Learning by Sorting: Self-supervised Learning with Group Ordering Constraints

Nina Shvetsova[1,2,3] Felix Petersen[4] Anna Kukleva[2] Bernt Schiele[2] Hilde Kuehne[1,3,5]
[1]Goethe University Frankfurt, [2]Max-Planck-Institute for Informatics, [3]University of Bonn, [4]Stanford University, [5]MIT-IBM Watson AI Lab
{shvetsov@uni-frankfurt.de, mail@felix-petersen.de}

Abstract

Contrastive learning has become an important tool in learning representations from unlabeled data mainly relying on the idea of minimizing distance between positive data pairs, e.g., views from the same images, and maximizing distance between negative data pairs, e.g., views from different images. This paper proposes a new variation of the contrastive learning objective, Group Ordering Constraints (GroCo), that leverages the idea of sorting the distances of positive and negative pairs and computing the respective loss based on how many positive pairs have a larger distance than the negative pairs, and thus are not ordered correctly. To this end, the GroCo loss is based on differentiable sorting networks, which enable training with sorting supervision by matching a differentiable permutation matrix, which is produced by sorting a given set of scores, to a respective ground truth permutation matrix. Applying this idea to groupwise pre-ordered inputs of multiple positive and negative pairs allows introducing the GroCo loss with implicit emphasis on strong positives and negatives, leading to better optimization of the local neighborhood. We evaluate the proposed formulation on various self-supervised learning benchmarks and show that it not only leads to improved results compared to vanilla contrastive learning but also shows competitive performance to comparable methods in linear probing and outperforms current methods in k-NN performance. [1]

1. Introduction

Self-supervised learning has become a topic of growing interest over the last years as it allows models to learn representations from large-scale data without the need for human annotation. Many approaches rely on the idea of contrastive learning and were able not only to narrow the gap to the supervised learning performance in vision [23, 14, 60, 30, 3, 63], but also to train state-of-the-art vision-language [52, 56] and multimodal models [40]. All of these methods rely on the concept of the pairwise contrastive loss, which is based on the idea that a so-called positive pair, e.g., an image serving as an anchor and an augmentation of the same image, should be closer to each other in an embedding space than a so-called negative pair, e.g., a pair made up of an anchor image and a different image, should be far away from each other. However, it has been noted that the idea of a pairwise contrastive loss also has some limitations, such as the alignment of the embedding space based on individual pairs. Several attempts have been made to address this issue, e.g., combining the contrastive idea with concepts based on local neighborhoods, such as clustering (SwAV [11]), or minimizing distances between multiple positive pairs for the same instance together (Whitening [24]). Another limitation of the contrastive loss is that is that the embedding space is optimized with respect to all negatives, i.e., even negatives that are far away

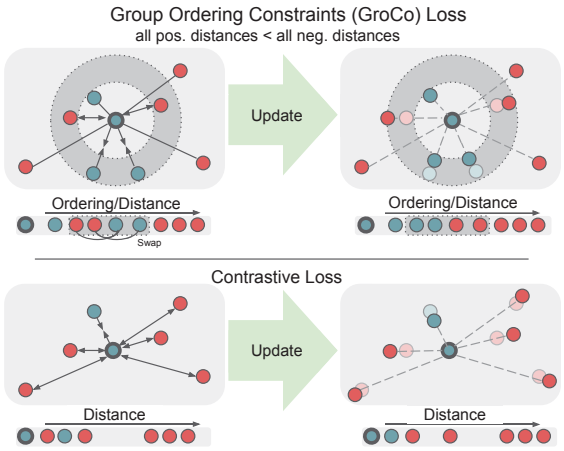

Figure 1. The idea of the proposed group ordering constraints loss compared to pairwise contrastive losses: GroCo arranges positive and negative data points so that the largest distance to positives must be smaller than the smallest distance to negative points. To this end, the loss implicitly minimizes the amount of necessary swap operation to achieve the ordering constraint. Thus, it focuses on overlapping positives and negatives compared to standard contrastive losses that minimize resp. maximize all pairwise distances.

[1] https://github.com/ninatu/learning_by_sorting

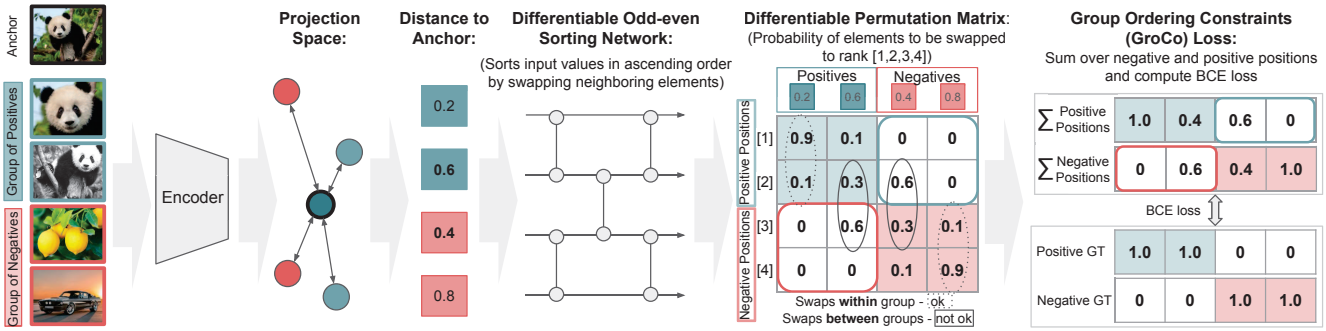

Figure 2. Overview of the proposed loss: Distances of positives and negatives are computed with respect to an anchor. The concatenated distances are sorted via a differentiable sorting network that computes the *swapping probability*. The result is a differentiable permutation matrix, in which the column values can be considered as the probabilities of sorting the elements to the corresponding positions. To enforce only the relationships *between groups*, we sum over the positive and negative rows of the permutation matrix. The loss is then computed as the BCE between the row-wise entries and the ground truth.

from the anchor will contribute to the optimization of the representation. Other methods were proposed to address this issue, such as hard negative selection by controlling the hardness of examples [53] or negative selection by sparse support vectors [57]. Nevertheless, these methods still require manual selection of the hardness level [53] or incur an additional optimization cost [57].

To shift away from the concept of minimizing resp. maximizing all pairwise distances, this paper proposes a variation of the contrastive learning formulation, namely Group Ordering Constraints (GroCo). The idea of GroCo is that positive and negative distances should be sorted in a way that any positive should be closer to an anchor image than any negative, thus forming a group of positive pairs and a group of negative pairs. The idea is illustrated in Figure 1. In comparison to pairwise contrastive losses, the GroCo loss combines the distance information of groups of positive and negative pairs and optimization mainly depends on incorrectly "sorted" pairs. To enforce the group ordering constraints in the projection space, we propose the idea of *learning by sorting*: we suggest sorting positives and negatives by distance to the anchor image in a differentiable way and swapping them if they are in the wrong order. This leads to a more holistic approach considering all relationships between data points, thereby better utilizing and optimizing the embeddings (esp. for multiple positive pairs), and leading to improved down-stream performance. To create an end-to-end training pipeline, we leverage recent advances in differentiable sorting [21, 28, 44, 45, 46, 47, 48]. Specifically, we utilize a differentiable sorting algorithm to obtain a differentiable permutation matrix for sorting a list of distances to the positive and negative images, as shown in Figure 2. If we would know the full ground truth orderings among positives and negatives (such as which positive sample should be closer to the anchor than another positive sample), we could create a ground truth permutation matrix, and calculate how much the predicted permutation matrix would deviate from the ground truth one [28, 21, 45, 47]. Because we do not know the ground truth distance ordering within the positive or the negative groups, we propose the GroCo loss as a relaxed formulation of the original sorting supervision that captures how many negative elements appear in the positive positions and vice versa. The proposed GroCo loss alleviates some aspects of vanilla contrastive learning: first, it treats positive and negative pairs as groups instead of individual pairs, and second, the resulting group ordering focuses on *optimizing the local neighborhood* around an anchor image by mainly optimizing too close negative and too distant positives, rather than optimizing all data points at once. Thus, it implicitly also focuses on the strongest positive (furthest from the anchor) and strongest negative (closest to the anchor) examples.

To show the capabilities of the proposed approach, we evaluated it on various competitive self-supervised learning benchmarks, namely in the context of linear probing, k-NN classification, transfer learning, as well as image retrieval. The evaluation shows that the model trained via group ordering constraints outperforms contrastive learning frameworks in linear probing and transfer learning and excels in the context of shaping local neighborhoods on the tasks such as k-NN classification and image retrieval.

The contributions of this work are summarized as follows:

- We advance the concept of contrastive learning by introducing Group Ordering Constraints (GroCo) that treat positive and negative elements as groups rather than individual pairs as in conventional contrastive learning.
- To derive a loss that optimizes the proposed constraints, we harness recent differentiable sorting methods and obtain a loss that suggests sorting positive and negative elements and swapping them if they are in the wrong order — thus, we introduce a new contrastive learning method called *learning by sorting*.
- The proposed method provides embeddings that achieve

competitive performance in linear probing and are especially suitable to model the local neighborhoods and outperform contrastive learning frameworks on a wide range of nearest-neighbor tasks.

2. Related Work

2.1. Self-supervised Representation Learning

Contrastive methods: Over the last years, self-supervised learning methods that enforce the model to be robust to different image distortions achieved great performance improvements in self-supervised learning [14, 30, 15, 16]. Such methods generally rely on sampling two augmented views of the image—a positive pair—and minimize the distance between those in the embedding space. To prevent the model from learning a trivial solution for any input, contrastive methods introduce the concept of a negative pair, i.e., two different images, to *contrast* positive against negative pairs. While earlier contrastive methods relied on the triplet loss [55], the probably most prominent method in many self-supervised learning scenarios is the InfoNCE loss, which is often referred to as a contrastive loss [14, 30, 51, 1], which requires accumulating strong negatives via a memory bank [30] or a large batch size [14]. Many extensions have been proposed to further improve the performance of this idea: data augmentation strategies [14, 62], projection head design [15], hard negative sampling [53], increasing the richness of positives with nearest neighbours [23], or mitigating the effect of false negatives [32]. The variation of the contrastive method proposed in this work should not be considered as opposed but rather as orthogonal to other approaches relying on positive and negative pairs because it changes the loss function itself and can therefore be used, e.g., on top of other techniques.

Alternative methods: There are also methods [17, 27] that do not rely on negatives and only maximize agreement between positive views. Such methods prevent collapsing of the representation space by using asymmetric architectures applied to different views [17, 27], an additional teacher network [27, 12], stop gradient [17, 27, 12], feature whitening [24], or information maximization [65, 4]. Another set of methods [11, 10, 2] utilizes clustering of latent embeddings. ReSSL [67] leverages relations in an embedding space in a self-labeling way, namely aligning the similarities between weakly-augmented images to the similarities to the similarities between strongly augmented images. SwAV [11] additionally proposes sampling more augmentations in a multi-crop setting, where two full-size augmented images are sampled together with several smaller crops, and Whitening [24] utilized more full-resolution samples. While currently methods relying on positive samples seem to outperform their classical contrastive counterparts, the we show that especially local neighborhood learning can profit from relying on positive and negative samples.

2.2. Differentiable Sorting and Ranking

Differentiable sorting and ranking methods provide a pipeline that allows training neural networks with ordering supervision in an end-to-end fashion with gradient descent [5, 21, 28, 44, 45, 46, 47, 48, 49]. Earlier pairwise learning-to-rank methods, such as RankNet [8] or LambdaRank [7], and listwise methods, such as SoftRank [59] or ListNet [9], are mostly based on heuristics and aim to optimize ranking metrics, e.g., NDCG. Many of the latest differentiable sorting approaches [21, 28, 44, 45, 47, 48, 49] focus on obtaining a differentiable relaxation for the sorting operator. The sorting operator can be seen as a function returning a permutation matrix that indicates the permutation necessary to sort the sequence of values (the matrix that multiplied with an input vector returns a sorted output vector.) In this context, differentiable sorting refers to relaxing the (hard) permutation matrix to a differentiable permutation matrix via continuous relaxations. The differentiable permutation matrix for a given sequence of values, which can, e.g., be scores predicted by a neural network, can then be used to compute the loss by comparison to a ground truth permutation matrix. Recently, multiple methods for relaxing the permutation matrix have been proposed, including an argsort approximation by unimodal row-stochastic matrices [28, 49], a formulation of entropy-regularized optimal transport [21], as well as networks of differentiable swap operations (differentiable sorting networks) [45, 47]. The latter method composes the full permutation matrix as a product of permutation matrices that arise from comparing only two elements at a time (usually neighbors) and either swapping them or not swapping them. Practically, differentiable sorting has been leveraged in various contexts, including recommender systems [37, 58], image patch selection [20], selection experts in multi-task learning [29], attention mechanisms [66], and audio representation learning [13]. To the best of our knowledge, the proposed method is the first work to leverage ordering supervision for self-supervised learning of visual representations.

3. Method

Given a dataset of images $\{x_i\}_{i=1}^{M} \subseteq \mathcal{X}$, the goal is to learn an encoder $g : \mathcal{X} \to \mathbb{R}^d$ that extracts image representations that can later be used for downstream tasks.

3.1. Training Pipeline

As in standard contrastive losses, the proposed method considers several augmented views of the same image as positive examples, which should be close together in the embedding space, and different images as negative examples, which should be apart in the embedding space. Starting from mini-batches of B images, $m \geq 2$ randomly augmented views are generated for each image, resulting in $m \cdot B$ data points overall per batch. Note that if $m = 2$, the

proposed method is close to the original contrastive learning setup [32, 14, 17, 4, 65]. The augmented views are processed with the encoder network $g(\cdot)$ and an MLP projection head $h(\cdot)$, that maps images to the latent space where distances between views are calculated. For each data point serving as an anchor x^a, there are $m-1$ positive examples $\{x_i^p\}_{i=1}^{m-1}$ and $m \cdot (B-1)$ negative examples $\{x_i^n\}_{i=1}^{m \cdot (B-1)}$. The measure of distance between data points is the cosine distance defined as: $\mathsf{d}(x, y) = -\frac{x^\top y}{\|x\| \|.y\|}$.

3.2. Group Ordering Constraints (GroCo)

In order to consider positives and negatives not individually but instead as a group, the proposed loss extends the contrastive loss to the idea that *the group of positives* should be closer to the anchor image than *the group of negatives* in the embedding space resulting in *group ordering constraints (GroCo)*. To simplify the notation, the distance between data point x^a and its positive x_i^p and negative examples x_i^n is denoted as $d_i^p = \mathsf{d}(x^a, x_i^p)$ and $d_i^n = \mathsf{d}(x^a, x_i^n)$. Assuming that K positives $x_1^p, ..., x_K^p$ are ordered with respect to their distances to the anchor x^a as $d_1^p \leq ... \leq d_K^p$ and N negatives $x_1^n, ..., x_N^n$ as $d_1^n \leq ... \leq d_N^n$, then the *group ordering constraints* can be defined as

$$d_1^p \leq ... \leq d_K^p < d_1^n \leq ... \leq d_N^n. \tag{1}$$

We note that all elements are considered in the constraints and the relevant constraint is the (bold) $<$ in the center. We remark that, although the constraint is already fulfilled if the largest positive distance d_K^p is smaller than the smallest negative distance d_1^n, it is suboptimal to define loss only on those elements. Comparing only the smallest negative and largest positive ignores other negatives (e.g., the second smallest) and positives (e.g., the second largest) that might also be misaligned, and such a loss would ignore them. In the next section, we propose our novel loss that optimizes the GroCo constraints implicitly.

3.3. Learning by Sorting

To enforce the constraint, the GroCo loss leverages recent advances in differentiable sorting [45, 47], which allow to derive a loss that fulfills ordering constraints. Namely, the training procedure can be seen as sorting positives and negatives in the embedding space with respect to an anchor image and swapping them if they are in the incorrect order, which relates to the proposed idea of *learning by sorting*.

3.3.1 Differentiable Sorting Networks

This section provides a review of the differentiable sorting algorithm *differentiable sorting networks* [45] used for the proposed loss function. Note that "networks" in "sorting networks" are not "neural networks" but instead refers to a category of sorting algorithms in the computer science literature [34] with no trainable parameters.

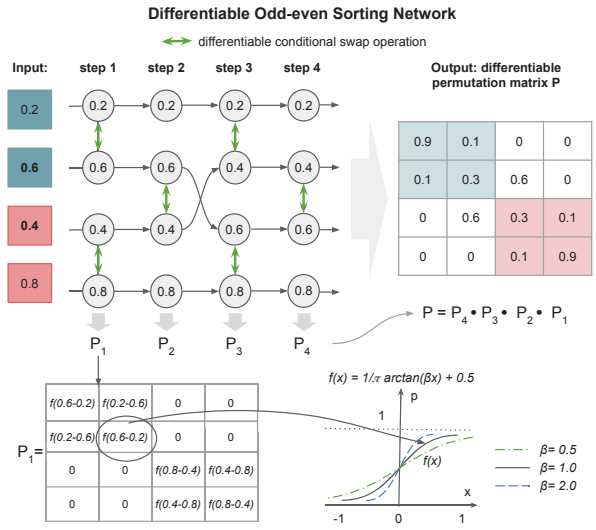

Figure 3. Overview of a differentiable sorting network with odd-even sorting: The network compares neighboring elements starting from odd and even indices alternatingly in each step and applies a differentiable swap operation if elements are in the wrong order. The swap operations on each step s also define a differentiable permutation matrix P_s. The network output is a differentiable permutation matrix P, defined as the multiplication of matrices of each step.

Differentiable sorting networks, e.g., based on the odd-even sorting network, sort an input sequence of $K+N$ elements in non-descending order as shown in Figure 3. They are defined as the concatenation of functions, e.g., representing the swap operations in each layer of an odd-even sorting network, where each function refers to one step of sorting, and pairs of elements of the input sequence are compared and swapped if they are in the wrong order via a *conditional swap operation*. For the odd-even sorting network, the algorithm compares neighbored elements on odd and even indices alternatingly and requires $K+N$ steps to sort a given input sequence of length $K+N$. By relaxing the conditional swap operator to a differentiable one, sorting networks can be made differentiable [45]. The conditional swap operation for elements (d_i, d_j) where $i < j$ can be defined as $d_i' = \min(d_i, d_j)$, $d_j' = \max(d_i, d_j)$, and the differentiable relaxation [47] of this operation is:

$$\begin{aligned} d_i' &= \text{softmin}(d_i, d_j) = d_i f(d_j - d_i) + d_j f(d_i - d_j), \\ d_j' &= \text{softmax}(d_i, d_j) = d_i f(d_i - d_j) + d_j f(d_j - d_i) \end{aligned} \tag{2}$$

with

$$f(x) = \frac{1}{\pi} \arctan(\beta x) + 0.5. \tag{3}$$

The hyperparameter $\beta > 0$ denotes an inverse temperature. For $\beta \to \infty$, the relaxation converges to the discrete swap operation. The differentiable conditional swap operation for the elements (d_i, d_j) can be defined as a permutation matrix $P_{swap(d_i,d_j)} \in R^{(K+N)\times(K+N)}$, which is an identity ma-

trix except for entries $P_{ii}, P_{ij}, P_{jj}, P_{ji}$ defined as:

$$\begin{aligned} P_{ii} = P_{jj} = f(d_j - d_i), \\ P_{ij} = P_{ji} = f(d_i - d_j). \end{aligned} \tag{4}$$

The permutation matrix P_s for step s is the product of matrices corresponding to independent (and thus parallel) swap operations in this step $P_s = \prod_{i \in R} P_{swap(d_i, d_{i+1})}$, where R is the set of odd indices if s is odd and the set of even indices if s is even. The complete permutation matrix P is defined as $P = P_{K+N} \cdot ... \cdot P_1$. In the discrete case, each column of a permutation matrix has exactly one entry of 1, indicating the position where the element that corresponds to this column should be placed. In the relaxed version, column values can be seen as a distribution over possible positions of the element. If the correct order of input values is known, we can create a ground truth matrix Q and define the loss as $L = \frac{1}{(K+N)^2} \sum_{i,j} \mathrm{BCE}(P_{ij}, Q_{ij})$ where BCE refers to binary cross-entropy.

3.3.2 GroCo Loss

If there was a known ground truth order of positives and negatives, the loss could be calculated directly based on this ground truth permutation matrix. However, as the algorithm is based on random augmentations, there are no known orders among positives and among negatives. Thus, the only information available is whether a pair belongs to the positive or the negative group.

To derive a loss to fulfil this group ordering constraint, we start by ordering positives and negatives first separately with respect to the computed distances to the anchor image as $d_1^p \leq ... \leq d_K^p$ for positives and $d_1^n \leq ... \leq d_N^n$ for negatives. As shown in Figure 3, positive and negative distances are then concatenated in a list as:

$$[d_1^p, ..., d_K^p, d_1^n, ..., d_N^n]. \tag{5}$$

Even though elements in the positive and negative are ordered, it is still open if the constraint $d_i^p < d_j^n$ is fulfilled for any $1 \leq i \leq K$ and $1 \leq j \leq N$.

A differentiable sorting network is applied to the concatenated list and a differentiable permutation matrix is obtained for sorting the list in non-descending order. As shown in 3, values in the permutation matrix column can be seen as probabilities to sort the corresponding element to the different positions, e.g., P_{11} would be the probability for assigning the first element in the list (d_1^p) to position 1, P_{21} to position 2, etc. Therefore, the sum of the first K elements in a column can be considered as a probability being sorted inside the first K elements. Thus, for a permutation matrix of size $(K+N) \times (K+N)$ the sum of the first K rows results in probabilities of being sorted in *positive places* and the later columns (from $K+1$ to $K+N$) in *negative places*. To enforce positives to be sorted in the positive places and negatives in the negatives places, the respective loss (with $\mathbb{1}$ as the indicator function) is defined as

$$\begin{aligned} L = \frac{1}{2(K+N)} \sum_{i=1}^{K+N} \Bigg(\mathrm{BCE}\left(\textstyle\sum_{k=1}^{K} P_{ki}, \mathbb{1}_{i \leq K}\right) + \\ + \mathrm{BCE}\left(\textstyle\sum_{k=K+1}^{K+N} P_{ki}, \mathbb{1}_{i > K}\right) \Bigg). \end{aligned} \tag{6}$$

As illustrated in Figure 2, the proposed loss is a relaxation of the sorting supervision in a way that it considers two types of swap operations: swap operations *within* the group of positive and negative samples, which should not contribute to the loss, and swap operations *between* the groups, which violate the positive-negative ordering assumption and which are used as the optimization criterion.

Role of β. One relevant hyper parameter is the inverse temperature β in differentiable swap operation (Equation 3), which corresponds to the degree of relaxation of the swap operation that converges to a discrete case when $\beta \to \infty$ (Figure 3). Therefore with lower β the swap operation is more "soft", which is beneficial for optimization, but the relaxation error accumulated by each step is larger, and vice versa in the case of larger β. With higher β even a small difference between values results in a high probability for a swap or not swap operation, resulting in a smaller margin between the positive and negatives group.

Number of samples. Since the strongest negatives have the strongest effect on the loss function, the selection can be limited to only the top-N strongest negatives. Further, as more negatives also result in more layers in the sorting network and each layer contributes to the overall differential permutation matrix, more layers also result in a softer swap probabilities. Therefore β should be selected based on a number of elements to sort. An ablation study on this effect is given in Section 4.5.

Role of Pre-ordering. Practically, positives and negatives are pre-ordered among themselves before concatenating and forwarding them to the differentiable sorting network. While the loss will still contrast positives to negatives no matter if the input is ordered or not, it shows that pre-ordering improves the overall performance of the GRoCo loss. This can be attributed to the fact that sorting networks perform comparisons between neighboring elements and swap them if they are in the wrong order. If the input was not pre-ordered and, thus, distances from positive and negative pairs would be mixed, this would result in additional swap operations. Using pre-ordered inputs the focus lies on comparing the strongest positives with the strongest negatives. In this way, elements are considered as a group, and the borders of the groups or their overlapping parts are emphasized in the loss. In Section 4.5 and Figure 4, we provide additional discussions and illustrate this behavior.

4. Experimental Evaluation

4.1. Implementation Details

Unless stated otherwise, the following setup is used for all experiments:

Model. Following previous works [14, 27, 11, 17] Resnet50 [31] is used as the encoder $g(\cdot)$ and an MLP block consisting of three fully connected layers with a size of 2048 and followed by a batch normalization layer [33] is used as the projection head $h(\cdot)$. All batch normalization layers except the last one are followed by a ReLU activation. The dimensionalities of the representation space and the latent space are both 2 048 as in [17].

Training. Following previous works [14, 27, 11, 17], we use the train set of the ImageNet ILSVRC-2012 dataset [54] for self-supervised training without any human annotation. To create m augmented views per image (considering $m = 2, 3, 4$), the DINO augmentation setup [12] is used. The model is trained with the SGD optimizer [64] with a learning rate of $6.0 \times (\text{batch size}/256)$ for 100 epochs and $3.0 \times (\text{batch size}/256)$ for 200 and 400 epochs. We use a cosine scheduler without restarts [38] and 10 epochs warm-up for 200 and 400 epochs training and 1 epoch linear warm-up for 100 epochs training (we find it beneficial for our method but not for SimCLR). During training, the stop gradient operation is used following the self-supervised learning setups of [14, 12, 27]; specifically, stop gradient is performed during distance computation $\mathsf{d}(x^a, x_i^{\cdot}) = \mathsf{d}(x^a, \texttt{stop\_grad}(x_i^{\cdot}))$. While training with stop gradient does not show a direct impact on the overall performance, we observed that it allows training with larger variations of hyperparameters while maintaining stable performance. By default, the top $N = 10$ strongest negatives are sampled from the batch and an inverse temperature of $\beta = 1$ is used. Due to resource constraints, the model is trained with a batch size of 1 024 with mixed precision. On an 8-GPU (NVIDIA A6000) server, training for 100 epochs with $m = 2$ views takes approximately 22 hours.

4.2. Evaluation Procedure

Linear Probing. Linear probing allows to evaluate the learned embedding space by linear evaluation [14, 27, 11, 17], capturing the linear separability of classes. For this, a linear classifier is trained on frozen representations in a fully-supervised way using the ImageNet train set. We follow the standard protocol [17] to train the linear classifier.

k-NN Evaluation. To analyze the local properties of the learned representation, namely how often neighbored data points correspond to the same semantic class, we further evaluate with respect to nearest neighbor classification, predicting the class by a simple weighted k nearest neighbor classifier (k-NN) with $k = \{1, 10, 20\}$ based on cosine distance as used in [12, 11]. Again, we use the ImageNet train set for supervision and test on the val set.

Method	BS	Views	100 ep	200 ep	400 ep
Linear Probing (Top-1)					
Max-Margin [57]	256	2×224	63.8	-	-
MoCo v2† [16] [17]	256	2×224	67.4	69.9	71.0
SimCLR† [14] [17]	4096	2×224	66.5	68.3	69.8
GroCo (ours)	1024	2×224	69.2	70.4	71.1
GroCo (ours)	1024	4×224	**69.6**	**70.6**	71.3
SimSiam [17]	256	2×224	68.1	70.0	70.8
VICReg [4]	2048	2×224	68.6	-	-
Barlow Twins [65]	2048	2×224	68.7	-	-
SwAV† [11]	4096	2×224	66.5	69.1	70.7
ReSSL [67]	256	2×224	-	69.9	-
BYOL† [27]	4096	2×224	66.5	**70.6**	**73.2**
Whitening [24]	4096	4×224	69.4	-	72.6
k-NN (weighted, k=20)					
MoCo v2 [16]	256	2×224	-	55.6	-
SimCLR† [14] [17]	4096	2×224	53.8	57.2	59.2
GroCo (ours)	1024	2×224	60.5	62.9	64.0
GroCo (ours)	1024	4×224	**61.8**	**63.6**	**64.8**
SimSiam [17]	256	2×224	57.4	-	-
SwAV [11]	4096	2×224	-	-	61.3

Table 1. **Comparison to state-of-the-art in linear probing k-NN classification on ImageNet.** We report results for training for $100, 200, 400$ epochs. *Backbone=Resnet50.* † denotes improved reproductions from SimSiam [17].

Method	Epochs	Batch Size	Views	Oxford		Paris	
				M	H	M	H
SimSiam [17]	100	256	2×224	26.89	7.04	46.92	19.31
MoCo v2 [16]	200	256	2×224	23.28	5.07	42.8	17.33
SimCLR [14]	400	4096	2×224	23.27	4.56	46.93	20.19
SwAV [11]	400	4096	2×224	28.01	**8.35**	46.23	17.4
GroCo (ours)	400	1024	2×224	**29.37**	7.11	**54.95**	**26.26**

Table 2. **Comparison to state-of-the-art in image retrieval.** We evaluate image retrieval performance on the Medium (M) and Hard (H) splits of the revisited Oxford and Paris datasets [50]. We evaluate nearest neighbor retrieval performance with ImageNet-trained encoders and report MAP. *Backbone=Resnet50.*

4.3. Comparison to State-of-the-Art

We start with a comparison of the proposed method to state-of-the-art self-supervised learning methods in linear probing and k-NN evaluation on the ImageNet [54], and in image retrieval on the revised Oxford and Paris dataset [50], as well as in transfer learning.

Linear Probing. In the case of linear probing (Table 1), the proposed method is compared to contrastive baselines using positive and negative samples, namely Max-Margin [57], SimCLR [14], and MoCo v2 [16], as well as to alternative methods. We observe that, in the given setting, the proposed loss is able to improve above all contrastive baselines and even outperforms most strong alternative baselines except BYOL [27] and Whitening [24] (in 400 epochs setup) that use $\times 4$ larger batch size and/or an additional teacher network. Another finding is that in the context of linear

Method	Epochs	BS	Aircraft	Caltech101	Cars	Cifar10	Cifar100	DTD	Flowers	Food	Pets	SUN397	VOC2007	Average
MoCo v2	200	256	20.1	78.9	12.3	86.9	61.4	68.7	67.3	47.2	67.2	46.9	73.9	57.35
SimCLR	400	4096	19.3	77.7	15.3	85.4	61.0	70.2	72.5	51.2	68.2	49.0	73.1	58.45
SimSiam	100	256	25.4	80.1	17.3	87.4	65.5	69.3	77.3	52.5	72.5	50.0	72.8	60.92
SwAV	400	4096	25.8	**82.2**	17.8	**88.6**	**66.0**	**71.1**	74.8	50.3	70.4	**52.9**	**75.6**	61.41
GroCo (ours)	400	1024	**27.6**	81.2	**19.1**	86.8	65.8	71.0	**79.2**	**56.3**	**80.6**	52.3	74.6	**63.14**

Table 3. **Comparison to state-of-the-art transfer performance in k-NN classification on 11 classification datasets.** Models are pre-trained on ImageNet. *Backbone=Resnet50, views=2x224.*

Method	BS	Views	100 ep	200 ep	400 ep
Linear Probing (Top-1)					
SwAV [11]	4096	2×224 + 6×96	72.1	73.9	74.6
GroCo (ours)	1024	2×224 + 6×96	71.8	72.9	73.7
k-NN (weighted, k=20)					
SwAV [11]	4096	2×224 + 6×96	61.7	63.7	64.9
GroCo (ours)	1024	2×224 + 6×96	62.3	64.2	65.2

Table 4. **Comparison to SwAV for the multi-crop augmentation strategy.** We report results for training for 100, 200, and 400 epochs. *Backbone=Resnet50.*

Method	Views	k-NN Evaluation			Linear Eval.	
		k=1	k=10	k=20	Top-1	Top-5
SimCLR	2×224	-	-	-	64.3	-
SimCLR‡	2×224	46.0	51.5	51.9	65.7	86.7
SimCLR‡	3×224	44.7	50.0	50.6	65.8	86.8
SimCLR‡	4×224	46.3	52.1	52.6	66.5	87.1
SimCLR‡	2×224+6×96	46.6	51.4	52.0	67.2	87.7
GroCo (ours)	2×224	55.3	60.3	60.5	69.2	88.4
GroCo (ours)	3×224	55.8	61.2	61.6	69.5	88.8
GroCo (ours)	4×224	56.4	61.5	61.8	69.6	88.9
GroCo (ours)	2×224+6×96	**57.2**	**62.0**	**62.3**	**71.8**	**90.4**

Table 5. **Comparison to SimCLR as a contrastive baseline on ImageNet.** *Backbone=Resnet50, #epochs=100, batch size=1024.* ‡ denotes our reproduction.

probing, having four positive samples does not significantly improve the approach compared to only two samples. We attribute this to the fact that missing fine-grained differences in the local neighborhood are compensated by the linear layer training, thus the initial pretraining is less relevant with respect to the local neighborhood in this setting.

k-NN Evaluation. Further, the proposed method is evaluated with respect to the k-NN performance and compared to state-of-the-art methods that officially released weights (Table 1). Here, it can be observed that the proposed method outperforms all methods in this setting. The results demonstrate that the margin, by which the loss improves over other methods, is increased compared to linear probing, where, e.g., for SwAV, the results were mainly on par ($+0.4\%$) in case of linear probing, while here they show a more substantial improvement ($+2.3\%$). A second hint that the strength of this method is in the optimization of local neighborhoods is given by the fact that the k-NN setting also shows an improved performance by leveraging multiple positive examples. This can be an indication that more positive examples contribute to a better local neighborhood in this setting.

Image Retrieval. To further assess the potential of the proposed method in nearest neighbors-based tasks, the model is evaluated on the task of image retrieval in Table 2. Results are reported as the Mean Average Precision (MAP) for the Medium (M) and Hard (H) splits of the datasets as in [12]. Our method outperforms all other methods in this task, confirming good local properties of learned representations.

Transfer Performance. Finally, we compare how well performance transfers on other datasets. In Table 3, we compare ImageNet pre-trained models in zero-shot k-NN evaluation on 11 classification datasets, including FGVC Aircraft [39], Caltech-101 [26], Stanford Cars [35], CIFAR10 [36], CIFAR-100 [36], DTD [19], Oxford 102 Flowers [41], Food-101 [6], Oxford-IIIT Pets [43], SUN397 [61] and Pascal VOC2007 [25]. We observe that the proposed method improves over the SimCLR and MoCo v2 baselines in all classes and on average even outperforms the publicly available SimSiam and SwAV baselines. This can be particularly attributed to the improved performance on the Pets and Food datasets. We credit the increased performance to the fact that both food and animal-related classes often appear in the ImageNet pretraining data, thus, learning a good local embedding helps with those datasets, specifically in the case of k-NN classification.

Multi-crop Augmentation. Since computational cost grows linearly with an increasing number of augmentations, the multi-crop augmentation strategy proposed in SwAV [11] is also considered. The idea is to sample low-resolution *local* views along with the standard 224×224 ones. Use $2 \times 224 + 6 \times 96$ scheme, where with two *global* 224×224 augmented views, six *local* 96×96 views are sampled, giving eight views per image. In this case, we follow the *"local-to-global"* correspondence idea [11, 12] and use only global views as positives for both local and global anchor images. While the proposed method shows slightly lower results compared to clustering-based SwAV, it again improves in the case of k-NN classification (Table 4).

4.4. Comparison to Contrastive Loss

To evaluate the properties of the proposed method in a direct comparison with the pairwise contrastive loss formula-

Method	Views	k-NN Evaluation			Linear Eval.	
		k=1	k=10	k=20	Top-1	Top-5
MoCo v3	2×224	61.5	66.6	66.8	**70.9**	**90.2**
GroCo (ours)	2×224	**63.1**	**67.6**	**67.5**	70.8	89.7

Table 6. **Comparison to MoCo v3 as a contrastive baseline on ImageNet.** *Backbone=ViT-Small, #epochs=100, batch size=1024.*

	k-NN Evaluation			Linear Probing	
	k=1	k=10	k=20	Top-1	Top-5
Triplet Loss (margin=0.8)	46.8	52.5	52.8	63.9	85.4
Triplet Loss (margin=1.6)	47.9	53.4	53.7	64.2	85.3
Triplet Loss (margin=$+\infty$)	47.9	53.3	53.8	64.3	85.3
GroCo (ours)	**55.3**	**60.3**	**60.5**	**69.2**	**88.4**

Table 7. **Comparison on ImageNet with Triplet Loss.** *Backbone=Resnet50, #epochs=100, batch size=1024.*

tion, we compare it further to the classical contrastive learning methods SimCLR [14] and MoCo v3 [18] that use the popular InfoNCE loss [42] as well as to the triplet loss.

Contrastive Loss. First, we compare the performance to the SimCLR method in Table 5, where we also analyze the properties of losses to align local neighborhood based on more positive data points. To ensure an identical setting for both methods, SimCLR is reproduced with a 3-layer MLP projection head [15]. Since SimCLR originally uses only two augmentations per image, we extend it to a group of positives by applying contrastive loss between all possible positive pairs (see supplementary material). While outperforming the SimCLR baseline by $+3\%$ (Top-1) in linear probing, the proposed method also advances in k-NN evaluation by more than $+8\%$ ($k = 20$), demonstrating that the loss helps to learn better representation not only in terms of linear separability but also in terms of local structure. Considering the multi-crop scenario, SimCLR shows mixed results from utilizing more views. While it benefits in linear evaluation, performance slightly decreases in k-NN for $k \in \{10, 20\}$. The proposed method profits from utilizing more positives in both evaluations, resulting in $+1.7\%$ in k-NN ($k = 1$), and $+2.4\%$ in linear evaluation (Top-1). We further compare performance with the more recent and better-engineered MoCo v3 approach [18] with a ViT-Small [22] backbone in Table 6. MoCo v3 additionally utilizes a predictor and a momentum encoder that incur an additional computational cost but are beneficial for contrastive training [18, 12]. We plugin our proposed GroCo loss in MoCo v3 setup (with the same architecture, optimizer, etc.) and perform minimal hyperparameter tuning. Table 6 shows that without any tips and tricks the proposed method outperforms MoCo v3 in all k-NN metrics and has a comparable performance in Top-1 linear probing.

Triplet Loss. We also compare the group ordering loss with the triplet loss formulation $L = \max\left(d_i^p - d_j^n + r, 0\right)$,

Inverse temp. β	Number of negatives N							
	1		5		10		20	
	k-NN	Lin.p.	k-NN	Lin.p.	k-NN	Lin.p.	k-NN	Lin.p.
0.062	52.1	64.5	-	-	-	-		
0.125	52.2	64.5	59.2	68.7	-	-		
0.25	52.1	64.4	59.6	68.6	-	-	-	-
0.5	-	-	59.4	68.4	60.1	69.1	-	-
1	-	-	59.2	68.1	**60.5**	**69.2**	54.6	65.5
2	-	-	-	-	60.2	68.7	55.9	66.0
4	-	-	-	-	-	-	58.6	67.5
8	-	-	-	-	-	-	59.0	68.0
16	-	-	-	-	-	-	53.0	65.8

(a) **An inverse temperature β, a number of negatives N.**

	k-NN Evaluation			Linear Probing	
	k=1	k=10	k=20	Top-1	Top-5
Randomly ordered	54.4	59.2	59.5	68.9	88.2
Pre-ordered	**55.3**	**60.3**	**60.5**	**69.2**	**88.4**

(b) **Pre-ordering in the groups.**

Batch Size	k-NN Evaluation			Linear Probing	
	k=1	k=10	k=20	Top-1	Top-5
256	52.4	57.2	57.2	67.8	88.1
512	53.1	57.9	58.2	68.2	88.0
1024	**55.3**	**60.3**	**60.5**	**69.2**	**88.4**

(c) **Batch size.**

Batch Sampling	k-NN Evaluation			Linear Probing	
	k=1	k=10	k=20	Top-1	Top-5
Regular	**55.3**	**60.3**	**60.5**	**69.2**	**88.4**
With many false negatives	51.5	56.6	56.9	67.6	87.7

(d) **Sensitivity to false negatives in a batch.**

Table 8. **Ablation Experiments.** For (a), we report k-NN performance with k=20, and linear probing Top-1, denoted as "Lin.p". The best results are bolded. Options used to obtain the main results are highlighted. *Backbone=Resnet50, Views=2×224, #epochs=100.*

where r is a margin (Table 7). For a fair comparison, we consider all positive and the 10 strongest negative samples and evaluate different margin parameters. Here, sorting is superior to a triplet loss with hard margin selection.

4.5. Ablation Study

Inverse Temperature, Number of Negatives. Table 8a shows the influence of the number of nearest neighbor negatives N used in the loss, as well as the value of the inverse temperature parameter β (Equation 2). We observe that usage of too many negatives is not be beneficial for the model. Since our loss focuses on negatives that are sorted incorrectly, increasing the number of negatives at some point does not bring any new learning signal (because more distant samples are unlikely to be sorted incorrectly). However, a larger N results in more steps of the sorting network, increasing the degree of relaxation. Using a larger inverse temperature β (leading to a lower degree of relaxation in the

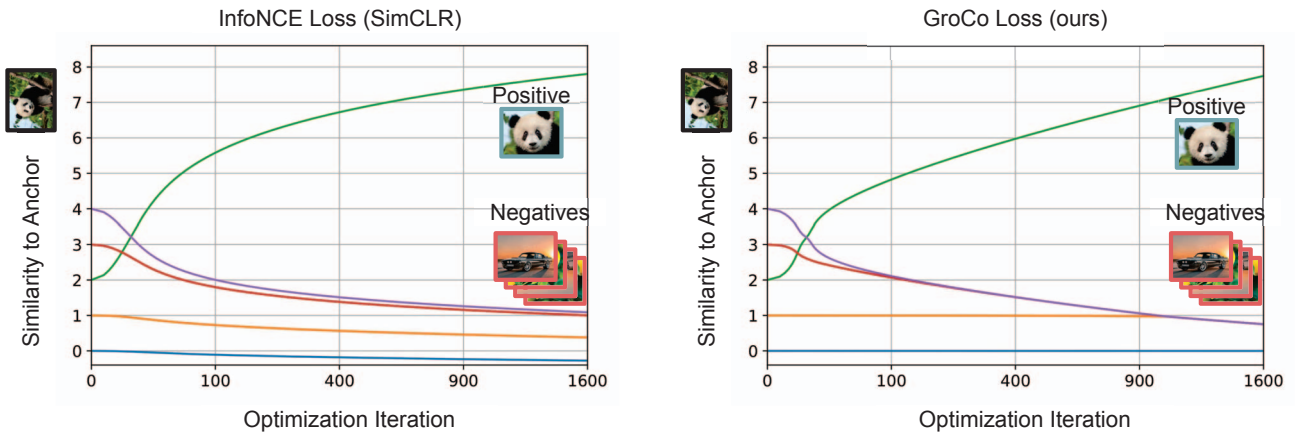

Figure 4. A toy experiment where we optimize five real variables, treating one of them as positive similarity and the other four as negatives, with both the GroCo loss and contrastive InfoNCE loss for multiple iterations. While the InfoNCE loss minimizes all negatives, the GroCo loss works behaves more similar to a margin optimization: negatives that further away from the border are optimized to a smaller degree.

swap operation) we can gain some performance; however, the variance of the gradients is larger with a larger β, which is not beneficial for optimization. We found $N = 10$ and $\beta = 1$ to be an efficient configuration for this setting.

Pre-ordering. In Table 8b, we analyze the impact of pre-ordering elements within negative and positive groups before forwarding them to the sorting network. We find that it achieves good performance even without pre-ordering, but also that pre-ordering further strengthens the method.

Batch Size. A large batch size can be an important factor in obtaining good performance for many self-supervised learning methods. Results in Table 8c show that our method also benefits from a large batch size, which we attribute to utilizing stronger negatives from a larger batch.

False negatives. To assess the sensitivity of proposed methods to the false negatives, we artificially sample a batch in a way that for each instance, there are three more instances of the same class on average (acting as false negatives). In Table 8d, we observe a performance decline in this scenario ("With many false negatives"). Given GroCo's implicit emphasis on strong negatives, enhancing its robustness to false negatives would require further adjustments, which we leave for future work.

Training Time. We also consider the training time of our model compared to SimCLR baseline. To eliminate the influence of distributed training, we measure the average time of training iteration on one GPU. We find that the iteration time of both models is comparable, 514ms for SimCLR vs 526ms for the proposed methods for a batch size of 128.

Optimization of Negatives. To better understand the rationale behind GroCo's superiority in k-NN tasks, we analyze the difference between the GroCo loss and contrastive InfoNCE loss (used in SimCLR) in structuring the embedding space. We conduct a toy experiment where we optimize five real variables treating one of them as positive similarity and the other four as negative similarities with both losses for multiple iterations. We demonstrate the optimization process in Figure 4. Although both losses elevate positive similarities and lower negative similarities, they differ in the optimization of the negatives. While the InfoNCE loss minimizes all negatives (even pushing the blue curve substantially below zero), the GroCo loss works behaves more similar to a margin optimization: negatives that are not on the border to the positive are optimized to a smaller degree (the blue curve is pushed only slightly below zero). This highlights GroCo's focus on neighborhood optimization.

5. Conclusion

In this paper, an alternative approach to the common pairwise contrastive learning formulation is proposed. The group ordering constraints consider positives and negatives as groups and enforce the group of positives to be closer to the anchor image than the negative group. To enforce these constraints, recent progress in the context of differentiable sorting approaches are leveraged to formulate a group ordering loss based on the given sorting supervision. Our evaluation shows that the proposed framework, does not only compete with current contrastive loss baselines, but actually outperforms standard contrastive learning in many settings with regards to k-NN-based metrics.

Acknowledgements

Nina Shvetsova is supported by German Federal Ministry of Education and Research (BMBF) project STCL - 01IS22067.

References

[1] Hassan Akbari, Liangzhe Yuan, Rui Qian, Wei-Hong Chuang, Shih-Fu Chang, Yin Cui, and Boqing Gong. Vatt:

Transformers for multimodal self-supervised learning from raw video, audio and text. *NeurIPS*, 2021. 3

[2] YM Asano, C Rupprecht, and A Vedaldi. Self-labelling via simultaneous clustering and representation learning. In *ICLR*, 2020. 3

[3] Philip Bachman, R Devon Hjelm, and William Buchwalter. Learning representations by maximizing mutual information across views. *NeurIPS*, 32, 2019. 1

[4] Adrien Bardes, Jean Ponce, and Yann LeCun. Vicreg: Variance-invariance-covariance regularization for self-supervised learning. In *ICLR*, 2022. 3, 4, 6

[5] Mathieu Blondel, Olivier Teboul, Quentin Berthet, and Josip Djolonga. Fast Differentiable Sorting and Ranking. In *ICML*, 2020. 3

[6] Lukas Bossard, Matthieu Guillaumin, and Luc Van Gool. Food-101–mining discriminative components with random forests. In *ECCV*, 2014. 7

[7] Christopher Burges, Robert Ragno, and Quoc Le. Learning to rank with nonsmooth cost functions. *NeurIPS*, 2006. 3

[8] Chris Burges, Tal Shaked, Erin Renshaw, Ari Lazier, Matt Deeds, Nicole Hamilton, and Greg Hullender. Learning to rank using gradient descent. In *ICML*, 2005. 3

[9] Zhe Cao, Tao Qin, Tie-Yan Liu, Ming-Feng Tsai, and Hang Li. Learning to rank: from pairwise approach to listwise approach. In *ICML*, 2007. 3

[10] Mathilde Caron, Piotr Bojanowski, Armand Joulin, and Matthijs Douze. Deep clustering for unsupervised learning of visual features. In *ECCV*, 2018. 3

[11] Mathilde Caron, Ishan Misra, Julien Mairal, Priya Goyal, Piotr Bojanowski, and Armand Joulin. Unsupervised learning of visual features by contrasting cluster assignments. *NeurIPS*, 2020. 1, 3, 6, 7

[12] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *ICCV*, 2021. 3, 6, 7, 8

[13] Andrew N. Carr, Quentin Berthet, Mathieu Blondel, Olivier Teboul, and Neil Zeghidour. Self-supervised learning of audio representations from permutations with differentiable ranking. *IEEE Signal Processing Letters*, 28, 2021. 3

[14] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *ICML*, 2020. 1, 3, 4, 6, 8

[15] Ting Chen, Simon Kornblith, Kevin Swersky, Mohammad Norouzi, and Geoffrey E Hinton. Big self-supervised models are strong semi-supervised learners. *NeurIPS*, 33, 2020. 3, 8

[16] Xinlei Chen, Haoqi Fan, Ross Girshick, and Kaiming He. Improved baselines with momentum contrastive learning. *arXiv preprint arXiv:2003.04297*, 2020. 3, 6

[17] Xinlei Chen and Kaiming He. Exploring simple siamese representation learning. In *CVPR*, 2021. 3, 4, 6

[18] X. Chen, S. Xie, and K. He. An empirical study of training self-supervised vision transformers. In *ICCV*, Los Alamitos, CA, USA, oct 2021. IEEE Computer Society. 8

[19] Mircea Cimpoi, Subhransu Maji, Iasonas Kokkinos, Sammy Mohamed, and Andrea Vedaldi. Describing textures in the wild. In *CVPR*, 2014. 7

[20] Jean-Baptiste Cordonnier, Aravindh Mahendran, Alexey Dosovitskiy, Dirk Weissenborn, Jakob Uszkoreit, and Thomas Unterthiner. Differentiable patch selection for image recognition. In *CVPR*, 2021. 3

[21] Marco Cuturi, Olivier Teboul, and Jean-Philippe Vert. Differentiable ranking and sorting using optimal transport. *NeurIPS*, 2019. 2, 3

[22] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *ICLR*, 2021. 8

[23] Debidatta Dwibedi, Yusuf Aytar, Jonathan Tompson, Pierre Sermanet, and Andrew Zisserman. With a little help from my friends: Nearest-neighbor contrastive learning of visual representations. In *CVPR*, 2021. 1, 3

[24] Aleksandr Ermolov, Aliaksandr Siarohin, Enver Sangineto, and Nicu Sebe. Whitening for self-supervised representation learning. In *ICML*, 2021. 1, 3, 6

[25] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *IJCV*, 88(2):303–338, 2010. 7

[26] Li Fei-Fei, Rob Fergus, and Pietro Perona. Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories. In *CVPR*, 2004. 7

[27] Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Guo, Mohammad Gheshlaghi Azar, et al. Bootstrap your own latent-a new approach to self-supervised learning. *NeurIPS*, 33, 2020. 3, 6

[28] Aditya Grover, Eric Wang, Aaron Zweig, and Stefano Ermon. Stochastic Optimization of Sorting Networks via Continuous Relaxations. In *ICLR*, 2019. 2, 3

[29] Hussein Hazimeh, Zhe Zhao, Aakanksha Chowdhery, Maheswaran Sathiamoorthy, Yihua Chen, Rahul Mazumder, Lichan Hong, and Ed Chi. Dselect-k: Differentiable selection in the mixture of experts with applications to multi-task learning. *NeurIPS*, 34, 2021. 3

[30] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. *CVPR*, 2012. 1, 3

[31] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 6

[32] Tri Huynh, Simon Kornblith, Matthew R Walter, Michael Maire, and Maryam Khademi. Boosting contrastive self-supervised learning with false negative cancellation. In *WACV*, 2022. 3, 4

[33] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, 2015. 6

[34] Donald E. Knuth. *The Art of Computer Programming, Volume 3: Sorting and Searching (2nd Ed.)*. Addison Wesley, 1998. 4

[35] Jonathan Krause, Jia Deng, Michael Stark, and Li Fei-Fei. Collecting a large-scale dataset of fine-grained cars. 2013. 7

[36] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009. 7

[37] Hyunsung Lee, Sangwoo Cho, Yeongjae Jang, Jaekwang Kim, and Honguk Woo. Differentiable ranking metric using relaxed sorting for top-k recommendation. *IEEE Access*, 9, 2021. 3

[38] Ilya Loshchilov and Frank Hutter. SGDR: stochastic gradient descent with warm restarts. In *ICLR*, 2017. 6

[39] Subhransu Maji, Esa Rahtu, Juho Kannala, Matthew Blaschko, and Andrea Vedaldi. Fine-grained visual classification of aircraft. *arXiv preprint arXiv:1306.5151*, 2013. 7

[40] Antoine Miech, Jean-Baptiste Alayrac, Lucas Smaira, Ivan Laptev, Josef Sivic, and Andrew Zisserman. End-to-End Learning of Visual Representations from Uncurated Instructional Videos. In *CVPR*, 2020. 1

[41] Maria-Elena Nilsback and Andrew Zisserman. Automated flower classification over a large number of classes. In *2008 Sixth Indian Conference on Computer Vision, Graphics & Image Processing*, pages 722–729. IEEE, 2008. 7

[42] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018. 8

[43] Omkar M Parkhi, Andrea Vedaldi, Andrew Zisserman, and CV Jawahar. Cats and dogs. In *CVPR*. IEEE, 2012. 7

[44] Felix Petersen. *Learning with Differentiable Algorithms*. PhD thesis, Universität Konstanz, 2022. 2, 3

[45] Felix Petersen, Christian Borgelt, Hilde Kuehne, and Oliver Deussen. Differentiable Sorting Networks for Scalable Sorting and Ranking Supervision. In *ICML*, 2021. 2, 3, 4

[46] Felix Petersen, Christian Borgelt, Hilde Kuehne, and Oliver Deussen. Learning with algorithmic supervision via continuous relaxations. *NeurIPS*, 2021. 2, 3

[47] Felix Petersen, Christian Borgelt, Hilde Kuehne, and Oliver Deussen. Monotonic differentiable sorting networks. *ICLR*, 2022. 2, 3, 4

[48] Felix Petersen, Hilde Kuehne, Christian Borgelt, and Oliver Deussen. Differentiable top-k classification learning. In *ICML*, 2022. 2, 3

[49] Sebastian Prillo and Julian Eisenschlos. Softsort: A continuous relaxation for the argsort operator. In *ICML*, 2020. 3

[50] Filip Radenović, Ahmet Iscen, Giorgos Tolias, Yannis Avrithis, and Ondřej Chum. Revisiting oxford and paris: Large-scale image retrieval benchmarking. In *CVPR*, 2018. 6

[51] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *ICML*, 2021. 3

[52] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. In *ICML*, 2021. 1

[53] Joshua David Robinson, Ching-Yao Chuang, Suvrit Sra, and Stefanie Jegelka. Contrastive learning with hard negative samples. In *ICLR*, 2021. 2, 3

[54] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *IJCV*, 115(3), 2015. 6

[55] Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In *CVPR*, 2015. 3

[56] Christoph Schuhmann, Romain Beaumont, Richard Vencu, Cade W Gordon, Ross Wightman, Mehdi Cherti, Theo Coombes, Aarush Katta, Clayton Mullis, Mitchell Wortsman, Patrick Schramowski, Srivatsa R Kundurthy, Katherine Crowson, Ludwig Schmidt, Robert Kaczmarczyk, and Jenia Jitsev. LAION-5b: An open large-scale dataset for training next generation image-text models. In *NeurIPS Datasets and Benchmarks Track*, 2022. 1

[57] Anshul Shah, Suvrit Sra, Rama Chellappa, and Anoop Cherian. Max-margin contrastive learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2022. 2, 6

[58] Robin Swezey, Aditya Grover, Bruno Charron, and Stefano Ermon. Pirank: Scalable learning to rank via differentiable sorting. *NeurIPS*, 2021. 3

[59] Michael Taylor, John Guiver, Stephen Robertson, and Tom Minka. Softrank: optimizing non-smooth rank metrics. In *Proceedings of the 2008 International Conference on Web Search and Data Mining*, pages 77–86, 2008. 3

[60] Yonglong Tian, Chen Sun, Ben Poole, Dilip Krishnan, Cordelia Schmid, and Phillip Isola. What makes for good views for contrastive learning? *NeurIPS*, 2020. 1

[61] Jianxiong Xiao, James Hays, Krista A Ehinger, Aude Oliva, and Antonio Torralba. Sun database: Large-scale scene recognition from abbey to zoo. In *CVPR*, 2010. 7

[62] Tete Xiao, Xiaolong Wang, Alexei A Efros, and Trevor Darrell. What should not be contrastive in contrastive learning. In *ICLR*, 2021. 3

[63] Mang Ye, Xu Zhang, Pong C Yuen, and Shih-Fu Chang. Unsupervised embedding learning via invariant and spreading instance feature. In *CVPR*, 2019. 1

[64] Yang You, Igor Gitman, and Boris Ginsburg. Large batch training of convolutional networks. *arXiv preprint arXiv:1708.03888*, 2017. 6

[65] Jure Zbontar, Li Jing, Ishan Misra, Yann LeCun, and Stéphane Deny. Barlow twins: Self-supervised learning via redundancy reduction. In *ICML*, 2021. 3, 4, 6

[66] Fangneng Zhan, Yingchen Yu, Rongliang Wu, Kaiwen Cui, Aoran Xiao, Shijian Lu, and Ling Shao. Bi-level feature alignment for versatile image translation and manipulation. In *ECCV*, 2021. 3

[67] Mingkai Zheng, Shan You, Fei Wang, Chen Qian, Changshui Zhang, Xiaogang Wang, and Chang Xu. Ressl: Relational self-supervised learning with weak augmentation. *NeurIPS*, 2021. 3, 6