2 emphasizes that the encoder learns representations that contain information about the transformation rather than semantics. Let $z_i^{(1)} = g_\phi(f_\theta(x_i^{(1)}))$ and $z_i^{(2)} = g_\psi(f_\theta(x_i^{(2)}))$ be the projections obtained by the encoder f_θ and two separate projectors g_ϕ and g_ψ . The network is trained by minimizing a convex combination of two noise contrastive estimators (NCE) (Gutmann & Hyvärinen, 2010)

$$\mathcal{L}_{\theta, \phi, \psi}^{\text{PIRL}} = \frac{1}{n} \sum_{i=1}^n \lambda \mathcal{L}_{\text{NCE}}(m_i, z_i^{(2)}, \bar{M}_i) + (1 - \lambda) \mathcal{L}_{\text{NCE}}(m_i, z_i^{(1)}, \bar{M}_i), \quad (59)$$

where m_i is a projection from a memory bank corresponding to the original image x_i , each positive sample is assigned a randomly drawn set of negative projections $\bar{M}_i$ of images other than x_i obtained from the memory bank, and $\lambda \in [0, 1]$ is a hyperparameter. In contrast to the previously introduced pretext tasks, the loss formulation of PIRL does not explicitly aim to predict particular properties of the applied transformations, such as rotation or patch indices. Instead, it is solely defined on images and their corresponding

transformed counterparts. NCE applies binary classification to each data point to distinguish positive and negative samples. Here, the NCE loss is formulated as

$$\mathcal{L}_{\text{NCE}}(m, z, \bar{M}) = -\log[h(m, z, \bar{M})] - \sum_{\bar{m} \in \bar{M}} \log[1 - h(z, \bar{m}, \bar{M})], \quad (60)$$

where h models the probability that (x_i, x'_i) is derived from X as

$$h(u, v, \bar{M}) = \frac{\exp(s_{\cos}(u, v)/\tau)}{\exp(s_{\cos}(u, v)/\tau) + \sum_{\bar{m} \in \bar{M}} \exp(s_{\cos}(\bar{m}, v)/\tau)} \quad (61)$$

for a temperature $\tau > 0$. Considering that the projections depend on the intermediate representations, the individual terms in Eq. 60 encourage $y_i^{(1)}$ to be similar to $y_i^{(2)}$ and also $y_i^{(2)}$ to be dissimilar to the representations of other images. Since this formulation alone does not compare features between different untransformed images, the authors propose to use the convex combination of two NCE losses as defined in Eq. 59. An overview of this approach is illustrated in Fig. 18. The encoder network f_θ consists of the final layer of ResNet50 (He et al., 2016), average pooling and a 128-dimensional fully connected layer. As for the image transformation in the lower branch of Fig. 18, we first extract nine image sections and apply them individually to f_θ to obtain patch representations $y_i^{(2,k)}$. These are then randomly concatenated and sent through another fully connected layer to obtain the 128-dimensional representation $y_i^{(2)}$. Although the authors focus their work on the Jigsaw pretext task, their approach can be generalized to any other pretext task. For demonstration purposes, the authors also conduct experiments with the rotation pretext task and its combination with the Jigsaw task. In this way we have to adapt the lower branch of Fig. 18 by transforming the image at the beginning and feeding forward the transformed image to achieve the representation $y_i^{(2)}$ directly. Thus, using a secondary fully connected layer is not necessary anymore.

Note that PIRL can also be classified as a pretext task approach as defined in Sect. 2. However, it also uses ideas of contrastive representation learning which is why we decided to discuss it at this point.

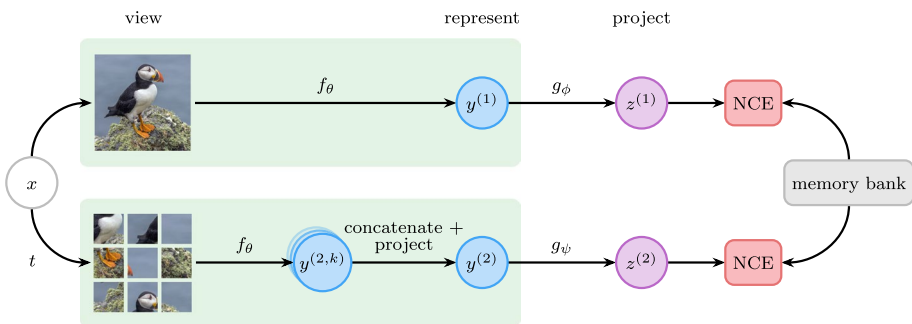


Fig. 18 Architecture of PIRL. Minimizing a contrastive loss promotes similarity between the representations of the image and its corresponding transformation

5.8 Advantages and disadvantages

Contrastive learning, despite its effectiveness on benchmarks, always faces several engineering and algorithmic challenges compared to other self-supervised learning paradigms. The primary issue lies in the identification and necessity of a large number of negative samples to distinguish positive pairs (Grill et al., 2020). Hereby, it is also important to note that identifying good negative samples is often inevitable to create a sufficient learning signal. Finding these negative samples can be particularly challenging, as poor choices may lead to sub-optimal model performance or even model collapse (Chen et al., 2020a; Grill et al., 2020). Moreover, negative pairs can only be determined by heuristics, which further complicates the optimization process. Finally, comparing against numerous negative samples within each batch requires fast memory bandwidth as well as significant memory and computational resources (Bardes et al., 2021), posing scalability issues for large-scale applications.

5.9 Comparison of contrastive methods

In general, contrastive methods tend to blend into each other very quickly, where every one of them has its own twist. However, all methods in this paper share a similar objective, namely NCE (Gutmann & Hyvärinen, 2010) or InfoNCE (van den Oord et al., 2018), which essentially characterizes them as contrastive methods in a broader context. To highlight their differences, we summarize the design choices in Table 1.

5.10 Relations to other methods

5.10.1 Pretext task methods

Although contrastive learning is in itself sufficient to achieve state-of-the-art performance on downstream tasks, it is notable that (Huang et al., 2022) are able to demonstrate a synergy between pretext tasks and contrastive learning. They use a method from the domain of masked image modeling (MIM), known as Masked Autoencoders (MAE), and combine it with ideas from this section. They emphasize the challenges that arise due to various distinctions, but can overcome them with adjustments to input augmentations, training objectives, model architectures and more. Their efforts achieve slightly better performance.

Table 1 Overview of the similarities and differences between the contrastive methods considered in this paper

Method	CPC	CMC	SimCLR	MoCo	PIRL
Objective	InfoNCE	InfoNCE	InfoNCE	InfoNCE	NCE
Negative sampling	Batch	Memory bank	Batch	Queue	Memory bank
Transformations	Random + patches	Deterministic	Random	Random	Random (Jigsaw)
Similarity	Dot product	Cosine sim	Cosine sim	Dot product	Cosine sim
# Positive pairs	mK	$m(m-1)$	2	1	2
# Encoders	1	m	1	2	1

The image transformations are either random, i.e., sampled from a predefined set $t \sim \mathcal{T}$, or deterministic. The number of positive pairs are counted per image

5.10.2 Information maximization methods

As stated in the introduction of this section, the encoder is encouraged to extract information that is unique to the anchor and the positive sample. More formally, van den Oord et al. (2018), Poole et al. (2019), and Song and Ermon (2020) show that minimizing the InfoNCE maximizes a lower bound on the mutual information between y^* and x^+ , i.e.,

$$I(Y^*; X^+) \geq \mathbb{E} \left[-\text{InfoNCE}_{s_\psi}(Y^*, Y^+, \bar{Y}) \right]. \quad (62)$$

Note, that the InfoNCE itself is upper bounded by $\log(n)$ where $n - 1$ is the number of negative samples, so the bound from Eq. 62 will be loose, if $I(Y^*; X^+) > \log(n)$ (Poole et al., 2019). Increasing the number of negative samples tightens this bound, and typically leads to better performance in practice. In this light, methods such as SimCLR and MoCo maximize the mutual information between two views, bearing a close resemblance to information maximization methods, e.g., VICReg, which maximizes the information content of the representations given the input samples.

Garrido et al. (2022) claim that non-contrastive methods, such as VICReg, actually are contrastive as well but on a feature level rather than contrasting entire samples. They propose two classes, one being *sample-contrastive* (see CPC, CMC or SimCLR), which minimize a contrastive objective on a batch of projections $Z \in \mathbb{R}^{d \times n}$ by considering the non-diagonal entries of its Gram matrix $Z^T Z$, i.e.,

$$\mathcal{L}_c = \|Z^T Z - \text{diag}(Z^T Z)\|_F^2, \quad (63)$$

and the other being *dimension-contrastive* (see VICReg, Barlow Twins, or WMSE), which includes methods that minimize the non-diagonal entries of the covariance matrix of the projections

$$\mathcal{L}_{nc} = \|ZZ^T - \text{diag}(ZZ^T)\|_F^2. \quad (64)$$

This means sample-contrastive methods try to make positive and negative representations orthogonal, while dimension-contrastive methods make the feature dimensions of the representations orthogonal. However, the authors show if Z is doubly-normalized (row and column normalized) that the following equality holds:

$$\mathcal{L}_{nc} + \sum_{j=1}^d \|Z_{j,:}\|_2^4 = \mathcal{L}_c + \sum_{i=1}^n \|Z_{:,i}\|_2^4, \quad (65)$$

where $Z_{:,i} = z_i$ is the representation of an input sample and $Z_{j,:}$ is a vector of the same feature across a batch. Their theorem proves that sample-contrastive and dimension-contrastive methods minimize similar objectives, but differ in practice especially when additional loss functions, e.g., the invariance loss from VICReg (Bardes et al., 2021), are introduced. In that case, $\mathcal{L}_c$ and $\mathcal{L}_{nc}$ cannot be used interchangeably.

5.10.3 Teacher-student methods

Several recent works argue that teacher-student methods perform an implicit type of contrastive learning. Tao et al. (2022) claim that BYOL can be connected to contrastive methods, although it does not use negatives explicitly. Roughly, the weight matrix of BYOL's

predictor network slowly approaches a feature correlation matrix of the representations, leading to gradients with similar effects as negative samples. Fetterman and Albrecht (2020) and Tian et al. (2020b) argue that methods such as BYOL implicitly perform contrastive learning via batch normalization. More specifically, the batch normalization layer redistributes the inputs according to mean and standard deviation. Thus, if before batch normalization different examples are distributed very similarly (indicating representation collapse), after batch normalization the outputs will be redistributed, hence preventing representation collapse. Taking into account Fig. 9, batch normalization identifies the common mode between projections in $Z^{(1)}$, which is then compared against the projections in $Z^{(2)}$ which implicitly serve as negative samples. Thus, according to Fetterman and Albrecht (2020), BYOL learns by considering how a sample is different from the average of other samples. Note, that here the samples are the augmented images for teacher and student respectively. Due to architectural similarities, this also holds for SimSiam, where (Chen & He, 2021) show that batch normalization is crucial to the method's performance. Taking a look at DINO's architecture seems to confirm this implicit contrastive learning. While DINO does not use batch normalization, it has a centering term for the teacher as explained in Sect. 4.2. The goal is to center the output probability distribution of the teacher to prevent representation collapse. Here too, the centering value is calculated across the batch, which captures the common mode between samples in the mini-batch.

6 Clustering-based methods

So far, some of the presented representation learning methods define a classification problem with hand-crafted labels to solve an auxiliary task (see Sect. 2). Instead of specifying these class labels by hand, clustering algorithms, e.g., k-means (Lloyd, 1982), can be used to create the labels in an unsupervised fashion.

The objective of clustering-based representation learning is to group images with similar representations into clusters. In contrastive learning, for example, this would allow us to discriminate between cluster assignments rather than individual images or representations, which significantly increases efficiency. Over the time, numerous clustering-based methods have been developed, each with its own strengths and weaknesses. In the subsequent sections, we present the most significant approaches.

6.1 DeepCluster

The first approach that implements the idea of clustering for representation learning is DeepCluster, which was introduced by Caron et al. (2018). DeepCluster alternates between inventing pseudo-labels via cluster assignments and adjusting the representation to classify images according to their invented labels. The motivation behind this is to increase the performance of convolutional architectures that already exhibit a strong inductive bias, as these already perform reasonably well with randomly initialized weights (Noroozi & Favaro, 2016). Overall, the authors propose to repeatedly alternate between the following two steps to further improve the encoder network:

1. Group the representations $y_i = f_\theta(x_i)$ produced by the current state of the encoder f_θ to k clusters (e.g., by using k -means clustering).

2. Use the cluster assignments from step 1 as pseudo-labels β_i for supervision to update the weights, i.e.,

$$\mathcal{L}_{\theta, \psi}^{\text{DeepCluster}} = \frac{1}{n} \sum_{i=1}^n d_{\text{classification}}(q_{\psi}(y_i), \beta_i), \quad (66)$$

where a predictor network q_{ψ} tries to predict the cluster assignments of the representations $y_i = f_{\theta}(x_i)$.

In their experiments, the authors utilize a standard AlexNet (Krizhevsky et al., 2017) with k -means and argue that the choice of the clustering algorithm is not crucial.

6.2 Self labelling (SeLa)

A common weakness of the naive combination of clustering and representation learning is its proneness to degenerate solutions where, for example, all representations are assigned to the same cluster. To circumvent this issue, Asano et al. (2020) have developed an refined alternating update scheme, called Self Labelling (SeLa), which imposes constraints on the labels such that each cluster is assigned the same amount of data points. The pseudo-labels corresponding to images $x_1, \dots, x_n$ are encoded as one-hot vectors $\beta_1, \dots, \beta_n \in \{0, 1\}^k$. To assign the pseudo-labels $\beta_1, \dots, \beta_n$ and fit encoder and predictor networks f_{θ} and q_{ψ} , respectively, the authors consider the optimization problem

$$\min_{\beta, \theta, \psi} \frac{1}{n} \sum_{i=1}^n d_{\text{ce}}(q_{\psi}(f_{\theta}(x_i)), \beta_i), \quad (67)$$

$$\text{s.t.} \quad \sum_{i=1}^n \beta_i[j] = \frac{n}{k}, \quad \beta_i[j] \in \{0, 1\} \quad \text{for } j \in \{1, \dots, k\}, \quad (68)$$

which they solve by alternating between the following two steps:

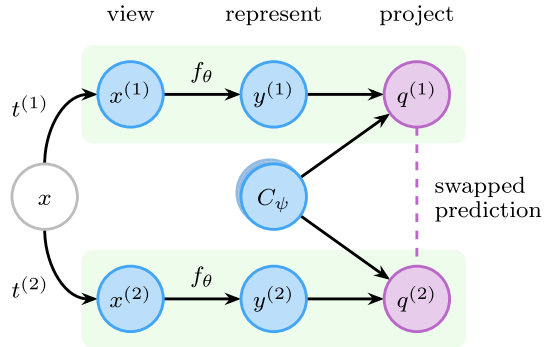
1. The problem of assigning the pseudo-label to the images is formulated as an optimal transport problem which is solved using a fast variant of the Sinkhorn-Knopp algorithm (Cuturi, 2013).
2. Fix the pseudo-labels from step 1 and update the parameters θ and ψ by minimizing the cross-entropy loss.

Note that step 2 is the same as used in DeepCluster. However, due to the choice of k -means in DeepCluster it is quite possible to obtain underrepresented cluster assignments that would lead to trivial solutions, i.e., constant representations are learned and consequently the minimum is achieved in both optimization steps.

6.3 Swapping assignments between multiple views (SwAV)

In general, contrastive methods are computationally challenging due to the need for numerous explicit pairwise feature comparisons. However, Caron et al. (2020) propose an alternative algorithm, called SwAV, that circumvents this problem by data clustering while promoting consistency among cluster assignments across different views. In contrast to

Fig. 19 SwAV does not actually measure the similarity between image representations of different views, but instead compares the representations with codes obtained by assigning the features to parameterized prototypes



DeepCluster and SeLa, SwAV is an online clustering-based approach, i.e., it does not alternate between a cluster assignment and training step. An encoder network f_θ is used to compute image representations $y^{(1)}$ and $y^{(2)}$ of two views of the same image x . These representations are then mapped to a set of k parameterized prototypes $C_\psi = [c_1, \dots, c_k]$, resulting in corresponding codes $q^{(1)}$ and $q^{(2)}$. Next, a swapped prediction problem is addressed, where the codes derived from one view are predicted using the encoding from the second view. To achieve this, we minimize

$$\mathcal{L}_{\theta, \psi}^{\text{SwAV}} = \frac{1}{n} \sum_{i=1}^n \ell(q_i^{(1)}, y_i^{(2)}) + \ell(q_i^{(2)}, y_i^{(1)}), \quad (69)$$

where $\ell(q, y) = d_{\text{ce}}(q, \text{softmax}_\tau(C^\top y))$ quantifies the correspondence between the representation y and the code q for a temperature $\tau > 0$. For an overview of the architecture we refer to Fig. 19. Note that, although SwAV takes advantage of contrastive learning, it does not require the use of a large memory bank or a momentum network.

In addition to this method, the authors also propose the augmentation technique called *multi-crop*, which was also used for DINO (see Sect. 4.2). Instead of using two views with full resolution, a mixture of views with different resolutions is used. In this approach, multiple transformations are compared by using considerably smaller ones, which leads to a further improvement of previous methods such as SimCLR, DeepCluster and SeLa.

6.4 Advantages and disadvantages

Clustering-based methods for representation learning, such as DeepCluster, SeLa, and SwAV, offer distinct advantages and disadvantages due to their unique approaches to handling unlabeled data. These methods enhance the quality of learned representations by ensuring consistency among cluster assignments, with SwAV specifically promoting consistency across different views of the data (Caron et al., 2020). Additionally, clustering-based methods tend to be more memory-efficient than traditional contrastive learning approaches, as they eliminate the need for a large memory bank or a momentum network.

However, these methods also come with their own challenges. A common issue in naive clustering-based approaches is the tendency toward degenerate solutions, where all representations may be assigned to the same cluster (Asano et al., 2020). To address this, sophisticated techniques, such as the optimal transport problem in SeLa, are required to ensure a balanced distribution of data points across clusters, adding complexity to the

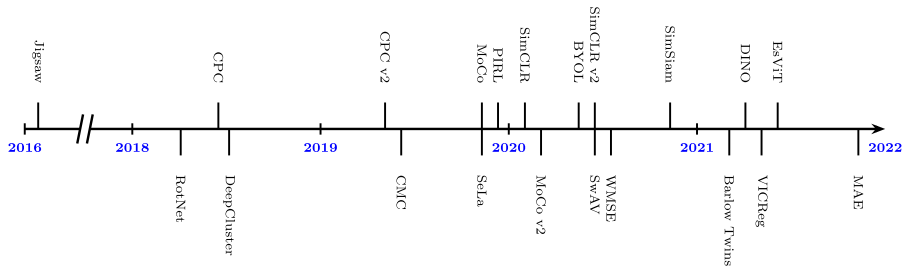


Fig. 20 Timeline of the first date of publication on arXiv for every method

implementation. Furthermore, clustering-based methods are sensitive to the choice of clustering algorithm and hyperparameters, which can significantly impact performance. While DeepCluster suggests that the choice of clustering algorithm is not crucial, in practice, selecting the appropriate number of clusters and other parameters often necessitates careful tuning and experimentation.

6.5 Relations to other methods

The idea of clustering-based representation learning is to exploit the coherence between image representations and their corresponding cluster assignments. We are interested in learning image representations which get assigned to the same cluster if the underlying images are semantically similar.

Other self-supervised methods, such as pretext task, information maximization or teacher-student methods build upon loss formulations that consider images and their transformations as separate classes. In contrast, clustering-based methods like DeepCluster and SeLa discriminate between pseudo-label assignments of similar representations instead of individual images. The only exception could be DINO (Caron et al., 2021), which we have mainly classified as a teacher-student method. On closer inspection, we can see that it also has some similarities with clustering-based methods. Here, the last layer of the projector can be considered as cluster prototypes, which act as subtle clustering of the output of the first part of the projector (Garrido et al., 2022).

The introduction of contrastive losses and memory banks also deviate from the notion of instance classes. However, these methods typically rely on a large number of pairwise feature comparisons, whereas linking with clustering-based methods, such as those done by Caron et al. (2020) in SwAV, can be much more memory efficient and can result in better performance.

7 Taxonomy of representation learning methods

As we have seen in this survey, there are several ways to learn meaningful representations of images. These include solving specific tasks for pre-training such as predicting the rotation angle of an image, maximizing the mutual information between different views of the same image, using a contrastive loss in order to separate positive and negative samples in latent space, learning from a teacher network and clustering and subsequently self-labeling

Table 2 Overview over the representation learning approaches discussed in this paper

Method	Class	Code
Autoencoders (Le Cun, 1987)	Pretext-Task	https://github.com/gidaris/FeatureLearningRotNet
RotNet (Gidaris et al., 2018)	Pretext-Task	https://github.com/MehdiNorozi/JigsawPuzzleSolver
Jigsaw (Norozi & Favaro, 2016)	Pretext-Task	https://github.com/facebookresearch/mae
MAE (He et al., 2022)	Pretext-Task	https://github.com/facebookresearch/dino
DINO (Caron et al., 2021)	Teacher-Student	https://github.com/microsoft/esvit
EsViT (Li et al., 2021)	Teacher-Student	https://github.com/deepmind/deepmind-research/tree/master/byol
BYOL (Grill et al., 2020)	Teacher-Student	https://github.com/Vicreg
VICReg (Bardes et al., 2021)	Info-Max	https://github.com/facebookresearch/barlowtwins
Barlow Twins (Zbontar et al., 2021)	Info-Max	https://github.com/facebookresearch/simsiam
SimSiam (Chen & He, 2021)	Info-Max	https://github.com/htdt/self-supervised
WMSE (Ermolov et al., 2021)	Info-Max	https://github.com/google-research/simclr
SimCLR (Chen et al., 2020a)	Contrastive	https://github.com/facebookresearch/moco
MoCo (He et al., 2020)	Contrastive	https://github.com/facebookresearch/moco
MoCo v2 (Chen et al., 2020b)	Contrastive	
CPC (van den Oord et al., 2018)	Contrastive/Info-Max	
CPC v2 (Henaff, 2020)	Contrastive/Info-Max	
CMC (Tian et al., 2020a)	Contrastive/Info-Max	https://github.com/HobbitLong/CMC
PIRL (Misra & Maaten, 2020)	Contrastive/Pretext-Task	https://github.com/facebookresearch/vissl/tree/main/projects/PIRL
DeepCluster (Caron et al., 2018)	Clustering	https://github.com/facebookresearch/deepluster
SeLa (Asano et al., 2020)	Clustering	https://github.com/yukimasano/self-label
SwAV (Caron et al., 2020)	Clustering	https://github.com/facebookresearch/swav

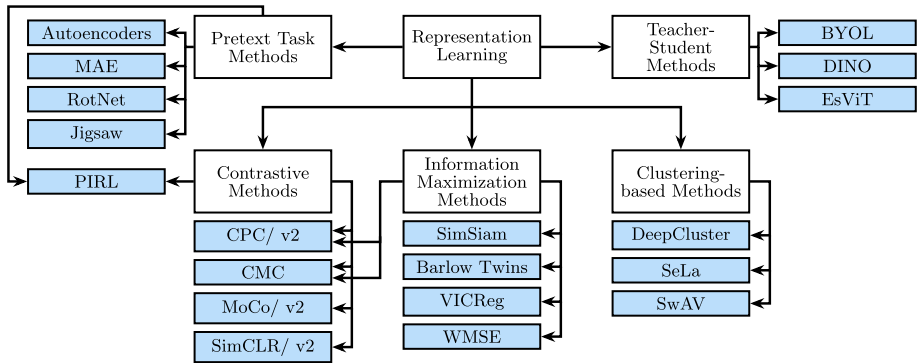


Fig. 21 Graphical overview of the taxonomy of image representation learning methods

images. Based on these distinctions, we adapt and expand the taxonomy, proposed by Bardes et al. (2021) in the following section, which includes the following five categories:

1. Pretext task methods
2. Information maximization methods
3. Teacher-student methods
4. Contrastive methods
5. Clustering-based methods

Note that some methods fit into multiple categories, as they combine different approaches. CPC (v2) and CMC, e.g., both use a contrastive loss, as well as information maximization. PIRL includes solving a pretext and a contrastive loss to learn representations. We used this taxonomy to structure our paper, Fig. 20 shows the chronological order in which all discussed methods have been published.

Figure 21 gives a visual overview on all methods within the proposed taxonomy. The inner nodes show the five categories that are connected to each reviewed method. Methods that can be assigned to several categories have multiple ingoing edges. As an additional overview, Table 2 lists all methods including their primary class assignment and the URL to the original implementation on Github,¹ if available.

8 Applications

Many of the described self-supervised paradigms and methods in this work serve as a foundation for solving challenging tasks in the area of machine learning. Specifically, the ability to improve models by simply adding more unlabelled data together with the learned emergent features have made self-supervised learning indispensable and ubiquitous in machine learning research and its applications. These self-supervised methods are increasingly used as foundation models to perform few-shot image classification, semantic segmentation, object detection and more (Ericsson et al., 2021), which proves useful in many machine learning tasks where these inductive biases are inherent. The vast majority of use cases

¹ <https://github.com>.

fall into the area of medical imaging, which is a well known field of application for computer vision and therefore self-supervised learning, but other use cases also include fields such as satellite imagery, audio and reinforcement learning. In the following, we show how self-supervised learning methods presented in this work have been used for other areas of research in machine learning. Note that while we highlight several works in this section, the list is not exhaustive.

8.1 Pretext task methods

The use of pretext task methods covers a wide range of applications, particularly in the medical imaging field. Bai et al. (2019) improve image segmentation by predicting the correct positions of anatomical features in magnetic resonance imaging (MRI) scans. The pretext task is similar to the jigsaw puzzle task described in Sect. 2.3 and consists in finding the correct positions defined by nine bounding boxes. The labels are provided automatically by the MRI scans. Chen et al. (2019) adopt a masked autoencoder architecture, where they swap random image patches and then reconstruct the original image. This approach aims to learn good representations for downstream tasks such as multi-organ localization on fetal 2D ultrasound images. To improve medical image segmentation, Zhou et al. (2023) first pre-train a ViT with a masked autoencoder on a target dataset and then finetune it on the respective downstream segmentation tasks. They demonstrate that masked autoencoder pre-training enhances the performance in several medical image tasks such as chest X-ray disease classification, abdominal computed tomography (CT) multi-organ segmentation, and MRI brain tumor segmentation. Chen et al. (2023c) study the effectiveness of masked autoencoders on tasks such as 3D segmentation on abdominal CT scans, brain tumor MRI images, and chest CT scans with COVID19 infections.

8.2 Information maximization methods

Despite being fairly new, information maximization methods such as Barlow Twins and VICReg found successful application both in medical imaging and reinforcement learning. Sheng et al. (2024) propose the BarlowTwins-CXR approach for chest X-ray abnormality localization, which consists of a two-stage training process. In the first stage, self-supervised pre-training is performed using an adjusted Barlow Twins algorithm on the National Institutes of Health (NIH) dataset. In the second stage, the pre-trained encoder is used for fine-tuning on the VinDr-CXR dataset. The authors improve upon traditionally trained ImageNet baselines and show increased precision in localizing chest abnormalities. Pun and Agarwal (2022) similarly use the Barlow Twins mechanism to pre-train the encoder of a U-Net, aiming to learn good representations. Their results indicate that combining Barlow Twins with U-Nets improves biomedical image segmentation compared to using a standard U-Net. In reinforcement learning, Pechác et al. (2024) analyze random network distillation, a popular exploration method, to study whether representations learned with VICReg are better than random representations for exploration.

8.3 Teacher-student methods

Teacher-student methods, especially BYOL and DINO, have found application in various research areas including medical imaging, audio signal processing, and reinforcement learning.

In medical image recognition, Feng et al. (2024) leverage a semi-supervised learning framework that uses BYOL to learn representations that are clustered into pseudo-labels. The pseudo-labeled data is then combined with the labeled data to improve classification performance. Zeng et al. (2023) improve the selection of positive samples by extending them to include augmentations from different images with the same label. Niizumi et al. (2023) use BYOL to learn robust audio representations for the task of heart murmur detection. Mojtahedi et al. (2023) apply SimSiam to learn better representations for classification of liver cancers. Pérez-García et al. (2024) leverage DINO to learn a general purpose medical image encoder, solely learning on images where previously data with multi-modal annotations were considered state-of-the-art. Jimenez-Perez et al. (2024) apply embeddings extracted from DINO to condition the generation process for a latent diffusion model (LDF), thus also eliminating the reliance on additional annotations.

In reinforcement learning, Guo et al. (2022b) are able to improve exploration by using BYOL to learn state-action pair representations, where the reconstruction error between observations of subsequent states indicates novelty in the environment. Wagner and Harmeling (2024) also use DINO embeddings to cluster similar states in partially observed 3D environments. The clustered states can then be counted allowing for a better description of the state-space distribution.

8.4 Contrastive methods

Contrastive learning being one of the earlier self-supervised paradigms has found many applications in computer vision. Chen et al. (2023b) use contrastive predictive coding for underwater image enhancement. The authors perform contrastive learning on a feature level by defining positive samples as image features of high-quality images and negatives as image features of low-quality images. The anchor is simply the embedding of a low-quality input image that was fed through a neural network. The objective is to push this anchor towards the high-quality features and repel the low-quality features.

Chen et al. (2023a) combine a masked autoencoder with contrastive representation learning to align masked lung X-ray images with text reports. Nguyen et al. (2021) apply contrastive predictive coding to model-based reinforcement learning. To become more robust against irrelevant noise the authors maximize the mutual information between the past latent codes and actions against the future latent code, which encodes predictable components from the environment. Contrastive predictive coding was used by de Haan and Löwe (2021) to perform anomaly detection and segmentation of high resolution images of objects that are placed on various backgrounds. This setting is very relevant in industrial visual inspection.

8.5 Clustering-based methods

Self-supervised clustering methods have also found use in medical imaging tasks. Kart et al. (2021) propose DeepMCat which builds on DeepCluster to cluster large-scale datasets of cardiac magnetic resonance (CMR) images. The authors show that clustering

Table 3 Overview on datasets (ImageNet, Pascal VOC, Microsoft COCO, CIFAR and Places-205) and tasks (image classification, object detection and instance segmentation) each representation learning method has been evaluated on

	ImageNet	VOC IC	VOC OD	VOC Seg	COCO OD	COCO Seg	CIFAR 10	CIFAR 100	Places 205	# Total
RotNet	✓	✓	✓	✓			✓		✓	6
Jigsaw	✓	✓	✓	✓						4
DINO	✓						✓	✓		3
MAE	✓				✓	✓		✓	✓	4
EsViT	✓	✓			✓	✓	✓	✓		6
BYOL	✓	✓	✓	✓			✓			6
VICReg	✓	✓	✓		✓	✓		✓	✓	6
Barlow Twins	✓	✓	✓		✓	✓				5
SimSiam	✓		✓		✓	✓				4
WMSE	✓						✓	✓		3
SimCLR	✓	✓					✓	✓	✓	5
MoCo	✓		✓							4
MoCo v2	✓		✓		✓	✓				2
CPC	✓									1
CPC v2	✓		✓							2
CMC	✓									1
PIRL	✓	✓	✓						✓	4
DeepCluster	✓	✓	✓	✓					✓	5
SeLa	✓	✓	✓	✓			✓	✓	✓	7
SwAV	✓	✓	✓		✓	✓			✓	6
# Total	20	11	13	5	7	7	8	7	8	

The numbers underneath indicate how many of the shown 20 methods used the corresponding dataset for evaluation. The numbers on the right indicate on how many of the presented datasets each method has been evaluated

Table 4 Top-1 and top-5 accuracy on ImageNet classification for a linear evaluation (on the left) and semi-supervised learning, where the classifier is fine-tuned on 1% and 10% respectively of the labeled ImageNet data (on the right)

	#Params	Top-1	Top-5	Top-1		Top-5	
Method				1%	10%	1%	10%
<i>ResNet-50</i>							
Supervised ³	24 M	76.9	93.3	25.4	56.4	48.4	80.4
DINO	24 M	75.3	—	—	—	—	—
BYOL	24 M	74.3	91.6	53.2	68.8	78.4	89.0
VICReg	24 M	73.2	91.1	54.8	69.5	79.4	89.5
Barlow Twins	24 M	73.2	91.0	55.0	69.7	79.2	89.3
SimSiam	24 M	71.3	—	—	—	—	—
SimCLR	24 M	69.3	89.0	48.3	65.6	75.5	87.8
MoCo	24 M	60.6	—	—	—	—	—
MoCo v2	24 M	71.1	90.1	—	—	—	—
CPC v2	24 M	63.8	85.3	—	-	—	—
CMC ⁴	24 M	66.2	87.0				
PIRL	24 M	63.6	—	30.7	60.4	57.2	83.8
SeLa	24 M	61.5	84.0				
SwAV with multi-crop	24 M	75.3	—	53.9	70.2	78.5	89.9
<i>other architectures</i>							
RotNet ⁵ (AlexNet)		55.4	77.9	—	—	—	—
Jigsaw ⁵ (AlexNet)		44.6	68.0	—	—	—	—
MAE (ViT-H)	643 M	76.6	—	—	—	—	—
DINO (ViT-B/8)	85 M	80.1	-	-	-	-	-
EsViT (Swin-B)	87 M	80.4	-	-	-	-	-
BYOL (ResNet-200x2)	250 M	79.6	94.8	71.2	77.7	89.5	93.7
WMSE (ResNet-18)		79.0	94.5	—	—	—	—
SimCLR (ResNet-50x4)	375 M	76.5	93.2	—	—	85.8	92.6
MoCo (ResNet-50x4)	375 M	68.6	—	—	—	—	—
CPC (ResNet v2-101)	28 M	48.7	73.6	—	—	—	—
CPC v2 (ResNet-161)	305 M	71.5	90.1	-	—	—	—
CPC v2 (ResNet-33)		—	-	52.7	73.1	78.3	91.2
CMC ³ (ResNet-50x2)	188 M	70.6	89.7	—	—	—	—
DeepCluster (AlexNet)	62 M	41.0	—	—	—	—	—

³Reported numbers for top-1 and top-5 accuracy from Pham et al. , for top-1 and top-5 accuracy trained only on 1% and 10%, respectively, from Zbontar et al.

⁴Trained with RandAugment (Cubuk et al., 2020)

⁵Reported numbers are from Kolesnikov et al. (2019)

The upper part shows performance for a ResNet-50 encoder and the lower part shows more results where other architectures have been used. We also report the number of parameters for each network

approaches that use basic deep neural network architectures are effective for this purpose. Xiao and Lu (2023) alternately run a semi-supervised framework together with DeepCluster to improve medical image classification, where the combination of supervised labels and deep clustering is effective for downstream classification. Lafabregue et al.

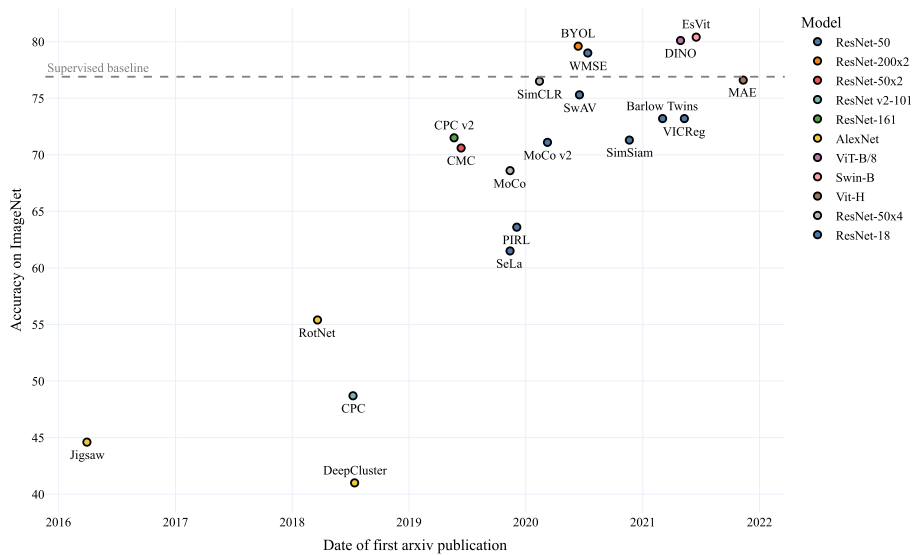


Fig. 22 Reviewed methods top-1 accuracy on ImageNet, sorted by their first publication on arXiv². The gray line marks the supervised benchmark

(2022) and Guo et al. (2022a) study how DeepCluster among other methods can be applied to satellite image time series.

Some studies have also compared various methods regarding their transfer learning ability. Ericsson et al. (2021) investigate how MoCo, PIRL, SimCLR, SeLa, BYOL,

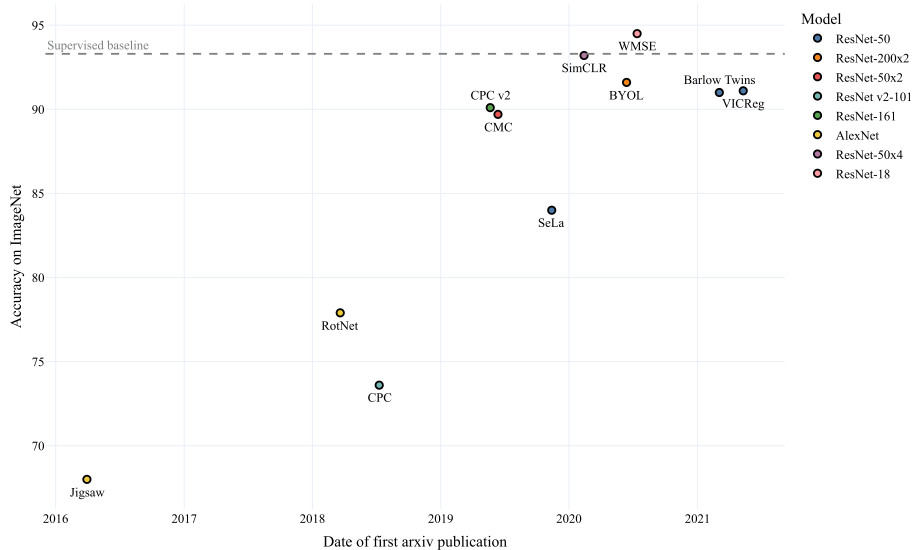


Fig. 23 Reviewed methods top-5 accuracy on ImageNet, sorted by their first publication on arXiv²

DeepCluster, SwAV and other methods perform on 40 transfer learning tasks covering many-shot and few-shot image classification, object detection, surface normal prediction, and semantic segmentation (natural scenes, medical imaging, agricultural datasets, and more). Anton et al. (2022) also perform a comparative study of BYOL, MoCo, PIRL, SwAV, and SimCLR on different medical imaging datasets.

9 Meta-study of quantitative results

Assessing the quality of the learned representations can be tricky. One approach that has become established in literature is to evaluate the obtained representations on downstream computer vision tasks, such as image classification, object detection, or instance segmentation. In this section we explain the evaluation process for representation learning methods and take a closer look at the most common evaluation tasks in literature. We compare all reviewed methods with regard to their performance and summarize the results for three different datasets. For experimental details please consider the original works. We perform a quantitative comparison and provide some insights on potential future directions to further evaluate and compare the reviewed methods.

9.1 Evaluation of representation learning methods

The performance of representation learning models is often measured and compared by letting pre-trained models solve downstream tasks such as image classification. For pre-training, a base architecture is chosen as encoder and trained in a self-supervised manner without the use of labels. Many authors experiment with multiple architectures. One default architecture for image classification is the ResNet-50 (He et al., 2016), newer methods often use Vision Transformers (ViT) (Dosovitskiy et al., 2021) as encoder. The learned representations are then evaluated on different downstream tasks. We give more details on the evaluation protocols later in this section.

We identified five datasets that were most commonly used to evaluate representation learning methods: ImageNet (Russakovsky et al., 2015), the Pascal visual object classes (VOC) (Everingham et al., 2009), Microsoft common objects in context (COCO) (Lin et al., 2014), CIFAR-10, CIFAR-100 (Krizhevsky et al., 2009) and Places205 (Zhou et al., 2014). All listed datasets include one or multiple of the following tasks: image classification (IC), object detection (OD) and instance segmentation (Seg). Table 3 shows which of the most common datasets are used for evaluating each method.

To get a sense of the performance of all reviewed methods at a glance, we conduct a quantitative comparison in the following. We report the evaluation results for ImageNet, Pascal VOC and Microsoft COCO and identify gaps where further evaluation would be interesting. All methods have been pre-trained on the ImageNet training set before further evaluation.

9.1.1 ImageNet

Image classification on ImageNet, which includes 1000 different classes and has established as an evaluation standard for representation learning methods, is mostly evaluated in two ways. A linear classifier is either trained on top of the frozen pre-trained representations, or the model weights are initialized with the pre-trained weights and fine-tuned on

Table 5 Evaluation of a linear classifier on the PASCAL VOC image classification (IC), object detection (OD) and instance segmentation (Seg) and COCO OD and Seg tasks

Method	Pascal VOC					COCO					
	IC		OD			Seg			Seg		
	AP	AP _{all}	AP ₅₀	AP ₇₅	mIoU	AP _{all} ^{BB}	AP ₅₀ ^{BB}	AP ₇₅ ^{BB}	AP _{all} ^{MK}	AP ₅₀ ^{MK}	AP ₇₅ ^{MK}
RotNet ⁵	72.9	54.4	–	–	39.1	–	–	–	–	–	–
Jigsaw ⁵	67.6	53.2	–	–	37.6	–	–	–	–	–	–
MAE	–	–	–	–	–	53.3	–	–	47.2	–	–
EsViT	85.5	–	–	–	–	46.2	68.0	50.6	41.6	64.9	44.8
BYOL	85.4	–	77.5	–	76.3	–	–	–	–	–	–
VICReg	86.6	–	82.4	–	–	39.4	–	–	36.4	–	–
Barlow Twins	86.2	56.8	82.6	63.4	–	39.2	59.0	42.5	34.3	56.0	36.5
SimSiam	–	57.0	82.4	63.7	–	39.2	59.3	42.1	34.4	56.0	36.7
SimCLR	80.5	–	–	–	–	–	–	–	–	–	–
MoCo	–	55.9	81.5	62.6	–	40.8	61.6	44.7	36.9	58.4	39.7
MoCo v2	–	57.4	82.5	64.0	–	–	–	–	–	–	–
CPC v2	–	76.6	–	–	–	–	–	–	–	–	–
PIRL	81.1	54.0	80.7	59.7	–	–	–	–	–	–	–
DeepCluster	73.7	65.9	–	–	45.1	–	–	–	–	–	–
SeLa	75.9	57.8	–	–	44.7	–	–	–	–	–	–
SwAV with multi-crop	88.9	56.1	82.6	62.7	–	41.6	62.3	45.5	37.8	59.6	40.5

⁶Uses a data dependent initialization as proposed by Krähenbühl et al. (2015)

For IC we report the mean average precision (mAP), the AP_{all}, AP₅₀ and AP₇₅ for the VOC OD, COCO OD and COCO Seg and for PASCAL VOC Seg the mean intersection over union (mIoU). We report the best values reported in the original papers using different architectures

1% and 10% of the labeled ImageNet training data, respectively. Table 4 shows the accuracy for each method either using a ResNet-50 encoder for better comparability or a different architecture. We report both the top-1, as well as the top-5 accuracies. The top-1 accuracy refers to the proportion of samples that were correctly classified by the model. The top-5 accuracy measures the proportion of samples that have the target label in the top five predictions of the model. Among the methods evaluated with a ResNet-50, DINO and SwAV perform best, nearly reaching the performance of a supervised trained ResNet50. Considering the reported top-5 accuracy, BYOL performs best, right before VICReg and Barlow Twins. For other, bigger architectures, EsViT and DINO both perform best, while utilizing architectures with a comparatively little number of parameters.

Figure 22 shows the best reported top-1 accuracy of all methods on the ImageNet validation set in chronological order of their first publication on arXiv²,² Fig. 23 shows the top-5 accuracy, respectively. The gray line shows the supervised baseline. It would be interesting to measure the performance of every method under the same conditions to make them comparable. Nevertheless, the plot reveals some interesting points, e.g., that CPC v2 and CMC perform better than most other early published self-supervised methods and EsViT using the Swin-B architecture has not been beaten by any of the compared methods.

² <https://arxiv.org/>.

The goal of every representation learning method is to extract meaningful features from images that are useful for various tasks. The quality of extracted features learned on the ImageNet data can therefore be further assessed by transferring them to solve tasks on other datasets like the Pascal VOC and COCO object detection and instance segmentation. To evaluate the features on other tasks, the learned weights serve as initialization for a network and a linear classifier is trained on top, while the network layers are fine-tuned. Usually, the features are extracted from different layers of the network, while freezing the weights of the others. The best values are reported for each method. In the following we take a closer look at the results on the Pascal VOC and Microsoft COCO datasets. The goal when performing object detection on these two datasets is to predict bounding boxes for every object shown on the image. The task of instance segmentation is conducted pixel-wise, where every pixel is classified separately.

9.1.2 Pascal VOC

With only 20 classes, the Pascal VOC (Everingham et al., 2009) dataset is rather small and designed to resemble a real world setting. The data includes annotations for image classification, object detection, as well as instance segmentation. The standard metrics for the classification and object detection task are mean average precision (AP) with different intersection over union (IoU) threshold values. For the task of segmentation the mean IoU is reported. For a detailed overview on the aforementioned metrics we refer to the work of Padilla et al. (2020).

The first five columns of Table 5 show the results for the Pascal VOC tasks side by side for different architectures. For the object detection task Fast R-CNN (Girshick, 2015) and Faster R-CNN (Ren et al., 2015) are most widely used. SwAV with multicrop performs best on the image classification task, just before ViT and Barlow Twins. Barlow Twins and SwAV also perform best on the object detection task. From all methods that have been evaluated on the segmentation task, BYOL, by far, outperforms every other method. However, the results cannot be deemed as representative due to a lack of comparative values for other methods.

9.1.3 COCO

The Microsoft COCO dataset is a large dataset for object detection and segmentation including objects from 91 different classes, depicted in their natural setting. In the second half of Table 5 we present the average precision for object detection and instance segmentation. In the case of object detection it is the bounding box (BB) AP and in case of the segmentation it is the AP of the segmentation mask (MK). Note that we again report values for different encoder architectures. The most commonly used architecture for both object detection and segmentation is a Mask R-CNN (He et al., 2017) model with either the C4 or feature pyramid network encoder (Lin et al., 2017). For both tasks the overall AP is best for MAE, other values are missing and EsViT outperforms every other method for the AP50 and AP75. Again, for a profound insight a more detailed experimental evaluation is necessary.

9.2 Future directions

To conclude our quantitative meta-study we want to point out some interesting insights and suggest some experiments to conduct in the future.

As the comparison of all representation learning methods has shown, the performance strongly depends on the used network architecture. Some architectures, datasets and tasks have been utilized throughout various works to evaluate the quality of self-supervised trained image representations. Nevertheless, there is no standardized benchmark for a consistent comparison of methods, yet. Goyal et al. (2019) suggest a range of tasks as benchmark to cover many aspects and obtain a detailed comparison of methods in the future.

One main contribution of this meta-study is the categorization and comprehensive presentation of different approaches to the overall aim to learn meaningful representations of images. In Sect. 7 five main categories have been described into which different approaches can be grouped. There already are some methods that combine multiple approaches that work well, which shows that the combination of different representation learning methods holds potential for future research.

10 Conclusion

The goal of self-supervised representation learning is to extract meaningful features from unlabeled image data and use them to solve all kinds of downstream tasks. In this work we saw different strategies for representation learning and how they are related. We gave an extensive overview of methods that have been developed over the last years and adapted a framework to categorize them.

Author Contributions All authors reviewed the literature, wrote the main manuscript text, and edited the manuscript. JR and TU prepared the figures. TU supervised the project.

Funding Open Access funding enabled and organized by Projekt DEAL.

Declarations

Funding Not applicable.

Conflict of interest None.

Ethics approval Not applicable.

Consent to participate Not applicable.

Consent for publication Not applicable.

Availability of data and material Not applicable.

Code availability Not applicable.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

References

- Anton, J., Castelli, L., Chan, M. F., Outters, M., Tang, W. H., Cheung, V., Shukla, P., Walambe, R., & Kotcha, K. (2022). How well do self-supervised models transfer to medical imaging? *Journal of Imaging*, 8(12), 2313–433X.
- Asano, Y. M., Rupprecht, C., & Andrea, V. (2020). Self-labelling via simultaneous clustering and representation learning. In *International conference on learning representations*.
- Assran, M., Caron, M., Misra, I., Bojanowski, P., Joulin, A., Ballas, N., & Rabbat, M. (2021). Semi-supervised learning of visual features by non-parametrically predicting view assignments with support samples. In *2021 IEEE/CVF international conference on computer vision (ICCV)* (pp. 8423–8432).
- Ba, J. L., Kiros, J. R., & Hinton, G. E. (2016). Layer normalization. [arXiv:1607.06450](https://arxiv.org/abs/1607.06450)
- Bai, W., Chen, C., Tarroni, G., Duan, J., Guitton, F., Petersen, S. E., Guo, Y., Matthews, P. M., & Rueckert, D. (2019). Self-supervised learning for cardiac mr image segmentation by anatomical position prediction. In Shen, D., Liu, T., Peters, T. M., Staib, L. H., Essert, C., Zhou, S., Yap, P.-T., & Khan, A. (Eds.), *Medical image computing and computer assisted intervention—MICCAI 2019* (pp. 541–549). Springer. ISBN 978-3-030-32245-8
- Bardes, A., Ponce, J., & LeCun, Y. (2021). Vicreg: Variance-invariance-covariance regularization for self-supervised learning. In *International conference on learning representations*.
- Barlow, H. B. (1961). Possible principles underlying the transformation of sensory messages. *Sensory Communication*, 1(01).
- Bastos, A. M., Usrey, W. M., Adams, R. A., Mangun, G. R., Fries, P., & Friston, K. J. (2012). Canonical microcircuits for predictive coding. *Neuron*, 76(4), 695–711.
- Botteghi, N., Poel, M., & Brune, C. (2022). Unsupervised representation learning in deep reinforcement learning: A review. [arXiv:2208.14226](https://arxiv.org/abs/2208.14226)
- Bromley, J., Guyon, I., LeCun, Y., Säckinger, E., & Shah, R. (1993). Signature verification using a “Siamese” time delay neural network. *Advances in Neural Information Processing Systems*, 6.
- Caron, M., Bojanowski, P., Joulin, A., & Douze, M. (2018). Deep clustering for unsupervised learning of visual features. In *Proceedings of the European conference on computer vision (ECCV)* (pp. 132–149).
- Caron, M., Misra, I., Mairal, J., Goyal, P., Bojanowski, P., & Joulin, A. (2020). Unsupervised learning of visual features by contrasting cluster assignments. *Advances in Neural Information Processing Systems*, 33, 9912–9924.
- Caron, M., Touvron, H., Misra, I., Jégou, H., Mairal, J., Bojanowski, P., & Joulin, A. (2021). Emerging properties in self-supervised vision transformers. In *Proceedings of the IEEE/CVF international conference on computer vision* (pp. 9650–9660).
- Chen, C., Zhong, A., Wu, D., Luo, J., & Li, Q. (2023a). Contrastive masked image-text modeling for medical visual representation learning. In *International conference on medical image computing and computer-assisted intervention* (pp. 493–503). Springer.
- Chen, L., Bentley, P., Mori, K., Misawa, K., Fujiwara, M., & Rueckert, D. (2019). Self-supervised learning for medical image analysis using image context restoration. *Medical Image Analysis*, 58(101539), 101539.
- Chen, R., Cai, Z., & Yuan, J. (2023). Uiesc: An underwater image enhancement framework via self-attention and contrastive learning. *IEEE Transactions on Industrial Informatics*, 19(12), 11701–11711.
- Chen, T., Kornblith, S., Norouzi, M., & Hinton, G. (2020a). A simple framework for contrastive learning of visual representations. In *International conference on machine learning* (pp. 1597–1607). PMLR.
- Chen, X., & He, K. (2021). Exploring simple siamese representation learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 15750–15758).
- Chen, X., Fan, H., Girshick, R., & He, K. (2020b). Improved baselines with momentum contrastive learning. [arXiv:2003.04297](https://arxiv.org/abs/2003.04297)
- Chen, Z., Agarwal, D., Aggarwal, K., Safta, W., Balan, M., & Brown, K. (2023c). Masked image modeling advances 3d medical image analysis. In *2023 IEEE/CVF winter conference on applications of computer vision (WACV)* (pp. 1969–1979). IEEE Computer Society.
- Cubuk, E. D., Zoph, B., Shlens, J., & Le, Q. V. (2020). Randaugment: Practical automated data augmentation with a reduced search space. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops* (pp. 702–703).
- Cuturi, M. (2013). Sinkhorn distances: Lightspeed computation of optimal transport. In C. J. Burges, L. Bottou, M. Welling, Z. Ghahramani, & K. Q. Weinberger (Eds.), *Advances in neural information processing systems*. (Vol. 26). Curran Associates Inc.
- de Haan, P., & Löwe, S. (2021). Contrastive predictive coding for anomaly detection. [arXiv:2107.07820](https://arxiv.org/abs/2107.07820)

- Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., & Houlsby, N. (2021). An image is worth 16x16 words: Transformers for image recognition at scale. In *International conference on learning representations*.
- Eldele, E., Ragab, M., Chen, Z., Wu, M., Kwok, C. K., Li, X., & Guan, C. (2021). Time-series representation learning via temporal and contextual contrasting. In *Proceedings of the thirtieth international joint conference on artificial intelligence (IJCAI-21)*.
- Ericsson, L., Gouk, H., & Hospedales, T. M. (2021). How well do self-supervised models transfer? In *2021 IEEE/CVF conference on computer vision and pattern recognition (CVPR)* (pp. 5410–5419).
- Ermolov, A., & Sebe, N. (2020). Latent world models for intrinsically motivated exploration. *Advances in Neural Information Processing Systems*, 33, 5565–5575.
- Ermolov, A., Siarohin, A., Sangineto, E., & Sebe, N. (2021). Whitening for self-supervised representation learning. In *International conference on machine learning* (pp. 3015–3024). PMLR.
- Everingham, M., Van Gool, L., Williams, C. K. I., Winn, J., & Zisserman, A. (2009). The pascal visual object classes (VOC) challenge. *International Journal of Computer Vision*, 88, 303–308.
- Feng, H., Jia, Y., Xu, R., Prasad, M., Anaissi, A., & Braytee, A. (2024). Integration of self-supervised BYOL in semi-supervised medical image recognition. [arXiv:2404.10405](https://arxiv.org/abs/2404.10405)
- Fetterman, A., & Albrecht, J. (2020). Understanding self-supervised and contrastive learning with bootstrap your own latent (BYOL).
- Garrido, Q., Chen, Y., Bardes, A., Najman, L., & Lecun, Y. (2022). On the duality between contrastive and non-contrastive self-supervised learning. [arXiv:2206.02574](https://arxiv.org/abs/2206.02574)
- Geiping, J., Garrido, Q., Fernandez, P., Bar, A., Pirsaviash, H., LeCun, Y., & Goldblum, M. (2023). A cookbook of self-supervised learning. [arXiv:2304.12210](https://arxiv.org/abs/2304.12210)
- Gidaris, S., Singh, P., & Komodakis, N. (2018). Unsupervised representation learning by predicting image rotations. In *International conference on learning representations*.
- Girshick, R. (2015). Fast R-CNN. In *Proceedings of the IEEE international conference on computer vision* (pp. 1440–1448).
- Gou, J., Yu, B., Maybank, S. J., & Tao, D. (2021). Knowledge distillation: A survey. *International Journal of Computer Vision*, 129(6), 1789–1819.
- Goyal, P., Mahajan, D., Gupta, A., & Misra, I. (2019). Scaling and benchmarking self-supervised visual representation learning. In *Proceedings of the IEEE/CVF international conference on computer vision* (pp. 6391–6400).
- Grill, J.-B., Strub, F., Altché, F., Tallec, C., Richemond, P., Buchatskaya, E., Doersch, C., Pires, A., Bernardo, G., Zhaohan Mohammad, G. A., et al. (2020). Bootstrap your own latent-a new approach to self-supervised learning. *Advances in Neural Information Processing Systems*, 33, 21271–21284.
- Grover, A., & Leskovec, J. (2016). node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining* (pp. 855–864).
- Guo, W., Zhang, W., Zhang, Z., Tang, P., & Gao, S. (2022a). Deep temporal iterative clustering for satellite image time series land cover analysis. *Remote Sensing*, 14(15).
- Guo, Z., Thakoor, S., Pislár, M., Pires, A., Bernardo, A., Florent, T., Corentin, S., Alaa, C., Daniele, G., Jean-Bastien, T., Yunhao, V., Michal, M., Remi Mohammad, G. A., & Piot, B. (2022). Byol-explore: Exploration by bootstrapped prediction. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, & A. Oh (Eds.), *Advances in neural information processing systems* (Vol. 35, pp. 31855–31870). Curran Associates Inc.
- Gutmann, M., & Hyvärinen, A. (2010). Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics* (pp. 297–304). JMLR workshop and conference proceedings.
- He, K., Chen, X., Xie, S., Li, Y., Dollár, P., & Girshick, R. (2022). Masked autoencoders are scalable vision learners. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 16000–16009).
- He, K., Fan, H., Wu, Y., Xie, S., & Girshick, R. (2020). Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 9729–9738).
- He, K., Gkioxari, G., Dollár, P., & Girshick, R. (2017). Mask R-CNN. In *Proceedings of the IEEE international conference on computer vision* (pp. 2961–2969).
- He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 770–778).
- Henaff, O. (2020). Data-efficient image recognition with contrastive predictive coding. In *International conference on machine learning* (pp. 4182–4192). PMLR.

- Huang, Y., & Rao, R. P. N. (2011). Predictive coding. *Wiley Interdisciplinary Reviews: Cognitive Science*, 2(5), 580–593.
- Huang, Z., Jin, X., Lu, C., Hou, Q., Cheng, M.-M., Fu, D., Shen, X., & Feng, J. (2022). Contrastive masked autoencoders are stronger vision learners. [arXiv:2207.13532](#)
- Ioffe, S., & Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning* (pp. 448–456). PMLR.
- Jimenez-Perez, G., Osorio, P., Cersovsky, J., Montalt-Tordera, J., Hooge, J., Vogler, S., & Mohammadi, S. (2024). DiNO-diffusion. Scaling medical diffusion via self-supervised pre-training. [arXiv:2407.11594](#)
- Kart, T., Bai, W., Glocker, B., & Rueckert, D. (2021). Deepmcat: Large-scale deep clustering for medical image categorization. In Engelhardt, S., Oksuz, I., Zhu, D., Yuan, Y., Mukhopadhyay, A., Heller, N., Huang, S. X., Nguyen, H., Sznitman, R., & Xue, Y., (Eds.), *Deep generative models and data augmentation, labelling, and imperfections* (pp. 259–267). Springer. ISBN 978-3-030-88210-5
- Kenton, J. D., Chang, M.-W., & Toutanova, L. K. (2019). Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of NAACL-HLT* (pp. 4171–4186).
- Kingma, D. P., & Welling, M. (2013). Auto-encoding variational Bayes. [arXiv:1312.6114](#)
- Kipf, T. N., & Welling, M. (2016). Variational graph auto-encoders. [arXiv:1611.07308](#)
- Kolesnikov, A., Zhai, X., & Beyer, L. (2019). Revisiting self-supervised visual representation learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 1920–1929).
- Krähenbühl, P., Doersch, C., Donahue, J., & Darrell, T. (2015). Data-dependent initializations of convolutional neural networks. [arXiv:1511.06856](#)
- Krizhevsky, A., Hinton, G., et al. (2009). Learning multiple layers of features from tiny images.
- Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2017). ImageNet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6), 84–90.
- Lafabregue, B., Puissant, A., Weber, J., & Forestier, G. (2022). Deep clustering methods study applied to satellite images time series. In *IEEE international geoscience and remote sensing symposium (IGARSS)* (pp. 195–198). IEEE
- Le Cun, Y. (1987). Modèles connexionnistes de l'apprentissage. Ph.D thesis, Paris 6.
- Li, A. C., Efros, A. A., & Pathak, D. (2022). Understanding collapse in non-contrastive siamese representation learning. In *European conference on computer vision* (pp. 490–505). Springer.
- Li, C., Yang, J., Zhang, P., Gao, M., Xiao, B., Dai, X., Yuan, L., & Gao, J. (2021). Efficient self-supervised vision transformers for representation learning. In *International conference on learning representations*.
- Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., & Wierstra, D. (2019). Continuous control with deep reinforcement learning.
- Lin, M., Chen, Q., & Yan, S. (2013). Network in network. [arXiv:1312.4400](#)
- Lin, T.-Y., Dollár, P., Girshick, R., He, K., Hariharan, B., & Belongie, S. (2017). Feature pyramid networks for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 2117–2125).
- Lin, T.-Y., Maire, M., Belongie, S., Hays J., Pietro, P., Deva, R., Piotr, D., & Lawrence, Z. C. (2014). Microsoft COCO: Common objects in context. In *Proceedings of the European conference on computer vision (ECCV) 2014* (pp. 740–755). Springer.
- Liu, K.-J., Suganuma, M., & Okatani, T. (2022). Bridging the gap from asymmetry tricks to decorrelation principles in non-contrastive self-supervised learning. In Oh, A. H., Agarwal, A., Belgrave, D., & Cho, K. (Eds.), *Advances in neural information processing systems*.
- Lloyd, S. (1982). Least squares quantization in PCM. *IEEE Transactions on Information Theory*, 28(2), 129–137.
- Loshchilov, I., & Hutter, F. (2017). SGDR: Stochastic gradient descent with warm restarts. In *International conference on learning representations*.
- Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013a). Efficient estimation of word representations in vector space. [arXiv:1301.3781](#)
- Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., & Dean, J. (2013b). Distributed representations of words and phrases and their compositionality. *Advances in Neural Information Processing Systems*, 26.
- Misra, I., & Maaten, L. van der. (2020). Self-supervised learning of pretext-invariant representations. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 6707–6717).
- Mojtahedi, R., Hamghalam, M., Jarnagin, W. R., Do, R. K. G., & Simpson, A. L. (2023). Leveraging contrastive learning with simsiam for the classification of primary and secondary liver cancers. In *International conference on medical image computing and computer-assisted intervention* (pp. 311–321). Springer.
- Nguyen, T. D., Shu, R., Pham, T., Bui, H., & Ermon, S. (2021). Temporal predictive coding for model-based planning in latent space. In *International conference on machine learning* (pp. 8130–8139). PMLR.

- Niizumi, D., Takeuchi, D., Ohishi, Y., Harada, N., & Kashino, K. (2023). Byol for audio: Exploring pre-trained general-purpose audio representations. *IEEE/ACM Transactions on Audio, Speech and Language Processing*, 31, 137–151.
- Noroozi, M., & Favaro, P. (2016). Unsupervised learning of visual representations by solving jigsaw puzzles. In *European conference on computer vision* (pp. 69–84). Springer.
- Nozawa, K., & Sato, I. (2022). Evaluation methods for representation learning: A survey. In *Proceedings of the thirty-first international joint conference on artificial intelligence (IJCAI-22)—survey track* (pp. 5556–5563).
- Oquab, M., Darcet, T., Moutakanni, T., Vo, H. V., Szafraniec, M., Khalidov, V., Fernandez, P., HAZIZA, D., Massa, F., El-Nouby, A., Assran, M., Ballas, N., Galuba, W., Howes, R., Huang, P.-Y., Li, S.-W., Misra, I., Rabbat, M., Sharma, V., Synnaeve, G., Xu, H., Jegou, H., Mairal, J., Labatut, P., Joulin, A., & Bojanowski, P. (2024). DINOv2: Learning robust visual features without supervision. *Transactions on Machine Learning Research*.
- Ozbulak, U., Lee, H. J., Boga, B., Anzaku, E. T., Park, H., Van Messem, A., De Neve, W., & Vankerschaver, J. (2023). Know your self-supervised learning: A survey on image-based generative and discriminative training. [arXiv:2305.13689](https://arxiv.org/abs/2305.13689)
- Padilla, R., Netto, S. L., & da Silva, E. A. B. (2020). A survey on performance metrics for object-detection algorithms. In *2020 International conference on systems, signals and image processing (IWSSIP)* (pp. 237–242).
- Pecháč, M., Chovanec, M., & Farkaš, I. (2024). Self-supervised network distillation: An effective approach to exploration in sparse reward environments. *Neurocomputing*, 599, 128033.
- Pérez-García, F., Sharma, H., Bond-Taylor, B., Kenza, S., Valentina, I., Maximilian, B., Shruthi, C., Daniel, C., Schwaighofer, A., Lungren, M. P., Wetscherek, M., Codella, N., Hyland, S. L., Alvarez-Valle, J., & Oktay, O. (2024). RAD-DINO: Exploring scalable medical image encoders beyond text supervision. [arXiv:2401.10815](https://arxiv.org/abs/2401.10815)
- Perozzi, B., Al-Rfou, R., & Skiena, S. (2014). Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on knowledge discovery and data mining* (pp. 701–710).
- Pham, H., Dai, Z., Xie, Q., & Le, Q. V. (2021). Meta pseudo labels. In *2021 IEEE/CVF conference on computer vision and pattern recognition (CVPR)* (pp. 11552–11563). <https://doi.org/10.1109/CVPR46437.2021.01139>
- Pham, T., Zhang, C., Niu, Axi, Zhang, K., & Yoo, C. D. (2022). On the pros and cons of momentum encoder in self-supervised visual representation learning.
- Poole, B., Ozair, S., Van D. O., Aaron, A. A., & Tucker, G. (2019). On variational bounds of mutual information. In *International conference on machine learning* (pp. 5171–5180). PMLR.
- Punn, N. S., & Agarwal, S. (2022). BT-Unet: A self-supervised learning framework for biomedical image segmentation using barlow twins with u-net models. *Machine Learning*, 111(12), 4585–4600.
- Ravanbakhsh, E., Liang, Y., Ramanujam, J., & Li, X. (2023). Deep video representation learning: A survey. *Multimedia Tools and Applications*, 1–31.
- Ren, S., He, K., Girshick, R., & Sun, J. (2015). Faster R-CNN: Towards real-time object detection with region proposal networks. *Advances in Neural Information Processing Systems*, 28.
- Rifai, S., Vincent, P., Muller, X., Glorot, X., & Bengio, Y. (2011). Contractive auto-encoders: Explicit invariance during feature extraction. In *Proceedings of the 28th international conference on machine learning, ICML'11* (pp. 833–840). Omnipress.
- Russakovsky, O., Deng, J., Hao, S., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., et al. (2015). Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115, 211–252.
- Schölkopf, B., Janzing, D., Peters, J., Scouritsa, E., Zhang, K., & Mooij, J. (2012). On causal and anti-causal learning. [arXiv:1206.6471](https://arxiv.org/abs/1206.6471)
- Sheng, H., Ma, L., Samson, J.-F., & Liu, D. (2024). BarlowTwins-CXR: enhancing chest x-ray abnormality localization in heterogeneous data with cross-domain self-supervised learning. *BMC Medical Informatics and Decision Making*, 24(1), 126.
- Shorten, C., & Khoshgoftaar, T. M. (2019). A survey on image data augmentation for deep learning. *Journal of Big Data*, 6(1), 1–48.
- Shwartz-Ziv, R., & LeCun, Y. (2024). To compress or not to compress-self-supervised learning and information theory: A review. *Entropy*, 26(3), 252.
- Song, J., & Ermon, S. (2020). Multi-label contrastive predictive coding. *Advances in Neural Information Processing Systems*, 33, 8161–8173.

- Tao, C., Wang, H., Zhu, X., Dong, J. S., Shiji, H., Gao, & Dai, J. (2022). Exploring the equivalence of siamese self-supervised learning via a unified gradient framework. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 14431–14440).
- Tian, Y., Krishnan, D., & Isola, P. (2020a). Contrastive multiview coding. In *European conference on computer vision* (pp. 776–794). Springer.
- Tian, Y., Yu, L., Chen, X., & Ganguli, S. (2020b). Understanding self-supervised learning with dual deep networks. [arXiv:2010.00578](https://arxiv.org/abs/2010.00578)
- van den Oord, A., Kalchbrenner, N., Espeholt, L., Vinyals, O., Graves, A., et al. (2016). Conditional image generation with pixelCNN decoders. *Advances in Neural Information Processing Systems*, 29.
- van den Oord, A., Li, Y., & Vinyals, O. (2018). Representation learning with contrastive predictive coding. [arXiv:1807.03748](https://arxiv.org/abs/1807.03748)
- Vaswani, A., Ramachandran, P., Srinivas, A., Parmar, N., Hechtman, B., & Shlens, J. (2021). Scaling local self-attention for parameter efficient visual backbones. [arXiv:2103.12731](https://arxiv.org/abs/2103.12731). <https://doi.org/10.48550/arXiv.2103.12731>
- Vincent, P., Larochelle, H., Lajoie, I., Bengio, Y., Manzagol, P.-A., & Bottou, L. (2010). Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of Machine Learning Research*, 11(12).
- Wagner, S. S., & Harmeling, S. (2024). Just cluster it: An approach for exploration in high-dimensions using clustering and pre-trained representations. In Salakhutdinov, R., Kolter, Z., Heller, K., Weller, A., Oliver, N., Scarlett, J., Berkenkamp, F. (Eds.), *Proceedings of the 41st international conference on machine learning*, volume 235 of *Proceedings of machine learning research* (pp. 49788–49807). PMLR, 21–27.
- Wu, Z., Xiong, Y., Yu, S. X., & Lin, D. (2018). Unsupervised feature learning via non-parametric instance discrimination. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 3733–3742).
- Xiao, B., & Lu, C. (2023). Semi-supervised medical image classification combined with unsupervised deep clustering. *Applied Sciences*, 13(9).
- Yang, S., Xiao, W., Zhang, M., Guo, S., Zhao, J., & Shen, F. (2022). Image data augmentation for deep learning: A survey. [arXiv:2204.08610](https://arxiv.org/abs/2204.08610)
- You, Y., Gitman, I., & Ginsburg, B. (2017). Large batch training of convolutional networks. [arXiv:1708.03888](https://arxiv.org/abs/1708.03888).
- Zbontar, J., Jing, L., Misra, I., LeCun, Y., & Deny, S. (2021). Barlow twins: Self-supervised learning via redundancy reduction. In *International conference on machine learning* (pp. 12310–12320). PMLR.
- Zeng, D., Wu, Y., Hu, X., Xu, X., Hu, J., & Shi, Y. (2023). Additional positive enables better representation learning for medical images. In Greenspan, H., Madabhushi, A., Mousavi, P., Salcudean, S., Duncan, J., Syeda-Mahmood, T., & Taylor, R. (Eds.), *Medical image computing and computer assisted intervention—MICCAI 2023* (pp. 119–129). Springer. ISBN 978-3-031-43907-0
- Zhang, C., Zhang, K., Zhang, C., Pham, T. X., Yoo, C. D., & Kweon, I. S. (2022). How does SimSiam avoid collapse without negative samples? A unified understanding with self-supervised contrastive learning. In *International conference on learning representations*.
- Zhou, B., Lapedriza, A., Xiao, J., Torralba, A., & Oliva, A. (2014). Learning deep features for scene recognition using places database. *Advances in Neural Information Processing Systems*, 27.
- Zhou, L., Liu, H., Bae, J., He, J., Samaras, D., & Prasanna, P. (2023). Self pre-training with masked autoencoders for medical image classification and segmentation. In *2023 IEEE 20th international symposium on biomedical imaging (ISBI)*, (pp. 1–6).

Authors and Affiliations

Tobias Uelwer¹ · Jan Robine¹ · Stefan Sylvius Wagner² · Marc Höftmann¹ · Eric Upschulte^{3,4} · Sebastian Konietzny¹ · Maike Behrendt² · Stefan Harmeling¹

✉ Jan Robine
jan.robine@tu-dortmund.de

Tobias Uelwer
tobias.uelwer@tu-dortmund.de

Stefan Sylvius Wagner
stefan.wagner@hhu.de

Marc Höftmann
marc.hoeftmann@tu-dortmund.de

Eric Upschulte
e.upschulte@fz-juelich.de

Sebastian Konietzny
sebastian.konietzny@tu-dortmund.de

Maike Behrendt
maike.behrendt@hhu.de

Stefan Harmeling
stefan.harmeling@tu-dortmund.de

¹ Department of Computer Science, Technical University of Dortmund, Dortmund, Germany

² Department of Computer Science, Heinrich Heine University Düsseldorf, Düsseldorf, Germany

³ Institute of Neuroscience and Medicine (INM-1), Research Center Jülich, Jülich, Germany

⁴ Helmholtz AI, Research Center Jülich, Jülich, Germany