Computers & Graphics 115 (2023) 13–24

Contents lists available at ScienceDirect

Computers & Graphics

journal homepage: www.elsevier.com/locate/cag

Special Section on 3DOR 2023

Self-supervised learning for robust object retrieval without human annotations

Jarne Van den Herrewegen[a,b,*], Tom Tourwé[a], Francis wyffels[b]

[a] Oqton AI, Belgium
[b] AI and Robotics Lab, IDLab-AIRO, Ghent University - imec, Belgium

ARTICLE INFO

Article history:
Received 11 April 2023
Received in revised form 5 June 2023
Accepted 24 June 2023
Available online 28 June 2023

Keywords:
Object retrieval
Self-supervised learning
Cross-domain

ABSTRACT

This paper explores the potential of self-supervised learning as an alternative to supervised learning in the context of geometry-based 3D object retrieval. With the ongoing digitalization of many industries, an exponentially increasing number of 3D objects are processed by retrieval systems. In order to support new shapes, modern deep learning-based retrieval systems require retraining. The dominant paradigm for optimizing neural networks in this field is supervised classification training. Supervised learning requires time-consuming and expensive data annotation. Moreover, training neural networks for classification introduces a bias towards the classes in the training data, which is undesirable for retrieval systems encountering unseen object types in the wild.

Through extensive experiments, we make a direct comparison between supervised and self-supervised learning on four datasets from three different domains (household, manufacturing and medical). For object classes seen during training, self-supervised and supervised learning are competitive. For unseen classes, self-supervised learning outperforms supervised learning in many cases. We conclude that self-supervised learning provides a powerful tool for circumventing labeling costs and providing more robust retrieval systems.

1. Introduction

Object retrieval has been an essential instrument in various fields for decades, including game development, AR/VR [1,2] applications, medicine [3] and Computer-Aided Design/ Manufacturing (CAD/CAM) [4]. Object retrieval allows designers, artists and engineers to start their work from existing objects or use them as reference parts, drastically shortening the amount of time spent on a task [4,5]. With the ongoing digitalization in many industries, the need for 3D design and 3D processing keeps growing. Retrieval systems must support larger amounts of objects and handle an ever-increasing variety of shapes [6–9].

Modern retrieval systems represent 3D objects with feature vectors [9], where similar objects are expected to have similar vectors. In the past, feature vectors were the result of hand-crafted feature extraction algorithms [10]. Following the breakthrough of deep learning in the last decade [11], the current state-of-the-art feature extractors for 3D objects are neural networks, offering superior performance in most cases and requiring less development effort than feature engineering approaches.

The dominant paradigm to train a neural network for 3D object retrieval is to train in a supervised manner on a classification task [12–14]. The final layers in the network (the *head*) are removed after training, and the leftover *backbone* layers are kept as the feature extractor, see Fig. 1.

Two important challenges present themselves for supervised classification training. (1) Supervised learning requires human-annotated data. To obtain a deep learning feature extractor or to update a model for supporting new object categories, sufficient data has to be annotated. The process of annotating is typically time-consuming and costly, and is often a blocker during product development when insufficient domain expertise is available for data annotation. (2) Teaching a neural network to classify objects introduces a bias towards the classes seen during training. When extracting feature vectors for objects from unseen classes, the extracted vectors are less qualitative, ultimately resulting in worse retrieval results [14]. Supporting unseen object types is a crucial property of retrieval systems in the wild, as the number of 3D objects has been growing exponentially for decades together with the digitalization in many industries [6,7].

To overcome the challenges related to supervised classification training, self-supervised learning could provide an alternative training procedure for retrieval models. Self-supervised learning is a paradigm where neural networks are trained in a similar

* Correspondence to: Sint-Jacobsnieuwstraat 17, 9000 Gent, Belgium.
E-mail addresses: jarne.vandenherrewegen@oqton.com (J. Van den Herrewegen), tom.tourwe@oqton.com (T. Tourwé), francis.wyffels@ugent.be (F. wyffels).

https://doi.org/10.1016/j.cag.2023.06.029
0097-8493/© 2023 The Authors. Published by Elsevier Ltd. This is an open access article under the CC BY-NC-ND license (http://creativecommons.org/licenses/by-nc-nd/4.0/).

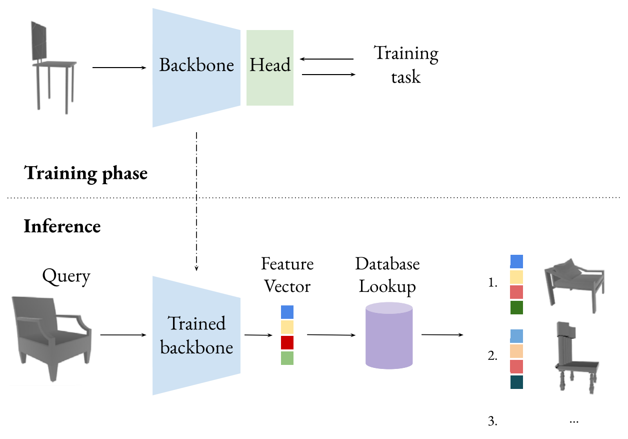


Fig. 1. Overview of a geometry-based 3D object retrieval system with a neural network as feature extractor. During the first phase, a neural network consisting of a backbone (general feature extraction) and a head (task prediction), is optimized for a training task. Next, the backbone is kept for inferring general feature vectors that act as descriptors for fast lookup in a vector database.

manner to supervised learning, with the difference that the training labels are automatically generated from the data [15]. As such, there is no need for human annotations in self-supervised learning and neither are neural networks explicitly biased towards specific classes. Recent advances in self-supervised learning have catalyzed numerous breakthroughs, significantly impacting many subfields in the broader artificial intelligence domain, starting with natural language processing [16,17] and spreading to image processing [18,19], video processing [20,21], robotics [22,23], protein folding [24,25] and speech processing [26,27].

Self-supervised learning for 3D object retrieval has been studied in the past, but only limited work has compared self-supervised learning to supervised learning in an extensive manner. Within this small subset of works, we argue that various aspects could be improved. First of all, self-supervised and supervised methods are compared with different feature extraction backbones [28,29], disallowing a direct comparison between the two training procedures. Second, the robustness and the capability to generalize to unseen object types is not evaluated [28–30], while retrieval systems in the real world need to handle unseen types. Moreover, only datasets with common objects (ModelNet40 [31] and Shapenet [32]) are considered [28,30], which is interesting for AR/VR applications in a household setting, but leaves many applications in medicine and CAD/CAM out of scope. Furthermore, experiments are only validated on datasets with objects aligned in a canonical orientation [28,30], while the correct alignment of objects in the wild is non-trivial. Notably, only view-based neural network architectures such as MVCNN [33] are considered [28], architectures for directly processing point clouds (e.g. PointNet++ [34] and DGCNN [35]) are left out. Finally, the proposed self-supervised methods are complex and extend the training loop significantly, e.g. by requiring clustering steps for negative sample mining [28] or by requiring training for two neural networks of different modality (images and point clouds) at the same time [30].

Our work addresses the aforementioned limitations in the state-of-the-art through a less complex self-supervised method and a larger, more realistic experimental setup. The important components of our work can be summarized as follows:

- Self-supervised and supervised methods are evaluated on the exact same feature extracting backbone;
- Four datasets from three different domains (household, manufacturing, medical) are used as training datasets, where some datasets have unaligned objects and others have aligned objects;
- The cross-domain performance of self-supervised and supervised similarity models is evaluated by performing retrieval on a different test dataset than the training set;
- The feature extracting backbone is a PointNet++ architecture, which processes point clouds in a direct manner;
- The self-supervised training procedure is based on the VICReg [36] method, a non-contrastive siamese network with a simple implementation that can run efficiently on 1 GPU.

We will show that self-supervised learning can be a valid, and often even a better-performing alternative over supervised learning in context of 3D object retrieval. The results can be summarized as follows:

- When evaluating and training on the same dataset, self-supervised learning can be a competitive replacement for supervised 3D object retrieval depending on the dataset and can thus circumvent labeling costs;
- Similarity models trained in a self-supervised setup can generalize better to datasets from other domains, providing better robustness;
- We show that non-contrastive implementations for siamese network setups such as VICReg are suitable for 3D deep learning, avoiding complex implementations.

In the remainder of this paper, related works on 3D object retrieval and self-supervised learning are described in Section 2, our application of VICReg to 3D deep learning is detailed in Section 3, the experiments are discussed in Section 4 and the conclusion with future work can be found in Section 5.

2. Background & Related work

In order to situate our work in the existing literature, an overview is presented on 3D deep learning & 3D object retrieval (Section 2.1), self-supervised learning (Section 2.2) and self-supervised object retrieval (Section 2.3).

2.1. 3D deep learning & 3D object retrieval

Historically, 3D object retrieval relied on hand-crafted features, based on spherical harmonics [37], 2D projection statistics [38], lightfields [39] or other methods. A more complete overview can be found in Bustos et al. [9], Iyer et al. [8], Tangelder et al. [40] and Li et al. [10]. In the past decade, the focus shifted towards *learning* the feature vectors with neural networks (NNs). The successes in image processing [11] with convolutional neural networks (CNNs) gave inspiration for 3D deep learning in two ways: 3D convolutional neural networks and view-based neural networks.

3D CNNs [31,41] are a natural extension of the pixel-based 2D convolutional layers to voxel-based 3D convolutional layers. Voxels are a relatively simple data structure compared to e.g. point clouds, meshes or projection-based representations. The most important downside for voxel-based methods is that the required computational resources scale cubically when increasing the resolution to smaller voxel sizes. Various architectures tried to overcome this downside, e.g. OctNet [42].

View-based neural networks [33] work by rendering a 3D object from multiple camera positions and processing the 2D projections as images with 2D convolutional neural networks. These architectures allow to directly make use of breakthrough architectures, e.g. AlexNet [11], VGGNet [43] and ResNet [44], and even allow to start from model weights pre-trained on large image datasets, e.g. ImageNet [45]. These methods have achieved state-of-the-art results in 3D deep learning since their inception and have been a staple in the 3D object retrieval literature, for example, with RotationNet [46] and GVCNN [12].

With PointNet++ [34], NN architectures applied to point clouds came on par with the state-of-the-art view-based methods. Point cloud-based architectures have the advantage of containing less parameters than view-based architectures and only require 1 pass through the NN backbone instead of 1 pass for each render in a view-based model. Other types of architectures include graph-based models such as DGCNN [35] and transformer-based models [47], e.g. Pointformer [48].

In geometry-based object retrieval, view-based architectures have traditionally outperformed other architectures [33,46] and this is no different in recent works [14,28]. However, view-based methods come with the downside of requiring more computation time [49] because these architectures have 10 to 100 times more parameters and requiring passing multiple views through the backbone. In this work, the PointNet++ architecture is the NN architecture of choice to allow for a faster iteration time.

2.2. Self-supervised learning

Self-supervised learning refers to methods where neural networks are trained with labels that are automatically generated from the data. The problem formulation for self-supervised learning looks similar to supervised learning schemes. In practice, self-supervised learning does not require expensive human annotations.

The advantages of self-supervised learning became clear in recent years thanks to breakthroughs in natural language processing [16,17]. The basis of these successes involved masked language modeling, next-sentence prediction and other text-related reconstruction methods which do not require human annotations.

While self-supervised learning has been around in the computer vision world since the early 1990s [50,51], the strong performance of pre-trained language models attracted more attention to self-supervised learning for computer vision. Important methods include siamese networks [52,53] and auto-encoding [54] approaches. As the focus of this work is only on feature vectors in latent space and does not require reconstructions in Euclidean space, experiments are performed with siamese networks.

In a siamese network setup [52], a neural network is trained to infer the same output vector for two similar data samples. Original applications include signature recognition [52] and face recognition [53], two cases where similar samples had to be retrieved from a database (as in 3D object retrieval).

The ability to control what a siamese network will learn to consider as similar, is the basis for self-supervised learning with siamese networks. It is possible to artificially generate similarity pairs, alleviating the need for human annotations. For example, similarity pairs can be created by generating two augmented versions of one sample and artificially labeling the two versions as similar [18,19,55–57].

The challenge for a siamese network setup is to prevent the model from predicting a constant vector that does not change depending on the input. In recent years, the most popular approach to solving this problem was *contrastive learning* [26,53,58,59]. This involves training on non-similar pairs as well and penalizing the siamese network for inferring similar vectors for the samples in a non-similar pair.

The current state-of-the-art siamese network methods try to avoid negative pairs however [36,60–62]. Contrastive learning methods work best when there are far more negative samples than positive samples in one batch [58], resulting in constraining VRAM requirements on GPUs or resulting in implementation complexity [59]. Important non-contrastive works include various forms of asymmetrical networks [60,61], regularization through auxiliary loss functions [36,63] and making smart use of batch normalization [62]. We will be using the VICReg [36] approach in this paper, which avoids the trivial siamese network solution by forcing the output vectors in a batch to take on a distribution with a regularizing auxiliary loss.

2.3. Self-supervised learning for 3D object retrieval

Diverse methods have been proposed for self-supervised 3D object retrieval. Pioneering works use auto-encoder setups in which neural networks are trained to reconstruct (a derivative of) the input. Leng et al. [64] apply convolutional auto-encoders, trained in a layer-wise manner, to 2D depth images. Zhu et al. [65] train auto-encoders in one phase and also consider 2D depth images as input. Furuya et al. [66] present a cross-modal auto-encoder, called the *Transcoder*, which trains encoders and decoders towards and from a shared latent space combining point clouds, voxels and 2D images. Luciano et al. [29] propose a feature extractor that takes in a vector of geodesic moments derived from a mesh. The feature extractor is trained through an auto-encoder setup in which the input vectors are reconstructed.

Other methods consider contrastive learning and siamese networks. Jing et al. [30] train a 2D–3D multi-modal siamese setup where an image feature extractor and a point cloud feature extractor learn to infer the same feature vector for a point cloud and a rendered image of the point cloud. The trained point cloud feature extractor serves as a retrieval model. Li et al. [28] train a siamese network with a view-based architecture only operating on 3D objects. The network is trained with a local loss based on contrasting views of each object and a global triplet loss based on contrasting positive and negative objects that were mined during a pre-training phase.

The experiments in this work will use siamese network methods for two reasons. First, the focus in modern object retrieval systems goes to feature vectors in latent space and less so to the Euclidean space where an auto-encoder setup performs reconstruction. Second, state-of-the-art non-contrastive siamese networks have low implementational complexity and VRAM requirements, allowing for fast iteration cycles and easy accessibility for the broader community.

3. Method

We describe our application of VICReg to neural networks for processing 3D point clouds. First, the general context of a siamese network and our data generation are described. Next, the regularization for stabilizing the training is explained. We end by detailing the model architectures that are used throughout our experiments. A high-level overview is depicted in Fig. 2.

3.1. Siamese network setup for 3D objects

This section describes the two key ideas behind self-supervised learning with siamese networks for 3D objects: the siamese network setup and the artificial generation of similarity pairs.

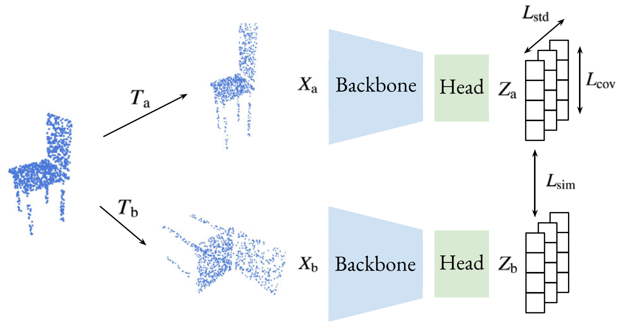


Fig. 2. Overview for our application of the VICReg [36] method in 3D deep learning. A siamese model is trained to infer the same batch of vectors Z_a and Z_b for augmented versions X_a and X_b of the same batch of 3D objects. The similarity loss L_{sim} is the Euclidean distance. The VICReg approach adds two auxiliary loss functions to the siamese model setup for stabilizing and for improving the model training. The first auxiliary loss function is the standard deviation loss L_{std} which brings the output vectors distribution per batch Z to the standard normal distribution. The standard deviation loss prevents the network from learning the unwanted trivial solution where the network always predicts the same vector, regardless of the input. The second auxiliary loss is the covariance loss L_{cov} which reduces the correlation between the dimensions in the output vectors. The covariance loss is in place to maximize the information yield of the output vectors, which has been shown to improve the feature vector quality [63,67].

3.1.1. *Siamese network fundamentals*

A siamese network setup aims to extract similar feature vectors for similar shapes. It consists of a neural network encoder $f_\theta(\mathbf{x})$ with parameters θ that learns a mapping of two samples in a pair $(\mathbf{x}_a, \mathbf{x}_b)$ onto the same vector in a high-dimensional vector space. More concretely, the encoder is trained to bring $\mathbf{z}_a = f_\theta(\mathbf{x}_a)$ and $\mathbf{z}_b = f_\theta(\mathbf{x}_b)$ close in vector space $\mathbb{V}^D$ by minimizing a similarity loss function L_{sim}. In our work, we use the mean-squared Euclidean distance between the unnormalized feature vectors $\mathbf{z}_a$ and $\mathbf{z}_b$:

$$L_{sim}(\mathbf{z}_a, \mathbf{z}_b) = \sum_{d=1}^{D} (z_a^d - z_b^d)^2 \tag{1}$$

with z_a^d the value for dimension d in vector $\mathbf{z}_a$. Siamese networks form the basis for obtaining similarity models throughout this work.

3.1.2. *Data generation*

The trick for training a siamese similarity model in a self-supervised manner lies in the acquisition process of the similarity pairs. To circumvent the need for human similarity annotations, it is possible to artificially generate similarity pairs by augmenting a sample twice. The two augmented versions can be considered similar (assuming the augmentations are not unreasonably strong) since they originate from the same sample.

During training, mini-batches of k point clouds are sampled from the dataset. For each of the k point clouds in the mini-batch, two augmented versions are generated and paired. The k resulting pairs are passed on to the siamese network for training.

Augmentations are generated through a sequence of geometric transformations. The following stochastic transformations are used in this work (Fig. 3), assuming that the point cloud is normalized to the unit sphere:

1. point cloud subsampling,
2. random 3D rotation,
3. global scaling with factor s, uniformly distributed between [0.8, 1.25],
4. stretching in x, y, z dimensions with respective factors s_x, s_y,
s_z, independently uniformly distributed in [0.8, 1.25],
5. point jittering by displacing each point with $\Delta_x, \Delta_y, \Delta_z$ sampled independently from a normal distribution $\mathcal{N}(0, 0.01)$, clipped in the interval $[-0.05, 0.05]$.

Point subsampling refers to the procedure of generating point clouds with a large number of samples before training and using subsampled versions during training. When generating two augmented versions of a point cloud, two different subsampled versions were given as input to the augmentation pipeline.

Note that in many cases, the rotation augmentation was not applied. In Section 4, we show that rotation augmentations can severely hurt the performance of the similarity model, in particular when working with aligned datasets.

3.2. *Improving siamese networks through regularization*

Below, two auxiliary loss functions are described which were proposed by the VICReg [36] work to stabilize and improve siamese network training without requiring negative samples.

3.2.1. *Avoiding collapse*

Training a siamese network setup with only a similarity loss leaves room for an unwanted trivial solution, i.e. the constant vector. It is possible for the network to predict the same constant vector for all possible inputs and still satisfy the loss function. For avoiding this collapse to the trivial solution, there is an additional loss function for regularizing the model.

The proposed auxiliary loss function works by penalizing the network for having a small standard deviation over the feature vectors in a mini-batch. Such an auxiliary loss pushes the network away from always predicting a constant vector as that would result in a standard deviation equal to 0. At the same time, the standard deviation should not be growing indefinitely as that would destabilize the training in another way. Therefore, the loss function is limited to a maximum value of 1. This is effectively the Hinge loss with threshold 1 applied to the variance within each dimension of the batch. The regularizing loss L_{std} takes a batch Z of N vectors $[\mathbf{z}_1, \ldots, \mathbf{z}_n]$ and returns the following result:

$$L_{std}(Z) = \frac{1}{ND} \sum_{i=1}^{N} \sum_{j=1}^{D} \max(0, 1 - S(z_i^j, \epsilon)) \tag{2}$$

where z_i^j is the value of dimension j for vector $\mathbf{z}_i$ and S is the standard deviation:

$$S(x, \epsilon) = \sqrt{\mathrm{Var}(x) + \epsilon} \tag{3}$$

with ϵ a small number to prevent numerical instabilities. From the perspective of generative modeling approaches, L_{std} forces the model output to take on a distribution and leaves no room for the constant solution.

An important caveat must be considered when applying this loss in conjugation with the similarity loss L_{sim}. The standard deviation loss L_{std} is designed to push the vectors in a mini-batch away from each, including the vectors from samples belonging to the same similarity pair. To prevent these losses from opposing each other, the L_{std} loss is not applied to the full mini-batch at once. Still, it is separately applied to the two halves of the mini-batch with each half containing one of the two members of each similarity pair.

3.2.2. *Maximizing information*

A second regularizing loss is the *covariance* loss L_{cov}, which reduces the correlation between the dimensions in the feature vector. The goal of this regularization is to maximize the information yield of each dimension and was introduced by the Barlow Twins [62] method. In other words, the network is forced

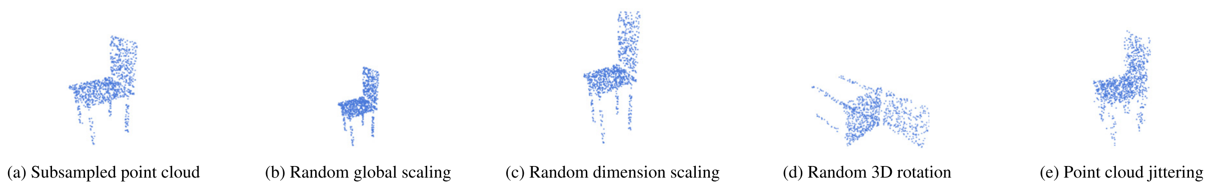

(a) Subsampled point cloud
(b) Random global scaling
(c) Random dimension scaling
(d) Random 3D rotation
(e) Point cloud jittering

Fig. 3. Examples of the transformations employed in the augmentation pipeline. To train siamese networks in a self-supervised manner, it is possible to artificially generate the required similarity pairs by creating augmented versions of 3D objects. This figure illustrates the five proposed augmentations separately.

to encode as much unique information in each dimension. For further intuition, we refer to [63,67] where the authors show that embeddings that are uniformly spread out over the hypersphere indicate strong representation learning and where the relationship to the *information bottleneck* [68,69] principle is investigated as well.

The covariance loss L_{cov} is derived from the off-diagonal elements in the covariance matrix C of the output batch Z with N vectors $[\mathbf{z}_1, \ldots, \mathbf{z}_n]$ with D dimensions:

$$C(Z) = \frac{1}{N-1}\sum_{i=1}^{n}(\mathbf{z}_i - \bar{\mathbf{z}})(\mathbf{z}_i - \bar{\mathbf{z}})^T, \text{ where } \bar{\mathbf{z}} = \frac{1}{N}\sum_{i=1}^{n}\mathbf{z}_i. \tag{4}$$

$$L_{\text{cov}}(Z) = \frac{1}{D}\sum_{i\neq j}[C(Z)]^2_{i,j}. \tag{5}$$

The two regularizing loss functions are combined with the similarity loss in a final weighted loss L_{VICReg}:

$$L_{\text{VICReg}} = \alpha * L_{\text{sim}} + \beta * L_{\text{std}} + L_{\text{cov}}, \tag{6}$$

where the same values were used as in the original work, $\alpha = 25$ and $\beta = 25$.

3.3. Neural network architectures

In this section, the structure of the models used throughout this work is described. Generally, each model exists out of two modules. The first comprises the initial layers of the neural network which extract local features and combine the local information into a global vector. This module is referred to as the *feature extractor* or the *backbone*. The second module consists of the last layers in the network, which process the global vector from the backbone module and perform the task for which the network is being trained. This module is called the *head* of the neural network.

After training the neural network, we only keep the backbone module for extracting global feature vectors, as shown in Fig. 1. These global feature vectors are used as descriptors for the 3D objects during object retrieval tasks.

3.3.1. PointNet++

The PointNet++ [34] architecture is designed for processing 3D point clouds in a hierarchical structure. The structure is made up of a sequence of *set abstraction* layers that extract local features with small PointNets [70] in increasing spherical neighborhoods. The small PointNet modules have the key property of being *permutation invariant*, i.e. no matter the order in the set of points, the resulting point cloud feature vector will be the same. The merit of the PointNet++ architecture is the hierarchical structuring which introduces weight sharing between different local neighborhoods. PointNet acts as the method which enables the direct processing of point clouds with neural networks. PointNet++ is the method that achieves state-of-the-art performance.

3.3.2. VICReg model

A VICReg model consists of a neural network backbone and a multilayer perceptron (MLP) head, the *expander*. The backbone network can be any feature extractor for 3D objects. We use the PointNet++ architecture as backbone where the dimension of the global vector is 1024. The expander network is an MLP head with two hidden layers (consisting of a fully-connected layer, batch normalization, ReLU activation) and one output layer (a fully-connected layer with no bias term), all of dimension 1024.

Regarding the siamese network setup, we use the same architecture $f_\theta(\mathbf{x})$ and the same weights θ for each of the two forward passes. There exist other works that use the same architecture but not the same weights [62]. Some works even use different architectures in the two siamese branches, e.g. in a student–teacher setup [71] or in a multi-modal setup [30].

3.3.3. Classification model

The classification models used in this work, have a PointNet++ backbone and an MLP head. We use the MLP head presented by Qi et al. [34]. The head consists of 2 hidden layers (consisting of a fully-connected layer, batch normalization, ReLU, 50% dropout) with dimensions 512 and 256, followed by an output layer where the dimension is equal to the number of classes considered in the dataset.

4. Experiments & Results

We compare self-supervised learning and supervised learning for 3D object retrieval in two experiments. The first experiment considers the default context in machine learning where the training set and the test set have the same distribution. We show that self-supervised learning (VICReg) and supervised learning (classification) show similar performance in this setting.

In the second experiment, evaluation happens on test sets containing object categories that were not seen during training, i.e. the training set and the test set have a different distribution. Our results show that self-supervised learning outperforms supervised learning in most cases.

In the following subsections, we discuss our evaluation pipeline, the importance of each dataset, the implementation details and the experiment results.

4.1. Evaluation pipeline

In each of our experiments, we aim to evaluate training methods for neural network backbones that encode the 3D objects onto latent feature vectors. As depicted in Fig. 1, the neural networks are first trained for a specific training task, which will be either classification (supervised learning) or VicReg (self-supervised learning). After the training phase, the neural network head, specific to the training task, is removed and only the task-agnostic backbone remains. All backbones for object retrieval in this work have the same architecture and the same encoding dimension (1024) regardless of the training task. This is important to allow a fair and direct comparison between the supervised and the self-supervised training methods.

Table 1
Dataset overview showing the domain of origin, the alignment and the dataset size.

Dataset	Domain	Alignment	# Classes	# Samples
ModelNet40	Household	Aligned	40	12.311
ShapeNet normal	Household	Aligned	55	51.162
ShapeNet perturbed	Household	Unaligned	55	51.162
Mechanical Components Benchmark	Manufacturing	Unaligned	68	58.696
Proprietary dataset	Medical/dental	Unaligned	38	12.038

After a backbone is trained on the training samples from a dataset, it can be evaluated on the same dataset or other datasets. The procedure for evaluating a backbone on a dataset goes as follows. First, the backbone infers latent feature vectors for all train and test samples in the evaluation dataset. Then, the quality of the feature vectors is evaluated through nearest neighbor classification and through similarity ranking.

The nearest neighbor classification works by finding the training sample with the closest feature vector (according to the cosine distance in latent space) to each test sample. The classification label of the training sample is transferred to the test sample as the prediction. To measure the nearest neighbor classification performance, we measure the accuracy over all test samples (referred to as NN in the tables) and the F1 score averaged over all classes (referred to as F1 macro in the tables). The F1 score is the harmonic mean of the recall and the precision for a class.

The similarity ranking performance of trained backbones is evaluated through the Normalized Discounted Cumulative Gain metric (referred to as NDCG in the tables). For each test sample, the following procedure is executed. First, the N closest training objects in latent space, according to cosine distance, are retrieved. The retrieved objects are ranked according to their distance to the test sample in latent space. Next, the Discounted Cumulative Gain metric is calculated, which gives a basic score of 1 point to each training sample with the same class label as the test sample. However, each point is discounted logarithmically based on its position in the list. The lower a training sample of the correct class is ranked, the lower the score. The highest score is achieved when all training samples of the test sample class are on top of the ranked list. Where s_i is the basic score (0 for different class, 1 for correct class) at ranking position i, the DCG_N is obtained as follows:

$$\mathrm{DCG}_N = \sum_{i=1}^{N} \frac{s_i}{\log_2(i+1)}. \tag{7}$$

To normalize the DCG_N metric, its value is divided by the maximum possible DCG_N, as if all s_i in the list would be 1. The NDCG value reported in the tables is the average NDCG over all test samples in an evaluation dataset. For ModelNet40 and the proprietary dataset, we use $N = 200$ and for the other, larger datasets we use $N = 1000$.

4.2. Datasets

Multiple datasets are considered to compare the performance of self-supervised learning and supervised learning. The combination of datasets described below was specifically chosen because they originated from different domains, see Fig. 4 for a visualization and Table 1 for an overview. The combination allows to verify if self-supervised learning works in different domains and it allows to test the generalization performance of trained similarity models from domain to domain.

4.2.1. ModelNet40

ModelNet40 is a selection of 40 classes out of the larger ModelNet [31] collection. The ModelNet40 contains 12.311 objects, split into a training set of 9.843 samples and a test set of 2.468 samples. The samples contain household objects and common shapes like chairs, shelves, plant pots, cars, airplanes, ...An important detail is that these objects have a canonical orientation and are aligned in some relation to the gravitational direction. This has implications regarding the use of rotation augmentation during training.

4.2.2. ShapeNetCore

ShapeNetCore is a subset of 55 classes from the ShapeNet [32] database. The selection contains 51.162 3D objects with a train/val /test split of 35.764/5.133/10.265 objects respectively. In our experiments, we train on the combined train and validation set of 40.897 samples. This dataset contains similar categories as the ModelNet40 dataset. There are two versions of the dataset available, of which the first one contains objects in an aligned, canonical pose, to which we will refer as ShapeNet normal. The second version contains the same objects as ShapeNet normal, but rotated. The rotated version is referred to as ShapeNet perturbed.

4.2.3. Mechanical components benchmark

The Mechanical Components Benchmark [72] dataset contains 58.696 parts from the manufacturing world, which is different from ModelNet40 or ShapeNetCore. The dataset contains 68 classes which were purposefully chosen to be confusing. The work by Kim et al. [72] specifies an A version (all samples) and a B version (containing only 25 of 68 classes), we use the A version. We follow the official 80%/20% train/test split.

4.2.4. Proprietary data

To further diversify the domains considered in this paper, we also added the results from a proprietary dataset. There are 38 classes present in the proprietary dataset, of which most are medical/dental-related and a few are jewelry/industrial-related. The dataset contains 2.560 test samples and 9.478 training samples.

4.3. Implementation details

Before training, large point clouds of 16.000 points are randomly sampled with the python library `trimesh` [73,74]. During training, the augmentation pipeline randomly subsamples the point clouds to of 2.048 points. All models receive the points and the normals. Point clouds are normalized to the unit sphere. The deep learning framework for the implementation is Pytorch [75] along with the Pytorch Lightning [76]. The VICReg implementation is derived from the official open-source repository by `facebookresearch` [77]. The PointNet++ code is a combination of an implementation by Erik Wijmans [78] and `torch points kernels` [79]. All models were trained on one Nvidia Tesla V100 GPU. The Adam optimizer was used for all models with learning rate 3.10^{-4}. The weighted loss coefficients α and β are kept at 25 for all experiments. All classification models are trained for 250 epochs with batch size 64. The siamese networks were trained with batch size 128 for either 300 epochs (ShapeNet and Mechanical Components) or either 900 epochs (ModelNet40 and proprietary dataset).

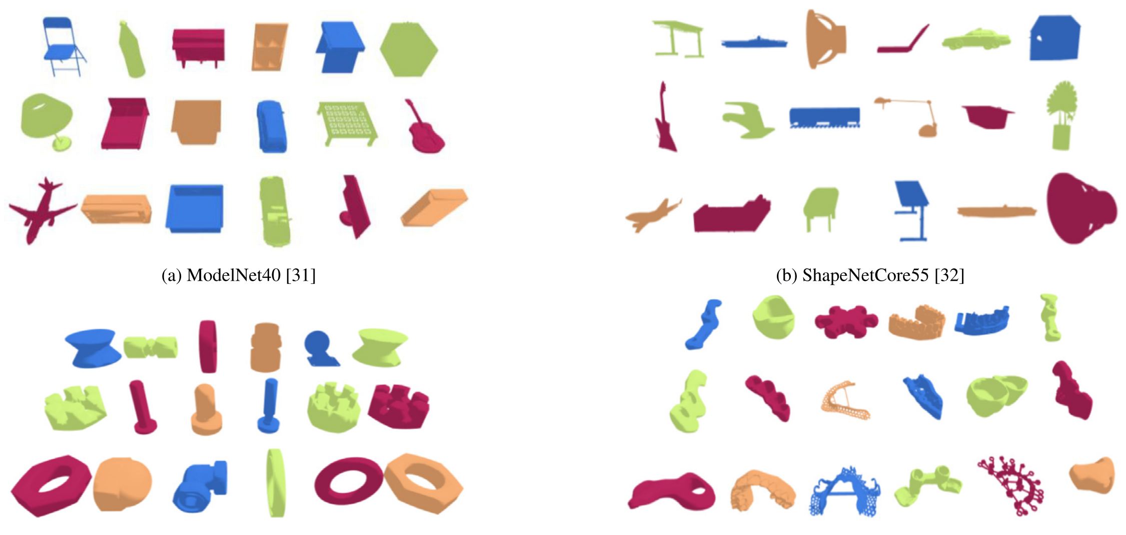

(a) ModelNet40 [31]
(b) ShapeNetCore55 [32]
(c) Mechanical Components Benchmark [72]
(d) Proprietary dataset

Fig. 4. Examples from each of the four datasets.

Table 2
Supervised (classification) and self-supervised (VICReg) training comparison on various datasets. After training a model, feature vectors are inferred with the backbone for all objects in the train and test set. Nearest neighbor classification and similarity ranking are performed to evaluate the backbones. In the table, NN% is the classification accuracy over all objects in the test set (classified by the label of their closest neighbor in the training set), F1% is the F1 score averaged over all classes and NDCG is the Normalized Discounted Cumulative Gain over all test samples.

Dataset + Training method	NN %	F1%	NDCG %
ModelNet40			
VICReg	84.8	80.9	38.9
Classification	**86.0**	**82.2**	**42.8**
ShapeNet normal			
VICReg	**83.1**	**68.2**	**66.0**
Classification	82.6	68.0	65.9
ShapeNet perturbed			
VICReg	**75.8**	**56.6**	**56.9**
Classification	71.7	53.6	55.4
Mechanical Components			
VICReg	95.2	**88.9**	**65.0**
Classification	**95.4**	88.8	64.8
Proprietary dataset			
VICReg	60.4	52.3	37.9
Classification	**81.3**	**72.8**	**48.4**

4.4. Experiment I: Supervised versus self-supervised learning

In this experiment, we make an apples-to-apples comparison between self-supervised learning and supervised learning trained and evaluated on the same data distribution, with the exact same backbone.

For each dataset, a classification model and a VICReg model are trained on the training split and evaluated on the test split belonging to the same dataset. Nearest neighbor classification and similarity ranking are performed as described in Section 4.1. The results are reported in Table 2. For datasets with objects in a canonical orientation, no rotation augmentation was used. The classification models are trained with the same augmentations as the VICReg models, except for the proprietary dataset where the classification model was trained without rotation augmentation and the VICReg model was trained with rotation augmentation.

The two training methods are competitive with each other. Which method works best depends on the dataset contents. For the two ShapeNet versions, VICReg training performs better. In the case of the Mechanical Components Benchmark, the two methods have a very similar performance. This is an impressive result for self-supervised learning as no human-annotated data is required at all for training.

For ModelNet40 and the proprietary dataset, supervised classification training performs better. We argue that these datasets might not be large enough to be effectively trained with VICReg. In particular, for the proprietary dataset, the performance is worse. An explanation could be that there is not enough variation in the dataset since 15 out of 38 classes are tooth-like or sequences of teeth, but more experiments would be required to confirm these claims.

4.5. Experiment II: Generalization over domains

In practice, retrieval systems come across object types that did not appear in the training set. We set up an experiment to compare the ability of self-supervised models and supervised models to generalize out of distribution by training on one dataset and testing on other datasets.

The training and evaluation procedure is the same as in Section 4.4. The results are reported in Tables 3–5, which also includes the results of experiment I for completeness.

In total there are 20 training configurations (rows) in these tables, showing the training dataset, the training method and the presence of rotation augmentation. 10 rows are supervised and 10 are self-supervised. Each supervised configuration can be compared to exactly one other self-supervised configuration with the same training dataset and the same presence for rotation augmentation. Given that each configuration is evaluated on 4 out-of-distribution datasets (apart from 1 other in-distribution dataset), there are 40 points of comparison between supervised and self-supervised learning.

Considering the NN accuracy results in Table 3, the self-supervised method outperforms the supervised method in 30 out of 40 cases. Leaving out the 8 cases trained on the proprietary dataset, self-supervised learning outperforms supervised learning in 29 out of 32 cases, which is impressive. For the F1 score in Table 4 and the NDCG metric in Table 5, we find similar strong results. We argue that this happens because supervised classification training introduces a stronger bias towards the classes seen during training than self-supervised learning. The self-supervised

Table 3
Self-supervised and supervised training with five datasets: ModelNet40 (M40), ShapeNet normal (SN normal), ShapeNet perturbed (SN perturbed), Mechanical Components Benchmark (MechComp) and the proprietary dataset (Prop). After training a model, feature vectors are inferred with the backbone for all objects in the train and test set. Nearest neighbor classification is performed to evaluate the backbones. In the table, the nearest neighbor classification accuracy over all objects in the test set is reported.

Training dataset	Training method	M40	SN normal	SN perturbed	MechComp	Prop
ModelNet40	VICReg					
	Not rotation augmented	84.8	**81.4**	45.3	**94.5**	58.8
	Rotation augmented	82.3	79.5	42.0	94.3	57.3
	Classification					
	Not rotation augmented	**86.0**	79.0	41.6	93.3	55.6
	Rotation augmented	81.5	77.6	**67.1**	93.2	**70.1**
ShapeNet normal	VICReg					
	Not rotation augmented	**86.6**	**83.1**	51.7	**94.3**	65.0
	Rotation augmented	81.7	76.9	**74.9**	**94.3**	**70.6**
	Classification					
	Not rotation augmented	82.0	82.6	43.6	93.1	59.6
	Rotation augmented	79.1	77.5	69.9	92.8	68.0
ShapeNet perturbed	VICReg					
	Not rotation augmented	**85.0**	**80.9**	38.3	92.2	60.4
	Rotation augmented	80.4	76.8	**75.8**	**93.9**	**71.7**
	Classification					
	Not rotation augmented	80.6	79.7	58.3	92.7	57.6
	Rotation augmented	77.1	75.6	71.7	92.9	70.2
MechComp	VICReg					
	Not rotation augmented	**82.1**	**80.4**	47.4	95.0	63.7
	Rotation augmented	70.0	69.6	**66.5**	95.2	**72.5**
	Classification					
	Not rotation augmented	76.9	76.4	40.4	**95.5**	58.0
	Rotation augmented	69.7	67.6	64.5	95.4	71.6
Proprietary dataset	VICReg					
	Not rotation augmented	73.7	75.5	40.9	92.4	59.1
	Rotation augmented	65.2	67.1	40.5	91.0	60.4
	Classification					
	Not rotation augmented	**78.1**	**77.6**	40.1	94.2	**81.3**
	Rotation augmented	70.9	69.2	**65.3**	**94.6**	74.5

method does not focus on specific classes as it is trained to infer similar feature vectors for similar geometries, which appears to be a better proxy task for training robust similarity models.

The cases where supervised outperforms self-supervised training, occur when training on the proprietary dataset and for ModelNet40. We suspect that the same reasons as in Section 4.4 are responsible for this, i.e. the dataset is too small or contains insufficient variation between the classes. In Fig. 5, we give a glimpse of the proprietary dataset and show some retrieval examples with the best performing model for the proprietary dataset (classification trained on proprietary dataset without rotation augmentation 81.3% NN) and the worst performing VICReg model (trained on ModelNet40 without rotation augmentation). Fig. 5(a) clearly shows how certain dental objects are incorrectly retrieved as similar objects. Given the strong performance of self-supervised learning when trained on other datasets, we believe that the few underwhelming results should be attributed primarily to the training data and not to the training method.

Next, we would like to point out the impact of rotation augmentation during training as this had an important influence on the results. When an evaluation dataset contains aligned objects and the training method used rotation augmentations, the results were negatively influenced and vice-versa. The impact of using rotation augmentations can go up to an absolute difference of 25+% (see results for ShapeNet Perturbed) and is often much larger than the impact of the training method.

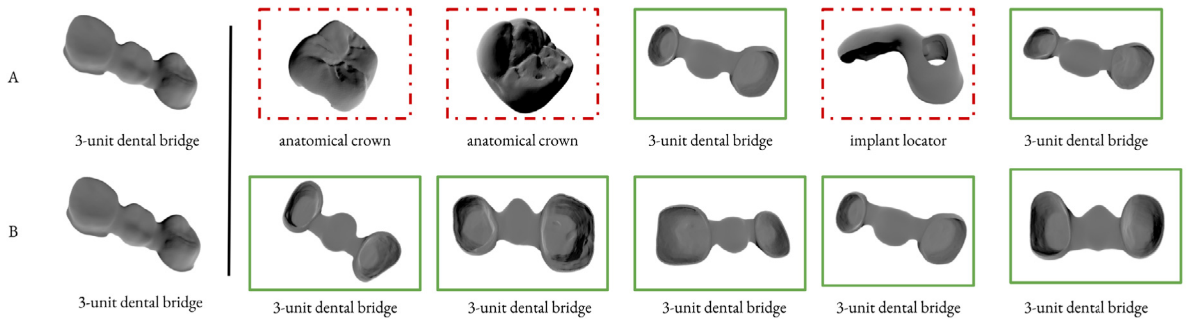
Again, there is an important difference: the self-supervised method does not require any human annotations at all. With sufficient care for the training data, self-supervised learning can be competitive with supervised learning and even has an edge for generalizing out of distribution.

5. Conclusion

Modern object retrieval systems represent 3D objects by feature vectors extracted by neural networks. Due to the ongoing digitalization of many industries, the number of objects and the variety of object types keeps growing exponentially. Supporting new object types is a challenge for neural networks and requires retraining, which traditionally happens in a supervised manner through classification. This procedure brings an annotation cost and a bias towards classes seen during training, hurting the generalization capabilities.

In this paper, we explored the potential of self-supervised 3D object retrieval. With a simple, non-contrastive siamese network setup, we made a fair and direct comparison between self-supervised and supervised learning. In many cases, self-supervised learning outperformed supervised learning, with a few exceptions for models trained on smaller datasets. This finding is validated in experiments over four datasets from three different domains. Moreover, we show that the alignment of objects to a canonical pose plays an influential role in the transferability of a neural network from one dataset to another. If a target dataset contains unaligned objects and the training dataset has aligned objects, the neural network requires rotation augmentation. When the training data is curated in the right manner, self-supervised learning outperforms supervised learning for 3D object retrieval. We would like to highlight the data-related aspect of our work, as it might be a fruitful research direction for future work.

Considering that self-supervised learning alleviates large-scale labeling requirements, the presented results can have far-reaching consequences for practical applications.



(a) Failure case: dental bridge with 3 units. Only 2 out of the top 5 retrieved examples belong to the same class for model A, while model B achieves 5 out of 5.

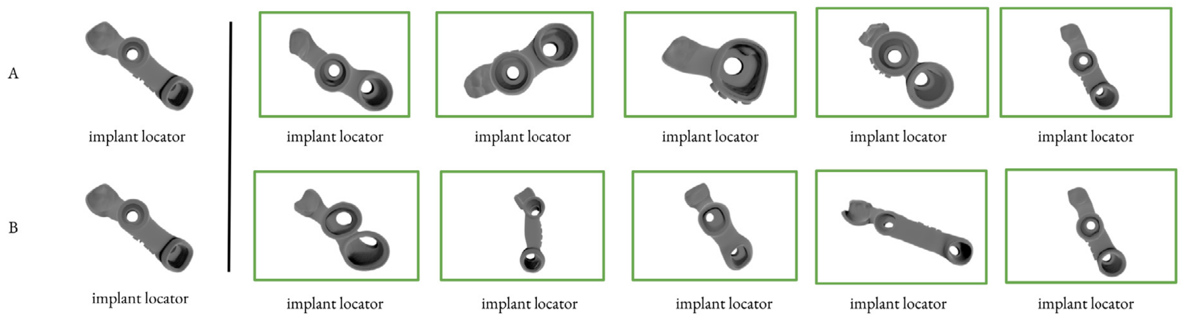


Fig. 5. Retrieval examples comparing model A, the worst performing VICReg model for the proprietary dataset trained on ModelNet40 with rotation augmentation (57.3% NN), and model B, the best performing classification model for the proprietary dataset trained on the proprietary dataset without rotation augmentation (81.3% NN. The most similar samples to the query sample are on the left.

Table 4
Self-supervised and supervised training with five datasets: ModelNet40 (M40), ShapeNet normal (SN normal), ShapeNet perturbed (SN perturbed), Mechanical Components Benchmark (MechComp) and the proprietary dataset (Prop). After training a model, feature vectors are inferred with the backbone for all objects in the train and test set. Nearest neighbor classification is performed to evaluate the backbones. In the table, the F1-score over all classes in the test set is reported.

Training dataset	Training method	M40	SN normal	SN perturbed	MechComp	Prop
ModelNet40	VICReg					
	Not rotation augmented	80.9	**65.8**	22.8	**88.2**	55.2
	Rotation augmented	77.4	63.6	21.1	86.8	53.1
	Classification					
	Not rotation augmented	**82.2**	62.1	20.5	86.1	52.5
	Rotation augmented	77.2	58.9	**46.9**	86.4	**63.1**
ShapeNet normal	VICReg					
	Not rotation augmented	**82.5**	**68.2**	26.5	**89.5**	59.1
	Rotation augmented	77.3	59.1	**56.6**	88.3	**63.2**
	Classification					
	Not rotation augmented	77.0	68.0	22.4	86.6	55.9
	Rotation augmented	74.5	59.3	49.9	85.3	61.3
ShapeNet perturbed	VICReg					
	Not rotation augmented	**80.9**	**63.6**	17.2	85.4	56.1
	Rotation augmented	75.5	59.1	**56.5**	**87.4**	**64.5**
	Classification					
	Not rotation augmented	76.3	**63.6**	34.5	86.3	54.1
	Rotation augmented	72.7	57.6	53.6	84.9	62.4
MechComp	VICReg					
	Not rotation augmented	**77.0**	**62.9**	23.5	88.1	59.7
	Rotation augmented	64.2	51.3	**48.5**	**88.9**	**65.3**
	Classification					
	Not rotation augmented	71.3	59.4	19.9	88.4	55.1
	Rotation augmented	64.5	49.2	45.1	88.8	64.1
Proprietary dataset	VICReg					
	Not rotation augmented	65.7	56.9	18.8	84.3	52.8
	Rotation augmented	59.0	47.6	20.2	80.3	52.3
	Classification					
	Not rotation augmented	**73.6**	**61.4**	20.2	87.4	**72.8**
	Rotation augmented	65.9	50.3	**46.3**	**88.2**	67.0

Table 5
Self-supervised and supervised training with five datasets: ModelNet40 (M40), ShapeNet normal (SN normal), ShapeNet perturbed (SN perturbed), Mechanical Components Benchmark (MechComp) and the proprietary dataset (Prop). After training a model, feature vectors are inferred with the backbone for all objects in the train and test set. Nearest neighbor classification is performed to evaluate the backbones. In the table, the NDCG over all objects in the test set is reported.

Training dataset	Training method	M40	SN normal	SN perturbed	MechComp	Prop
ModelNet40	VICReg					
	Not rotation augmented	38.9	**63.6**	37.0	**57.4**	39.2
	Rotation augmented	39.3	60.2	36.3	56.7	39.4
	Classification					
	Not rotation augmented	**42.8**	60.2	36.7	53.7	38.2
	Rotation augmented	41.0	60.2	**54.6**	52.8	**43.1**
ShapeNet normal	VICReg					
	Not rotation augmented	**42.2**	**66.0**	36.2	**58.4**	37.8
	Rotation augmented	40.2	57.4	**56.4**	58.1	**43.8**
	Classification					
	Not rotation augmented	39.3	65.9	35.6	53.3	39.5
	Rotation augmented	39.2	58.2	54.9	53.5	42.5
ShapeNet perturbed	VICReg					
	Not rotation augmented	**42.0**	**64.2**	33.3	51.6	36.5
	Rotation augmented	39.9	56.6	**56.9**	**58.0**	**43.8**
	Classification					
	Not rotation augmented	39.8	61.2	46.6	52.3	40.3
	Rotation augmented	38.3	56.4	55.4	53.6	42.8
MechComp	VICReg					
	Not rotation augmented	**39.7**	**61.2**	36.1	60.6	38.4
	Rotation augmented	36.9	51.1	**50.5**	**65.0**	43.6
	Classification					
	Not rotation augmented	37.9	58.1	37.5	64.8	39.3
	Rotation augmented	37.4	51.7	**50.5**	64.8	**44.1**
Proprietary dataset	VICReg					
	Not rotation augmented	37.1	55.4	35.4	51.6	38.0
	Rotation augmented	35.0	52.8	39.7	49.8	37.9
	Classification					
	Not rotation augmented	**37.4**	**58.9**	35.5	59.0	**48.4**
	Rotation augmented	37.2	52.4	**51.1**	**60.6**	45.5

CRediT authorship contribution statement

Jarne Van den Herrewegen: Conceptualization, Investigation, Methodology, Software, Validation, Data curation, Writing – original draft, Funding acquisition. **Tom Tourwé:** Conceptualization, Writing – review & editing, Supervision, Funding acquisition. **Francis wyffels:** Conceptualization, Writing – review & editing, Supervision, Funding acquisition.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Jarne Van den Herrewegen and Tom Tourwé are affiliated with Oqton, a suborganization in the 3D Systems corporation.

Data availability

The data that has been used is confidential.

Declaration of Generative AI and AI-assisted technologies in the writing process

During the preparation of this work, the author(s) used the “text-davinci-003” GPT model by OpenAI in order to find synonyms and to improve individual sentences throughout the manuscript. After using this tool/service, the author(s) reviewed and edited the content as needed and take(s) full responsibility for the content of the publication.

Acknowledgments

We would like to thank Ruben Verhack and Miloš Nešić for proofreading the draft and for sharing their advice. This work was supported by the Flemish Institute for Innovation and Entrepreneurship (VLAIO), Belgium, grant number HBC.2022.0165.

References

[1] Schleußinger M. Information retrieval interfaces in virtual reality—A scoping review focused on current generation technology. Plos One 2021;16(2).
[2] Pascoal P, Ferreira A, Jorge J. Towards an immersive interface for 3D object retrieval. In: Proceedings of the 5th eurographics conference on 3D object retrieval. 2012, p. 51–4.
[3] Keim DA. Efficient geometry-based similarity search of 3D spatial databases. ACM SIGMOD Rec 1999;28(2):419–30.
[4] Hoffmann C, Shapiro V, Srinivasan V. Geometric interoperability via queries. Comput Aided Des 2014;46:148–59.
[5] Li Z, Liu M, Ramani K. Review of product information retrieval: representation and indexing. In: International design engineering technical conferences and computers and information in engineering conference, Vol. 46970. 2004, p. 971–9.
[6] Williams J, Scott S, Hindanov T, Roedig C. Data management challenges for internet-scale 3D search engines. 2022, arXiv preprint arXiv:2209.03913.
[7] Flath CM, Friesike S, Wirth M, Thiesse F. Copy, transform, combine: exploring the remix as a form of innovation. J Inf Technol 2017;32(4):306–25.
[8] Iyer N, Jayanti S, Lou K, Kalyanaraman Y, Ramani K. Three-dimensional shape searching: state-of-the-art review and future trends. Comput Aided Des 2005;37(5):509–30.
[9] Bustos B, Keim DA, Saupe D, Schreck T, Vranić DV. Feature-based similarity search in 3D object databases. ACM Comput Surv 2005;37(4):345–87.
[10] Li B, Lu Y, Li C, Godil A, Schreck T, Aono M, Burtscher M, Chen Q, Chowdhury NK, Fang B, et al. A comparison of 3D shape retrieval methods based on a large-scale benchmark supporting multimodal queries. Comput Vis Image Underst 2015;131:1–27.
[11] Krizhevsky A, Sutskever I, Hinton GE. ImageNet classification with deep convolutional neural networks. In: Proceedings of the 25th international conference on neural information processing systems - Volume 1. NIPS '12, Red Hook, NY, USA: Curran Associates Inc.; 2012, p. 1097–105.
[12] Feng Y, Zhang Z, Zhao X, Ji R, Gao Y. Gvcnn: Group-view convolutional neural networks for 3d shape recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition. 2018, p. 264–72.
[13] Hamdi A, Giancola S, Ghanem B. Mvtn: Multi-view transformation network for 3d shape recognition. In: Proceedings of the IEEE/CVF international conference on computer vision. 2021, p. 1–11.

[14] Feng Y, Gao Y, Zhao X, Guo Y, Bagewadi N, Bui N-T, Dao H, Gangisetty S, Guan R, Han X, et al. SHREC'22 track: Open-set 3D object retrieval. Comput Graph 2022;107:231–40.
[15] Jing L, Tian Y. Self-supervised visual feature learning with deep neural networks: A survey. IEEE Trans Pattern Anal Mach Intell 2020;43(11):4037–58.
[16] Devlin J, Chang M-W, Lee K, Toutanova K. Bert: Pre-training of deep bidirectional transformers for language understanding. 2018, arXiv preprint arXiv:1810.04805.
[17] Radford A, Wu J, Child R, Luan D, Amodei D, Sutskever I, et al. Language models are unsupervised multitask learners. OpenAI Blog 2019;1(8):9.
[18] Bachman P, Hjelm RD, Buchwalter W. Learning representations by maximizing mutual information across views. Adv Neural Inf Process Syst 2019;32.
[19] Ye M, Zhang X, Yuen PC, Chang S-F. Unsupervised embedding learning via invariant and spreading instance feature. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2019, p. 6210–9.
[20] Wang J, Jiao J, Bao L, He S, Liu Y, Liu W. Self-supervised spatio-temporal representation learning for videos by predicting motion and appearance statistics. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2019, p. 4006–15.
[21] Ahsan U, Madhok R, Essa I. Video jigsaw: Unsupervised learning of spatiotemporal context for video action recognition. In: 2019 IEEE winter conference on applications of computer vision. WACV, IEEE; 2019, p. 179–89.
[22] Sermanet P, Lynch C, Chebotar Y, Hsu J, Jang E, Schaal S, Levine S. Time-contrastive networks: Self-supervised learning from video. In: Proceedings of international conference in robotics and automation (ICRA). 2018, URL: http://arxiv.org/abs/1704.06888.
[23] Verleysen A, Biondina M, wyffels F. Learning self-supervised task progression metrics: a case of cloth folding. Appl Intell 2023;53(2):1725–43.
[24] Jumper J, Evans R, Pritzel A, Green T, Figurnov M, Ronneberger O, Tunyasuvunakool K, Bates R, Žídek A, Potapenko A, et al. Highly accurate protein structure prediction with AlphaFold. Nature 2021.
[25] Lu AX, Zhang H, Ghassemi M, Moses A. Self-supervised contrastive learning of protein representations by mutual information maximization. BioRxiv 2020. 2020-09.
[26] Oord Avd, Li Y, Vinyals O. Representation learning with contrastive predictive coding. 2018, arXiv preprint arXiv:1807.03748.
[27] Schneider S, Baevski A, Collobert R, Auli M. Wav2vec: Unsupervised pre-training for speech recognition. 2019, arXiv preprint arXiv:1904.05862.
[28] Li W, Zhao Z, Liu A-A, Gao Z, Yan C, Mao Z, Chen H, Nie W. Joint local correlation and global contextual information for unsupervised 3D model retrieval and classification. IEEE Trans Circuits Syst Video Technol 2022;32(5):3265–78. http://dx.doi.org/10.1109/TCSVT.2021.3099496.
[29] Luciano L, Hamza AB. Geodesic-based 3D shape retrieval using sparse autoencoders. In: Proceedings of the 11th eurographics workshop on 3D object retrieval. 2018, p. 21–8.
[30] Jing L, Zhang L, Tian Y. Self-supervised feature learning by cross-modality and cross-view correspondences. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2021, p. 1581–91.
[31] Wu Z, Song S, Khosla A, Yu F, Zhang L, Tang X, Xiao J. 3D shapenets: A deep representation for volumetric shapes. In: Proceedings of the IEEE conference on computer vision and pattern recognition. 2015, p. 1912–20.
[32] Chang AX, Funkhouser T, Guibas L, Hanrahan P, Huang Q, Li Z, Savarese S, Savva M, Song S, Su H, et al. Shapenet: An information-rich 3d model repository. 2015, arXiv preprint arXiv:1512.03012.
[33] Su H, Maji S, Kalogerakis E, Learned-Miller E. Multi-view convolutional neural networks for 3d shape recognition. In: Proceedings of the IEEE international conference on computer vision. 2015, p. 945–53.
[34] Qi CR, Yi L, Su H, Guibas LJ. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. Adv Neural Inf Process Syst 2017;30.
[35] Wang Y, Sun Y, Liu Z, Sarma SE, Bronstein MM, Solomon JM. Dynamic graph cnn for learning on point clouds. Acm Trans Graph (Tog) 2019;38(5):1–12.
[36] Bardes A, Ponce J, Lecun Y. VICReg: Variance-invariance-covariance regularization for self-supervised learning. In: ICLR 2022-10th international conference on learning representations. 2022.
[37] Saupe D, Vranić DV. 3D model retrieval with spherical harmonics and moments. In: Pattern recognition: 23rd DAGM symposium Munich, Germany, September 12–14, 2001 Proceedings 23. Springer; 2001, p. 392–7.
[38] Kuo C-T, Cheng S-C. 3D model retrieval using principal plane analysis and dynamic programming. Pattern Recognit 2007;40(2):742–55.
[39] Chen D-Y, Tian X-P, Shen Y-T, Ouhyoung M. On visual similarity based 3D model retrieval. In: Computer graphics forum, Vol. 22. Wiley Online Library; 2003, p. 223–32.
[40] Tangelder JW, Veltkamp RC. A survey of content based 3D shape retrieval methods. Multimedia Tools Appl 2008;39(3):441–71.
[41] Maturana D, Scherer S. Voxnet: A 3d convolutional neural network for real-time object recognition. In: 2015 IEEE/RSJ international conference on intelligent robots and systems. IROS, IEEE; 2015, p. 922–8.
[42] Riegler G, Osman Ulusoy A, Geiger A. Octnet: Learning deep 3d representations at high resolutions. In: Proceedings of the IEEE conference on computer vision and pattern recognition. 2017, p. 3577–86.
[43] Simonyan K, Zisserman A. Very deep convolutional networks for large-scale image recognition. 2014, arXiv preprint arXiv:1409.1556.
[44] He K, Zhang X, Ren S, Sun J. Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition. 2016, p. 770–8.
[45] Deng J, Dong W, Socher R, Li L-J, Li K, Fei-Fei L. Imagenet: A large-scale hierarchical image database. In: 2009 IEEE conference on computer vision and pattern recognition. Ieee; 2009, p. 248–55.
[46] Kanezaki A, Matsushita Y, Nishida Y. Rotationnet: Joint object categorization and pose estimation using multiviews from unsupervised viewpoints. In: Proceedings of the IEEE conference on computer vision and pattern recognition. 2018, p. 5010–9.
[47] Lu D, Xie Q, Wei M, Xu L, Li J. Transformers in 3d point clouds: A survey. 2022, arXiv preprint arXiv:2205.07417.
[48] Pan X, Xia Z, Song S, Li LE, Huang G. 3D object detection with pointformer. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. CVPR, 2021, p. 7463–72.
[49] Su J-C, Gadelha M, Wang R, Maji S. A deeper look at 3d shape classifiers. In: Proceedings of the European conference on computer vision (ECCV) workshops. 2018.
[50] Kramer MA. Nonlinear principal component analysis using autoassociative neural networks. AIChE J 1991;37(2):233–43.
[51] Becker S, Hinton GE. Self-organizing neural network that discovers surfaces in random-dot stereograms. Nature 1992;355(6356):161–3.
[52] Bromley J, Guyon I, LeCun Y, Säckinger E, Shah R. Signature verification using a "siamese" time delay neural network. Adv Neural Inf Process Syst 1993;6.
[53] Chopra S, Hadsell R, LeCun Y. Learning a similarity metric discriminatively, with application to face verification. In: 2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05), Vol. 1. IEEE; 2005, p. 539–46.
[54] He K, Chen X, Xie S, Li Y, Dollár P, Girshick R. Masked autoencoders are scalable vision learners. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2022, p. 16000–9.
[55] Hadsell R, Chopra S, LeCun Y. Dimensionality reduction by learning an invariant mapping. In: 2006 IEEE computer society conference on computer vision and pattern recognition (CVPR'06), Vol. 2. IEEE; 2006, p. 1735–42.
[56] Dosovitskiy A, Springenberg JT, Riedmiller M, Brox T. Discriminative unsupervised feature learning with convolutional neural networks. Adv Neural Inf Process Syst 2014;27.
[57] Sajjadi M, Javanmardi M, Tasdizen T. Regularization with stochastic transformations and perturbations for deep semi-supervised learning. Adv Neural Inf Process Syst 2016;29.
[58] Chen T, Kornblith S, Norouzi M, Hinton G. A simple framework for contrastive learning of visual representations. In: International conference on machine learning. PMLR; 2020, p. 1597–607.
[59] He K, Fan H, Wu Y, Xie S, Girshick R. Momentum contrast for unsupervised visual representation learning. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2020, p. 9729–38.
[60] Grill J-B, Strub F, Altché F, Tallec C, Richemond P, Buchatskaya E, Doersch C, Avila Pires B, Guo Z, Gheshlaghi Azar M, et al. Bootstrap your own latent-a new approach to self-supervised learning. Adv Neural Inf Process Syst 2020;33:21271–84.
[61] Chen X, He K. Exploring simple siamese representation learning. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2021, p. 15750–8.
[62] Zbontar J, Jing L, Misra I, LeCun Y, Deny S. Barlow twins: Self-supervised learning via redundancy reduction. In: International conference on machine learning. PMLR; 2021, p. 12310–20.
[63] Ermolov A, Siarohin A, Sangineto E, Sebe N. Whitening for self-supervised representation learning. In: International conference on machine learning. PMLR; 2021, p. 3015–24.
[64] Leng B, Guo S, Zhang X, Xiong Z. 3D object retrieval with stacked local convolutional autoencoder. Signal Process 2015;112:119–28.
[65] Zhu Z, Wang X, Bai S, Yao C, Bai X. Deep learning representation using autoencoder for 3D shape retrieval. Neurocomputing 2016;204:41–50.
[66] Furuya T, Ohbuchi R. Transcoding across 3D shape representations for unsupervised learning of 3D shape feature. Pattern Recognit Lett 2020;138:146–54.
[67] Wang T, Isola P. Understanding contrastive representation learning through alignment and uniformity on the hypersphere. In: International conference on machine learning. PMLR; 2020, p. 9929–39.
[68] Tishby N, Pereira FC, Bialek W. The information bottleneck method. 2000, arXiv preprint physics/0004057.
[69] Tishby N, Zaslavsky N. Deep learning and the information bottleneck principle. In: 2015 Ieee information theory workshop (Itw). IEEE; 2015, p. 1–5.

[70] Qi CR, Su H, Mo K, Guibas LJ. Pointnet: Deep learning on point sets for 3d classification and segmentation. In: Proceedings of the IEEE conference on computer vision and pattern recognition. 2017, p. 652–60.
[71] Caron M, Touvron H, Misra I, Jégou H, Mairal J, Bojanowski P, Joulin A. Emerging properties in self-supervised vision transformers. In: Proceedings of the IEEE/CVF international conference on computer vision. 2021, p. 9650–60.
[72] Kim S, Chi H-g, Hu X, Huang Q, Ramani K. A large-scale annotated mechanical components benchmark for classification and retrieval tasks with deep neural networks. In: Computer vision–ECCV 2020: 16th European conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XVIII 16. Springer; 2020, p. 175–91.
[73] Dawson-Haggerty, et al. Trimesh. 2017, URL: https://trimsh.org/.
[74] Weisstein E. Triangle point picking. 1999, URL: https://mathworld.wolfram.com/TrianglePointPicking.html.
[75] Paszke A, Gross S, Chintala S, Chanan G, Yang E, DeVito Z, Lin Z, Desmaison A, Antiga L, Lerer A. Automatic differentiation in PyTorch. 2017.
[76] Falcon WA, et al. Pytorch lightning. GitHub; 2019, URL: https://github.com/Lightning-AI/lightning.
[77] Bardes A, Ponce J, LeCun Y. VICReg implementation. 2022, URL: https://github.com/facebookresearch/vicreg.
[78] Wijmans E. Pointnet++ Pytorch. 2018, URL: https://github.com/erikwijmans/Pointnet2_PyTorch.
[79] Chaulet N, et al. 3D point cloud kernels. 2019, URL: https://github.com/torch-points3d/torch-points-kernels.