Supplementary Material
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}

In the supplementary material, we first discuss relations between the GroCo loss, the contrastive loss, and the triplet loss in Section A. Then, we provide additional experimental evaluation results in Section B and a qualitative analysis in Section C. In Section D, we describe odd-even sorting networks. Finally, we cover additional implementation details in Section E.

A. Discussion of GroCo / Contrastive / Triplet Loss Relations

In this section, we discuss the similarities and differences between the GroCo loss, the contrastive loss, and the triplet loss. For comparison purposes, let's consider a simplified version of losses when there is only one positive example x^p and one negative example x^n for the anchor x^a. We denote the distance from the anchor x^a to the positive sample x^p as $d^p = -\frac{{x^a}^\top x^p}{\|x^a\|\|x^p\|}$ and the distance from the anchor x^a to the negative sample x^n as $d^n = -\frac{{x^a}^\top x^n}{\|x^a\|\|x^n\|}$. Then contrastive InfoNCE loss (with respect to the anchor x^a) is defined as:

$$\begin{aligned} L_{Contrastive} &= -\log \frac{\exp(-d^p/\tau)}{\exp(-d^p/\tau) + \exp(-d^n/\tau)} \\ &= \log\left(1 + \exp(-(d^n - d^p)/\tau)\right) \end{aligned} \tag{1}$$

where τ is a temperature hyperparameter (Figure A.1a).

The triplet loss is defined as:

$$\begin{aligned} L_{Triplet} &= \max\left(d^p - d^n + r, 0\right) = \\ &= \max\left(r - (d^n - d^p), 0\right) \end{aligned} \tag{2}$$

where r is a margin hyperparameter (Figure A.1b).

For the GroCo loss, a permutation matrix $P \in R^{2\times 2}$ corresponds to only one conditional swap operation and is defined as:

$$\begin{aligned} P_{11} = P_{22} = f(d^n - d^p) = \frac{1}{\pi}\arctan(\beta(d^n - d^p)) + 0.5\,, \\ P_{12} = P_{21} = f(d^p - d^n) = \frac{1}{\pi}\arctan(\beta(d^p - d^n)) + 0.5 \end{aligned} \tag{3}$$

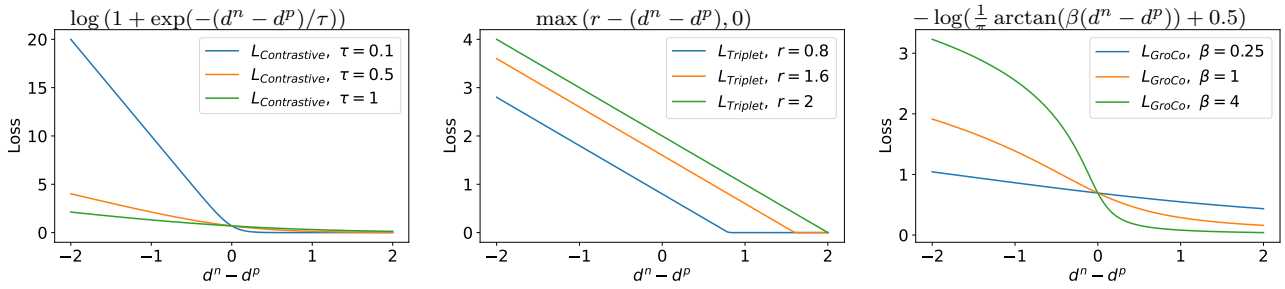
where β is an inverse temperature. Therefore, the GroCo loss is defined as:

$$\begin{aligned} L_{GroCo} = \frac{1}{4}\Bigg(&-2\log\Big(\frac{1}{\pi}\arctan(\beta(d^n - d^p)) + 0.5\Big) - \\ &-2\log\Big(1 - \frac{1}{\pi}\arctan(\beta(d^p - d^n)) - 0.5\Big)\Bigg) = \\ &= -\log\Big(\frac{1}{\pi}\arctan(\beta(d^n - d^p)) + 0.5\Big) \end{aligned} \tag{4}$$

where β is an inverse temperature hyperparameter (Figure A.1c).

In Figure A.1, we show the loss curves with different values of respective hyperparameters. We note that in this simplified example with only one positive and only one negative, all three losses try to maximize the difference between the distances to the positive and negative examples $(d^n - d^p)$. The temperature τ, the margin r, or the inverse temperature β define the flatness of the loss curve depending on the difference $(d^n - d^p)$.

However, in the case with more negative/positive examples for the anchor image, different losses integrate information from multiple negatives/positives in different ways. For the triplet loss, there are various strategies to sample one positive example and one negative example for the anchor image [7, 8]. The complete loss is defined as the sum (or average) of the losses for the chosen triplets $\sum_{ij}\max\left(r - (d_i^n - d_j^p), 0\right)$. On the other hand, the contrastive loss aggregates multiple negatives by contrasting the positive example to all negative examples, resulting in sum under logarithm: $\log\left(1 + \sum_i \exp(-(d_i^n - d^p)/\tau)\right)$. In contrast to an explicit sum over a predefined number of negatives, the GroCo loss aggregates multiple positives and negatives via the permutation matrix, conditionally swapping neighboring elements, and later applies the group ordering supervision, enforcing the distance between positive and negative groups.



(a) Contrastive InfoNCE loss (b) Triplet loss (c) GroCo loss

Figure A.1. Comparison of the contrastive loss, the triplet loss, and the GroCo loss in a simple scenario with only one positive example and one negative example for an anchor image. We denote the distance from the anchor to the positive sample as d^p and the distance from the anchor to the negative sample as d^n. We note that, in the simple case of only one positive and one negative, all three losses try to maximize the difference between the distances to the positive and negative examples $(d^n - d^p)$. The temperature τ, the margin r, or the inverse temperature β define the flatness of the loss curve depending on the difference $(d^n - d^p)$.

B. Additional Experimental Results

In this section, we provide additional experimental results:

Top negatives, stop gradient operation. We further analyze if using the top-10 strongest negatives and the stop gradient operation (stop-grad) can also boost the performance of the considered contrastive learning baseline SimCLR. In our method, the stop gradient operation stabilizes training (fewer spikes in gradients) and allows for convergence with large variations of hyperparameters. In Table B.1a, we observe that stop gradient does not boost SimCLR performance. However, if we utilize only the top-10 strongest negatives in the loss, it stabilizes SimCLR training. Moreover, usage only top-10 negatives indeed boosts SimCLR performance on $+3.1\%$ in k-NN ($k = 20$) and $+0.3\%$ in linear probing (Top-1), but the SimCLR still significantly underperforms the proposed GroCo method by 5.5% in k-NN ($k = 20$) and $+3.2\%$ in linear probing (Top-1).

Longer training. We further assess the performance of our model in a longer training regime of 800 epochs (Table B.1b). We find that the proposed model with a multi-crop augmentation strategy achieves 66.1% in k-NN ($k = 20$) and 73.9% in linear probing (Top-1).

Augmentation Strategy. In Table B.1c we evaluate the performance of the model with respect to different augmentation strategies for view sampling. We follow two setups: (1) the augmentation strategy as used in the SimCLR [3] method with a random resized crop, color jittering, and gaussian blur, grayscaling and horizontal flip and (2) the augmentation strategy as used in the DINO [2] method that extends the SimCLR list of augmentations with solarization. SimCLR augmentations are considered as "stronger" compared to DINO augmentations since they include a larger range of cropping sizes ($8\% - 100\%$ of original image compared to $14\% - 100\%$ in DINO augmentations) and larger range values in color jittering. We observe that the stronger SimCLR [3] augmentations are more beneficial for the SimCLR method than the weaker DINO augmentations, while for the proposed method, the DINO augmentation strategy is more beneficial. However, the difference between augmentation strategies diminishes with increasing number of training epochs and is no longer measurable at 400 epochs. For a fair comparison, we use the SimCLR augmentation strategy in all reproductions of the SimCLR method reported in the main paper.

Projection Dimensionality. We also ablate our method with respect to dimensionality of the projection space (or the latent space), where distances between samples are computed to calculate a training loss. Table B.1d shows that increasing dimensionality of the projection space increases performance in general, which is more noticeable for the k-NN performance. Note that we do not change the dimensionality of the embedding space (output space of the encoder that is used for the k-NN evaluation and linear evaluation), which is always 2048-dimensional.

Importance of Negatives. We also evaluate the importance of utilizing strong negatives for the successful training of our model. We train the model using ten random negatives instead of the top-10 strongest negatives as a negative group and report performance in Table B.1e. We observe that leveraging the strongest negatives increases performance across all metrics, demonstrating the importance of hard negatives during training with the GroCo loss, similarly as the contrastive loss benefits from hard negative sampling [6].

k-NN in Projection Space. We also evaluate the k-NN performance in the projection space (or the latent space) where the training loss is applied. We compare k-NN performance in the projection and representation spaces in Table B.1f. We observe that for both methods, k-NN performance is

	k-NN Evaluation			Linear Probing	
	k=1	k=10	k=20	Top-1	Top-5
InfoNCE (= SimCLR method)	46.0	51.5	51.9	65.7	86.7
InfoNCE + s. grad.	46.1	51.3	51.8	65.6	86.7
InfoNCE + top 10 neg.	unstable training				
InfoNCE + top 10 neg. + s. grad.	49.5	54.6	55.0	66.0	86.7
GroCo + top 10 neg.	unstable training				
GroCo + top 10 neg. + s. grad.	**55.3**	**60.3**	**60.5**	**69.2**	**88.4**

(a) **Usage of top-10 negatives, stop gradient operation**

Method	Views	Epochs	k-NN Evaluation			Linear Eval.	
			k=1	k=10	k=20	Top-1	Top-5
GroCo	2×224	800	59.9	65.0	65.3	71.2	89.9
GroCo	2×224+6×96	800	60.8	65.7	66.1	73.9	91.6

(b) **Longer training**

Method	Augmentations	Epochs	k-NN Evaluation			Linear Evaluation	
			k=1	k=10	k=20	Top-1	Top-5
SimCLR	as in SimCLR [2]	100	**46.0**	**51.5**	**51.9**	**65.7**	**86.7**
SimCLR	as in DINO [2]	100	43.3	48.6	49.1	63.7	85.4
GroCo	as in SimCLR [2]	100	54.0	59.0	59.4	68.4	88.3
GroCo	as in DINO [2]	100	**55.3**	**60.3**	**60.5**	**69.2**	**88.4**
GroCo	as in SimCLR [2]	200	56.7	61.6	61.8	69.8	89.1
GroCo	as in DINO [2]	200	**57.7**	**62.4**	**62.7**	**70.4**	**89.5**
GroCo	as in SimCLR [2]	400	58.3	63.3	63.8	71.1	89.7
GroCo	as in DINO [2]	400	58.7	63.4	63.6	71.0	89.7

(c) **Augmentation strategy**

Projection dim	Embedding dim	k-NN Evaluation			Linear Probing	
		k=1	k=10	k=20	Top-1	Top-5
128	2048	53.7	58.5	58.7	68.1	88.0
512	2048	55.2	59.8	60.1	69.0	**88.5**
2048	2048	**55.3**	**60.3**	**60.5**	**69.2**	88.4

(d) **Projection dimentionality**

	k-NN Evaluation			Linear Probing	
	k=1	k=10	k=20	Top-1	Top-5
10 random negatives	39.5	45.0	45.3	60.1	82.7
top-10 strongest negatives	**55.3**	**60.3**	**60.5**	**69.2**	**88.4**

(e) **Importance of negatives**

Method	Space	k-NN Evaluation		
		k=1	k=10	k=20
SimCLR	Projection Space	35.8	41.6	42.3
SimCLR	Representation Space	**46.0**	**51.5**	**51.9**
GroCo	Projection Space	51.4	56.9	57.3
GroCo	Representation Space	**55.3**	**60.3**	**60.5**

(f) **k-NN evaluation**

Table B.1. **Additional Experiments.** The best results are bolded. Options used to obtain the main results are highlighted. **Backbone=Resnet50, Views=2×224, #epochs=100.**

higher if we use embeddings from the representation space even though we train the model to compare embeddings in the projection space. This could be explained by the fact that the embedding space contains more general image representations since the representations in projection space could be overfitted to the respective augmentations and there become agnostic to some image attributes (like color, since we train the model to match views with different color jittering parameters).

C. Qualitative Analysis of Learned Representation Space

We additionally perform a qualitative analysis of the learned representations. In Figure C.1, we visualize representations for images from four classes of different types of cats and four classes of different types of dogs. We find that our method produces much more visually separable clusters with respect to "inter-class" variations (cats vs dogs) and "intra-class" variations (between different classes of cats) than the SimCLR baseline.

D. Odd-even Sorting Network

An odd-even sorting network, or odd-even sort, is a sorting algorithm from classic computer science literature [5]. Sorting networks, or networks for sorting, are a family of sorting algorithms that consist of the *fixed* sequence of comparisons, in a sense that the next comparisons (elements on which positions are compared) does not depend on the result of previous comparisons. An odd-even sorting network is a simple example of this family of algorithms. The odd-even sorting network compares neighbored elements starting from odd and even indices alternating on each step, and requires n steps to sort a sequence of n elements. We present a pseudocode of the odd-even sorting network in Algorithm 1. Additionally, we illustrate the (hard) odd-even sorting process in Figure D.1.

Algorithm 1 Python–style pseudocode of an odd-even sorting network for sorting an array of numbers in non-descending order

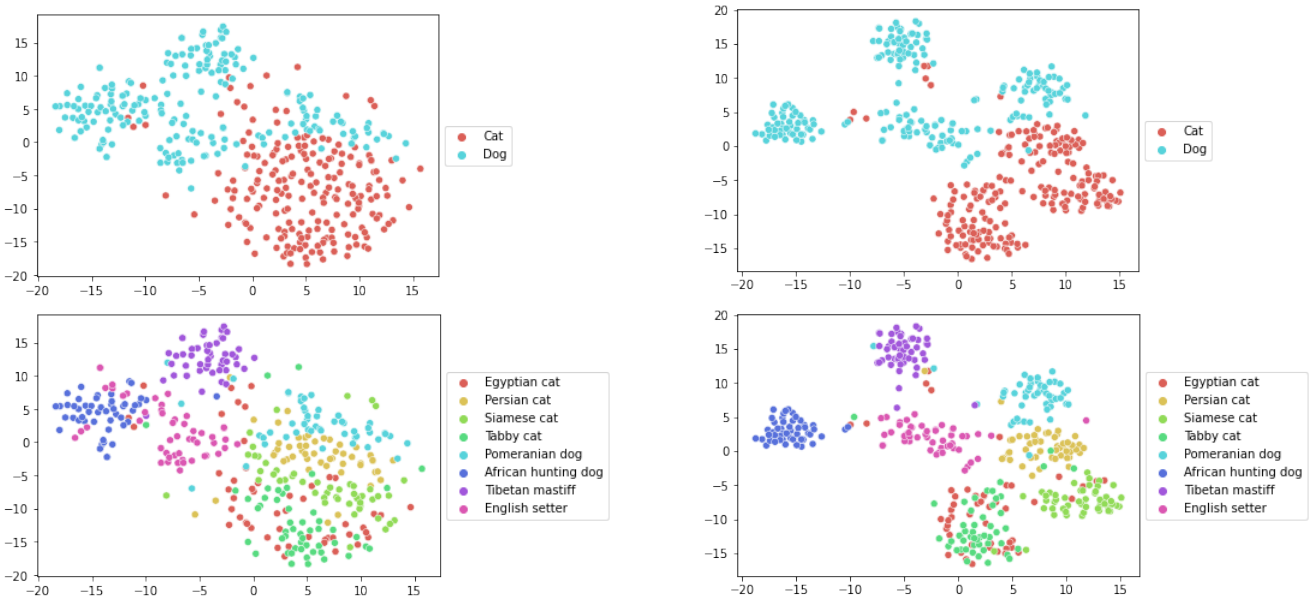
```
# arr: array to sort
# n: length of array

for s in range(1, n + 1):
    if s % 2 == 1:
        for i in range(0, n - 1, 2):
            if arr[i] > arr[i+1]:
                arr[i], arr[i+1] = arr[i+1], arr[i]
    else:
        for i in range(1, n - 1, 2):
            if arr[i] > arr[i+1]:
                arr[i], arr[i+1] = arr[i+1], arr[i]
```

E. Implementation Details

E.1. Linear Evaluation Details

For linear evaluation, we train a linear classifier on frozen representations in a fully-supervised way, using the training set of ImageNet for training and the validation set for evaluation. We follow the training protocol of SimCLR [3] and SimSiam [4] and train a linear classifier for 90



(a) SimCLR (b) GroCo (ours)

Figure C.1. t-SNE visualization of learned representations of Imagenet validation images from four classes of different types of cats (Egyptian cat, Persian cat, Siamese cat, Tabby cat) and four classes different types of dogs (Pomeranian dog, African hunting dog, Tibetan mastiff, English setter) for the SimCLR method and the proposed method. For visualization we use models with Resnet50 encoder trained for 100 epochs with a batch size of 1024 and 2×224 views.

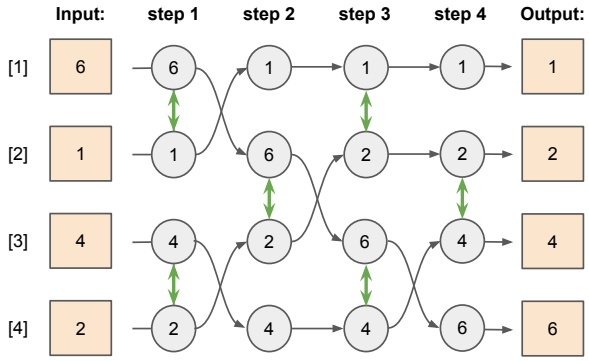


Figure D.1. An illustration of a hard odd-even sorting network for sorting four elements in non-descending order with an example of sorting of [6, 1, 4, 2] array.

epochs using the LARS optimizer [9] with the batch size of 4 096, the momentum of 0.9, the linear rate of 1.6 (following the rule: learning rate $= 0.1 \times$ batch size$/256$), without a warmup and weight decay. Following [3] and [4], we use weak data augmentation (only random cropping with horizontal flipping) and apply gradient stopping on the input of the classifier to prevent updating the encoder.

E.2. SimCLR with Multiple Positives

To train SimCLR with more than one positive view per anchor, we apply contrastive loss for all possible positive pairs, considering all views from other images in the batch as negatives (with a batch of B examples with have $m(B-1)$ negatives views). Let x_i^b denote the i'th view of the b'th image in a batch, and $P_{x_i^b}$ denote a set of positive samples for the anchor x_i^b, and $N_{x_i^b}$ denote a set of positive samples for the anchor x_i^b. Then, the loss is calculated as

$$\mathcal{L}_{\text{SimCLR}} = \frac{1}{B}\sum_{b=1}^{B}\frac{1}{m}\sum_{i=1}^{m}\frac{1}{\left|P_{x_i^b}\right|}\sum_{y\in P_{x_i^b}} -\log\left(\frac{\exp(-\mathrm{d}(x_i^b,y)/\tau)}{\exp(-\mathrm{d}(x_i^b,y)/\tau)+\sum_{z\in N_{x_i^b}}\exp(-\mathrm{d}(x_i^b,z)/\tau)}\right) \tag{5}$$

where τ is a temperature parameter. This extension of the SimCLR framework for $m > 2$ views per image is the same as used in the SwAV evaluations [1]. Note that in the multi-crop scenario, we use only full-resolution global views as positive examples following "local-global" correspondence idea [1, 2].

References

[1] Mathilde Caron, Ishan Misra, Julien Mairal, Priya Goyal, Piotr Bojanowski, and Armand Joulin. Unsupervised learning of visual features by contrasting cluster assignments. *NeurIPS*, 2020. 4

[2] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerg-

ing properties in self-supervised vision transformers. In *ICCV*, 2021. 2, 3, 4

[3] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *ICML*, 2020. 2, 3, 4

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

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

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

[7] Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In *CVPR*, 2015. 1

[8] Hong Xuan, Abby Stylianou, Xiaotong Liu, and Robert Pless. Hard negative examples are hard, but useful. In *ECCV*, 2020. 1

[9] Yang You, Igor Gitman, and Boris Ginsburg. Large batch training of convolutional networks. *arXiv preprint arXiv:1708.03888*, 2017. 4