RankMe: Assessing the Downstream Performance of Pretrained Self-Supervised Representations by Their Rank

Anonymous authors
Paper under double-blind review

Abstract

Joint-Embedding Self Supervised Learning (JE-SSL) has seen a rapid development, with the emergence of many method variations and few principled guidelines that would help practitioners to successfully deploy those methods. The main reason for that pitfall actually comes from JE-SSL's core principle of not employing any input reconstruction. Without any visual clue, it becomes extremely cryptic to judge the quality of a learned representation without having access to a labelled dataset. We hope to correct those limitations by providing a single –theoretically motivated– criterion that reflects the quality of learned JE-SSL representations: their effective rank. Albeit simple and computationally friendly, this method —coined *RankMe*— allows one to assess the performance of JE-SSL representations, even on different downstream datasets, without requiring any labels, training or parameters to tune. Through thorough empirical experiments involving hundreds of repeated training episodes, we demonstrate how RankMe can be used for hyperparameter selection with nearly no loss in final performance compared to the current selection method that involve dataset labels. We hope that RankMe will facilitate the use of JE-SSL in domains with little or no labeled data.

1 Introduction

Self-supervised learning (SSL) has shown great progress to learn informative data representations in recent years (Chen et al., 2020a; He et al., 2020; Chen et al., 2020b; Grill et al., 2020; Lee et al., 2021; Caron et al., 2020; Zbontar et al., 2021; Bardes et al., 2021; Tomasev et al., 2022; Caron et al., 2021; Chen et al., 2021; Li et al., 2022a; Zhou et al., 2022a;b; HaoChen et al., 2021; He et al., 2022), catching up to supervised baselines and even surpassing them in few-shot learning, i.e. when evaluating the SSL model from only a few labeled examples. Although various SSL families of losses have emerged, most are variants of the joint-embedding (JE) framework with a siamese network architecture (Bromley et al., 1994), denoted as JE-SSL for short. The only technicality we ought to introduce to make our study precise is the fact that JE-SSL has introduced some different notations to denote an input's representation. In short, JE-SSL often composes a *backbone* or *encoder* network e.g. a ResNet50 and a *projector* network e.g. a multilayer perceptron. This projector is only employed during training, and we refer to its outputs as *embeddings*, while the actual inputs' *representation* employed for downstream tasks are obtained at the encoder's output.

Although downstream tasks performance of JE-SSL representations might seem impressive, one pondering fact should be noted: *all existing methods, hyper-parameters, models — and thus performance — of JE-SSL are obtained by ad-hoc manual search involving the labels of the training samples*. In words, JE-SSL is tuned by monitoring the supervised performance of the model at hand. Hence, although labels are not directly employed to compute the weight updates, they are used as a proxy to signal the JE-SSL designer on how to refine their method. This single limitation prevents the deployment of JE-SSL in challenging domains where the number of available labelled examples is limited and such search can not be performed. Adding to the challenge, one milestone of JE-SSL is to move away from reconstruction based learning; hence without labels and without visual cues, tuning JE-SSL methods on unlabeled datasets remain challenging. This led to the application of feature inversion methods e.g. Deep Image Prior (Ulyanov et al., 2018) or conditional diffusion model (Bordes et al., 2021) to be deployed onto learned JE-SSL representation to try to visualize the learned features. This first step towards removing the need for labels has seen some success but is doomed by the computational complexity of the methods, and their biases towards natural images i.e. it is not clear how such methods would perform in different data modalities.

In this study, we propose a first attempt at making JE-SSL truly unsupervised by providing the first method —coined RankMe— able to assess a model's performance without having access to any label and without requiring any training or tuning. RankMe accurately predicts a model's performance both on In-Distribution (ID), i.e. same data distribution as used during the JE-SSL training, and on Out-Of-Distribution (OOD), i.e. different data distribution scenarios. We highlight this property at the top of fig. 1. The strength of RankMe lies in the fact that it is solely based on the singular values distribution of the learned representation, and thus does not rely on any parameters that need training, nor requires any ID/OOD labels. In fact, RankMe's construction hinges on Cover's theorem (Cover, 1965) that states how increasing the rank of a linear classifier's input increases its training performance, and three simple hypotheses that we summarize below and thoroughly validate empirically. As such, RankMe provides a significant step towards (unlabeled) JE-SSL by allowing practitioners to cross-validate hyper-parameters and select models without resorting to labels or feature inversion methods. We hope that RankMe will enable JE-SSL to be deployed even in challenging domains that possess no or little labelled data; we summarize our contributions below:

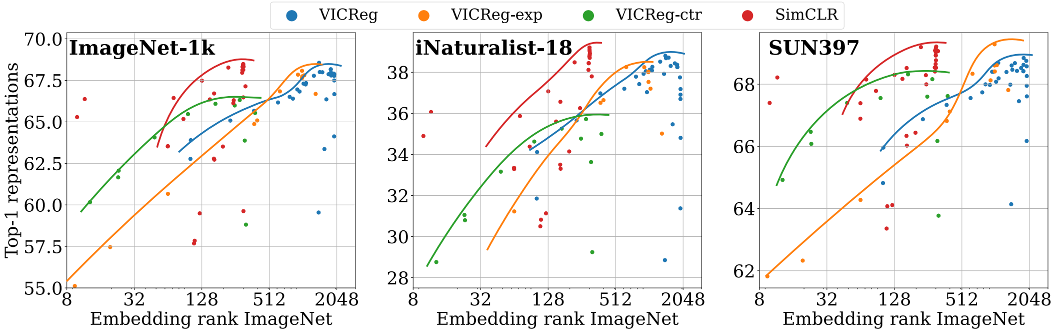
1. We introduce (eq. (1)) and theoretically motivate RankMe which combines Cover's theorem with the following three key hypotheses:
   (H1) **increasing training performance increases testing performance** on both representations and embeddings i.e. no over-fitting is observed from the (non)linear probe, validated in bottom left of fig. 2
   (H2) **embeddings' rank scale linearly between datasets.** Assuming a pretraining on the same dataset, if a set of embeddings have a greater rank than another on one dataset, it also holds on another one, validated in top row of fig. 2
   (H3) **increasing embeddings performance increases representations performance**, validated in bottom right of fig. 2

   concluding that embeddings with greater rank will have greater train performance (Cover's theorem) and test performance (H1) on ID and OOD cases (H2) and even before the projector (H3).
2. We demonstrate that RankMe's ability to assess JE-SSL downstream performance is robust across methods e.g. VICReg, SimCLR and their variants, and is also robust to architecture changes e.g. using a projector network and/or a nonlinear evaluation method (see fig. 3 and section 3.3).
3. We demonstrate that RankMe enables hyper-parameter cross-validation for any given JE-SSL method; RankMe is able to retrieve and sometimes surpass most of the performance previously found by manual search using labels, on both in domain and out of domain datasets; see bottom of fig. 1 and table 1.

We provide an hyper-parameter free numerically stable implementation of RankMe in section 3.1 and pseudo-code for cross-validation in fig. 5. Through extensive experiments involving 6 different datasets and more than 85 trained models over 4 methods, we demonstrate that in the linear and nonlinear probing regime, RankMe is able to tell apart high and low performing models, even on different downstream tasks without having access to labels or downstream task data samples.

2 Related works

Joint embedding self-supervised learning (JE-SSL). In JE-SSL, two main families of method can be distinguished: contrastive and non-contrastive. Contrastive methods (Chen et al., 2020a; He et al., 2020; Chen et al., 2020b; 2021; Yeh et al., 2021) mostly rely on the InfoNCE criterion (Oord et al., 2018) except for HaoChen et al. (2021) which uses squared similarities between the embedding. A clustering variant of contrastive learning has also emerged (Caron et al., 2018; 2020; 2021) and can be thought of as contrastive methods but between cluster centroids instead of samples. Non-contrastive methods (Grill et al., 2020; Chen & He, 2020; Caron et al., 2021; Bardes et al., 2021; Zbontar et al., 2021; Ermolov et al., 2021; Li et al., 2022b) aim at bringing together embeddings of positive samples, similar to contrastive learning. However, a key difference with contrastive learning lies in how those methods prevent a representational collapse. In the former, the criterion explicitly pushes away negative samples i.e., all samples that are not positive, from each other. In the latter, the criterion



Dataset	Method	VICReg				SimCLR		
		cov.	inv.	LR	WD	temp.	LR.	WD.
ImageNet	ImageNet Oracle	68.2	68.2	68.6	68.0	68.5	68.5	68.3
	RankMe	67.8	67.9	68.2	67.8	67.1	68.0	68.3
	Difference	−0.4	−0.3	−0.4	−0.2	−1.4	−0.5	=
OOD	ImageNet Oracle	62.0	62.0	62.2	62.0	62.3	62.3	62.4
	RankMe	60.7	61.4	62.0	62.0	61.4	61.8	62.4
	Difference	−1.3	−0.6	−0.2	=	−0.9	−0.5	=

Figure 1: **(Top)** Performance of JE-SSL representations (encoder output) in **y-axis** against the embeddings (projector output) RankMe values in **x-axis** on ImageNet-1k. Except for some degenerate solutions at full-rank, RankMe values correlate well with in-distribution (**left column**) and out-of-distribution (**right columns**) classification performance. **(Bottom)** Hyperparameter selection using the common supervised linear probe strategy and the proposed unsupervised RankMe strategy. OOD indicates the average performance over all the considered dataset other than ImageNet. Without any label, optimization or parameters, RankMe is able to recover most of the performance obtained by using ImageNet validation set, highlighting its strength as a hyperparameter selwction tool.

does not prevent collapse by distinguishing positive and negative samples, but instead considers the embeddings as a whole and encourage information content maximization e.g. by regularizing the empirical covariance matrix of the embeddings. Such a categorization is not needed for our development and we thus refer to any of the above method as JE-SSL. For completeness, we provide in appendix A the precise definitions of some JE-SSL methods that are VICReg, SimCLR, and their variants which we focus on.

Dimensional collapse in JE-SSL. The phenomenon of learning rank-deficient embeddings, or dimensional collapse, in JE-SSL has recently been studied from both a theoretical and empirical point of view. The emprivial emergence of dimensional collapse was studied in Hua et al. (2021) where they proposed the use of a whitening batch normalization layer to help alleviate it. In Jing et al. (2022), a focus on contrastive approaches in a linear setting enabled a better understanding of dimensional collapse and the role of augmentations in its emergence. Performance in a low label regime of a partially collapsed encoder can also be improved by forcing the whitening of its output, as shown in He & Ozay (2022). Furthermore, it was shown in Balestriero & LeCun (2022) how dimensional collapse is a phenomenon that should not necessarily happen in theory and how its emergence is mostly due to practical concerns. Interestingly, we will see through the lens of RankMe that while reducing dimensional collapse is often beneficial, doing so “at all cost” can lead to degenerate solutions. The collapse induced from training using a softmax layer was also studied in Ganea et al. (2019), where they show that high rank embeddings are desirable.

Evaluation of JE-SSL representations. Evaluating the representations learned by JE-SSL methods is fundamental to enable the optimal selection of those methods' hyper-parameters, which are numerous. Yet, due to the imprecise nature of what makes a good representation, multiple strategies have emerged which evaluate different properties of representations. The most common approach relies on the *strong assumption* of having labels on the dataset where the JE-SSL method is trained on. In this case, on trains a linear classifier on the JE-SSL representations (Misra & Maaten, 2020) and directly use the test accuracy to compare models. This method was extended to the use of nonlinear classifiers e.g. a k-nn classifier (Wu et al., 2018; Zhuang et al., 2019). Performance evaluation without labels can also be done using a pretext-task, such as rotation prediction. This technique helped in selecting data augmentation policies in Reed et al. (2021). One limitation lies in the need to select and train the classifier of the pretext-task, and on the strong assumption that rotation were not part of the transformations one aimed to be invariant to. Since (supervised) linear evaluation is the most widely used evaluation method, we will focus on showing how RankMe compares with it.

3 Representations' Rank Correlate with Downstream performance Across Tasks and Models

The goal of this section is to formally introduce and motivate RankMe while providing a numerically stable implementation (section 3.1). The construction of RankMe hinges on three hypotheses that we validate empirically throughout this section.

3.1 RankMe: From Theory to Implementation

We first want build notations and intuition into the construction of RankMe. To that hand, we first quantify approximation and classification errors of learned embeddings as a function of their rank, and then motivate how embeddings' rank is indeed sufficient to compare test performance of JE-SSL models's representations. This criterion should however only be used to compare different runs of a given method, since embedding rank is not the only factor that affects performance. To ease notations, we refer to the (train) dataset used to obtain the JE-SSL model as the *source dataset*, and the test set on the same dataset or a different OOD dataset as the *target dataset*.

From Source Embeddings's Rank to Target Representations performance. We first build some intuition in the regression settings. In this case, a common linear algebra result ties the best-case and worst-case approximation error of any target matrix $\boldsymbol{Y} \in \mathbb{R}^{N \times C}$ from a rank-R matrix $\boldsymbol{P} \in \mathbb{R}^{N \times C}$ to the singular values of $\boldsymbol{Y}$ that run from R to the rank of $\boldsymbol{Y}$ when ordered in decreasing order. Without loss of generality, we only consider the case $C > N$ in this study i.e. we have more samples than dimensions. Formally, this provides a lower bound on

$$\|\boldsymbol{Y} - \boldsymbol{P}\|_F^2 \geq \sum_{r=R+1}^{C} \sigma_r^2(\boldsymbol{Y}),$$

which is tight for $\boldsymbol{P}$ of rank R, and with σ_k the operator returning the k^{th} singular value of its argument, ordered in decreasing order. This result, on which RankMe relies on, demonstrates that a necessary (but not sufficient) condition for an approximation $\boldsymbol{P}$ to well approximate $\boldsymbol{Y}$ is to have at least the same rank as $\boldsymbol{Y}$. A similar result can be obtained in classification by considering multiple one-vs-all classifiers. In practice however, we commonly employ a linear probe network on top of given embeddings $\boldsymbol{Z}$ to best adapt them to the target $\boldsymbol{Y}$ i.e., $\boldsymbol{P} = \boldsymbol{Z}\boldsymbol{W} + \boldsymbol{1}\boldsymbol{b}^T$. However, a linear transformation is not able to increase the rank of the input matrix since

$$\text{rank}(\boldsymbol{P}) \leq \min(\text{rank}(\boldsymbol{Z}), \text{rank}(\boldsymbol{W})) + 1.$$

We directly obtain that $\min_{\boldsymbol{W},\boldsymbol{b}} \|\boldsymbol{Y} - \boldsymbol{Z}\boldsymbol{W} - \boldsymbol{1}\boldsymbol{b}^T\|_F^2 \geq \sum_{r=R+1}^{C} \sigma_r^2(\boldsymbol{Y})$. In short, the approximation lower bound is not improved by allowing linear transformation of the embeddings. Further supporting the above, we ought to recall Cover's theorem (Cover, 1965) stating that the probability of a randomly labeled set of points being linearly separable only increases if N is reduced or R is increased. We formalize those results below.

Proposition 1. *The maximum training accuracy of given embeddings in linear regression or classification increases with their rank. For classification, it plateaus when the rank surpasses the number of classes.*

We thus introduce RankMe formally as the following smooth rank measure, originally introduced in Roy & Vetterli (2007),

$$\text{RankMe}(\boldsymbol{Z}) = \exp\left(-\sum_{k=1}^{\min(N,K)} p_k \log p_k\right),\ p_k = \frac{\sigma_k(\boldsymbol{Z})}{\|\sigma(\boldsymbol{Z})\|_1} + \epsilon, \tag{1}$$

where $\boldsymbol{Z}$ is the source dataset's embeddings. By noticing that RankMe provides a smooth measure of the embeddings' rank (more details in the implementation section) we can lean on proposition 1 to see that given two models, the one with greater RankMe value will have greater training performance. This is only guaranteed for different models of the same method, since embedding rank is not necessarily the only factor that affects performance.

The above result is however not too practical yet since what we are truly interested in are (i) performance on unseen samples i.e. on the test set and out-of-distribution tasks, and (ii) performance on the representations and not the embeddings since it is common to ablate the projector network of JE-SSL models. Below, we validate three key hypotheses which, when verified, imply that we can extend the impact of RankMe such that *(OOD) test performance of JE-SSL representations are increased when RankMe's value on their train set embeddings is increased.*

Validating RankMe's Hypotheses The development of RankMe is theoretically grounded when it comes to guaranteeing improved source dataset embeddings performance. To extend it to target dataset representations performance we need to verify three hypotheses: (i) linear probes do not overfit, (ii) embeddings and representations performance are monotonically linked, and (iii) source and (OOD) target embeddings ranks are monitinically linked. Due to the different nature of datasets used for downstream tasks, there is no inherent reason for the rank of embeddings to transfer in a monotonic way to them. However if the source dataset is diverse enough and target datasets have some semantic overlap with the source dataset, then we have

$$\text{rank}(\boldsymbol{Z}_{\text{target}}) \propto \text{rank}(\boldsymbol{Z}_{\text{source}}). \tag{2}$$

We observe in section 3.2 and fig. 2 that the rank of JE-SSL representations scales linearly between different input distributions e.g. going from a *source* task such as Imagenet (Deng et al., 2009) to a *target* task such as iNaturalist. This is further confirmed by Pearson correlation coefficients greater than 0.99, except for StanfordCard where it is 0.88. Interestingly, we observe that the StanfordCars dataset suffers from a less distinctive linear scaling due to the dataset distribution having a small overlap with ImageNet. This indicates that as long as the source dataset is relatively diverse then using RankMe to select a model with greater embeddings' rank will also correspond to selecting a model with greater embeddings' rank on the target dataset.

Furthermore, as the train performance increases, so does the test performance. We validate this in fig. 2. As a result, using RankMe to select a model with greater train performance is enough to also select a model with greater test performance.

Finally, we report in fig. 2 that the performance of embeddings and representations scales almost monotonically. These results are supported by visualization of representations and embeddings from feature inversion models (Bordes et al., 2021). Hence, using RankMe to select the model maximizing the performance on the former also selects a model maximizing performance on the latter.

With these three hypotheses validated empirically, we can confidently say that RankMe computed on the embeddings of the source dataset is a predictor of representations' performance on target datasets.

Robust RankMe Implementation. One of the most crucial step of RankMe is the estimation of the embeddings's rank. A trivial solution could be to check at the number of nonzero singular values. Denoting by σ_k the k^{th} singular value of the $(N \times K)$ embedding matrix $\boldsymbol{Z}$, this would lead to $\text{rank}(\boldsymbol{Z}) = \sum_{k=1}^{\min(N,K)} 1_{\{\sigma_k>0\}}$. However, such a definition is too rigid for practical scenarios. For example, round-off error alone could have a dramatic impact on the rank estimate. Instead, alternative and robust rank definitions have emerged (Press et al., 2007) such as $\text{rank}(\boldsymbol{Z}) = \sum_{k=1}^{\min(N,K)} 1_{\{\sigma_k>\max_i \sigma_i \times \max(M,N) \times \epsilon\}}$, where ϵ is a small constant dependant on the data type, typically 10^{-7} for `float32`.

An alternative measure of rank comes from a probability viewpoint where the singular values are normalized to sum to 1 and the Shannon Entropy (Shannon, 1948) is used which corresponds to our definition of RankMe from eq. (1). As opposed to section 3.1, the chosen eq. (1) does not rely on specifying the exact threshold at which the singular value is treated as nonzero. Throughout our study we be employ eq. (1) and provide the matching analysis with section 3.1 in the appendix. Another benefit of RankMe's eq. (1) is in its quantification of the whitening of the embeddings in addition to their rank, which is known to simplify optimization of (non)linear probes put on top of them (Santurkar et al., 2018). Lastly, although eq. (1) is defined with the full embedding matrix $\boldsymbol{Z}$, we observe that not all the samples need to be used to have an accurate estimate of RankMe. In practice we use 25600 samples as ablation studies provided in supplementary section C and fig. S2 indicate that this provides an highly accurate estimate.

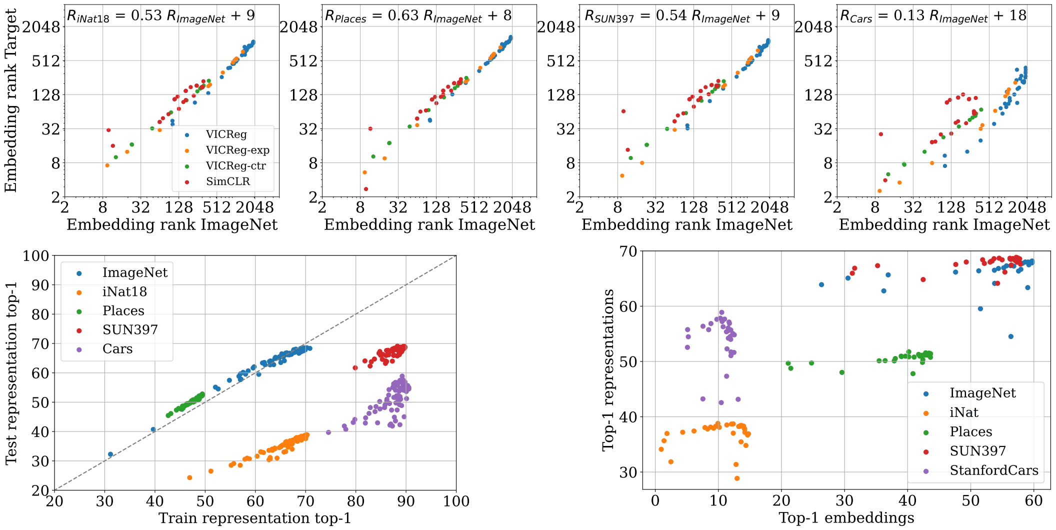


Figure 2: Validation of the hypotheses motivating RankMe.**(Top)** Embeddings' rank transfers from source to target datasets. The estimates use 25600 images from the respective datasets.**(Bottom Left)** Train and test accuracy are highly correlated across datasets.**(Bottom Right)** An increase in performance on embeddings leads to an increase in performance on representations.

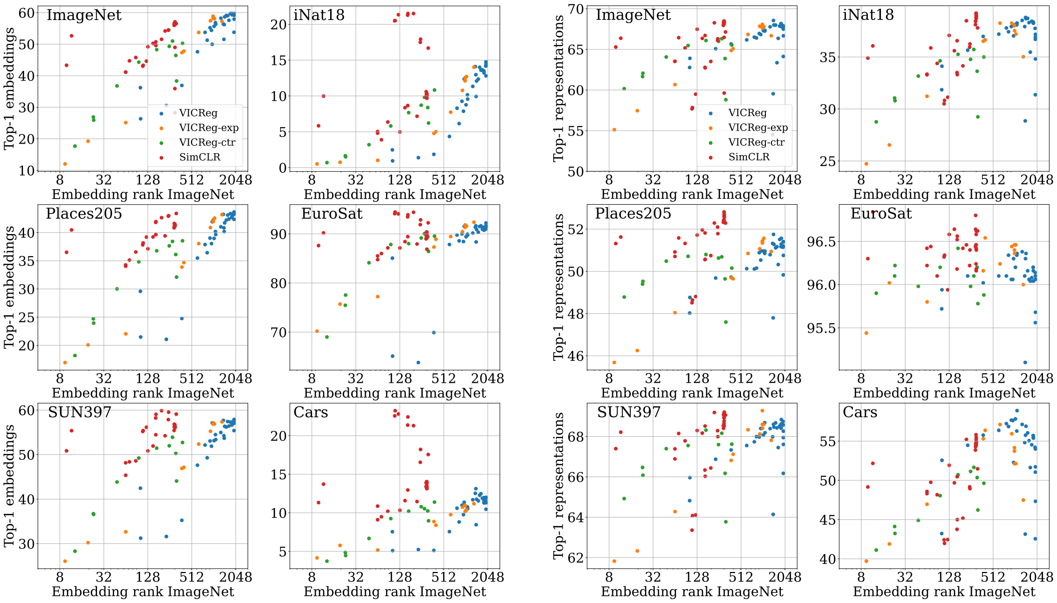


Figure 3: **(Left)** Validation of RankMe on embeddings, a higher ImageNet rank leads to improved performance across methods and datasets.**(Right)** Validation of RankMe on representations, where the link is even clearer, reinforcing RankMe's practical use.

3.2 RankMe Predicts Linear Probing Downstream performance Even on Unseen Datasets

In order to empirically validate RankMe, we compare it to linear evaluation, which is the default evaluation method of JE-SSL methods. Finteuning has gained in popularity with Masked Image Modeling methods but this can have a significant impact on the properties of the embeddings and alters what was learned during the pretraining. As such we do not focus on this evaluation.

Experimental Methods and Datasets Considered In order to provide a meaningful assessment of the embeddings rank's impact on performance we focus on 4 JE-SSL methods. We use SimCLR as a representative contrastive method, VICReg as a representative covariance based method, and VICReg-exp and VICReg-ctr which were introduced in Garrido et al. (2022). To make our work self-contained, we present the methods in supplementary section A. We chose to use VICReg-exp and VICReg-ctr as they provide small modifications to VICReg and SimCLR while producing embeddings with different rank properties. For each method we vary parameters that directly influence the rank of the embeddings, whether it is the temperature use in softmax based methods, which directly impacts the hardness of the softmax, or the loss weights to give more or less importance to the regularizing aspect of loss functions. We also vary optimization parameters such as the learning rate and weight decay to provide a more complete analysis. We provide the hyperparameters used for all experiments in supplementary section G. All approaches were trained in the same experimental setting with a ResNet-50 (He et al., 2016) backbone with a MLP projector having intermediate layers of size $8192, 8192, 2048$, which avoids any architectural rank constraints. The models were trained for 100 epochs on ImageNet with the LARS (You et al., 2017; Goyal et al., 2017) optimizer. In order to evaluate the methods, we used ImageNet (our source dataset), as well as INaturalist18 (Horn et al., 2018), Places205 (Zhou et al., 2014), EuroSat (Helber et al., 2019), SUN397 (Xiao et al., 2010) and StanfordCars (Krause et al., 2013) to evaluate the trained models on unseen datasets. These commonly used datasets provide a wide range of scenarios that differ from ImageNet and provide meaningful ways to test the robustness of RankMe. For example, iNaturalist18 consists of 8412 classes focused on fauna and flora which requires more granularity than similar classes on ImageNet, SUN397 focuses on scene understanding, deviating from the single object and object-centric images of ImageNet, and EuroSat consists of satellite images which again differ from ImageNet. Datasets such as iNaturalist can allow theoretical limitations to manifest themselves more clearly due to the number of classes being significantly higher than the rank of learned representations. In order to evaluate on those datasets, we relied on the VISSL library (Goyal et al., 2021). We provide complete details on the pretraining and evaluations in supplementary section E.

RankMe as a prediction of linear classification accuracy. As can be seen in fig. 3, for a given method the performance on the embedding is improved by with a higher embedding rank, whether we look on ImageNet on which the models were pretrained or on downstream datasets. Nonetheless, there are some visible outliers but they are mostly on SimCLR in settings with very high error rates compared to before the projector, such as in iNaturalist or StanfordCars. The conceptual closeness between VICReg-ctr and SimCLR pointed out in Garrido et al. (2022) would also suggest that these results need to be interpreted carefully, but they do reinforce the fact that a high rank is a necessary and not sufficient condition for improved performance on downstream tasks. It is also very tempting to draw conclusions when comparing different approaches, especially when looking at the ImageNet performance, however since dimensional collapse is not the only performance deciding factor one should refrain from doing so. The link between embedding rank and performance is even clearer when evaluating on the representations, as is usually done. In this scenario the link is even more consistent across datasets where we observe again than a higher rank is necessary for improved performance. This solidifies the use of RankMe as a performance metric that can be used in practical scenarios.

3.3 Going Further: RankMe Also Holds for Nonlinear Probing and for Different Architectures

Non-linear evaluation. While we have been focusing on linear evaluation, one can wonder if the behaviour changes when using a more complex task-related head. We thus give some evidence that the previously observed behaviours are similar with a non-linear classification head. We used a simple 3 layer MLP with intermediate dimensions 2048, where each layer is followed by a ReLU activation. This choice of dimensions ensures that there are no architectural rank constraints on the embeddings. We focused on SUN397 and StanfordCars for this study due to their conceptual differences to ImageNet. The low rank of embeddings produced by SimCLR on these datasets would suggest that a non-linear classifier might help improve performance since it is not as theoretically limited by the embeddings' rank as it is in the linear setting. However, as we can see in fig. 4, the behaviours for all methods is the same as in the linear regime. This would suggest that RankMe is also a suitable metric to evaluate downstream performance in a non-linear setting.

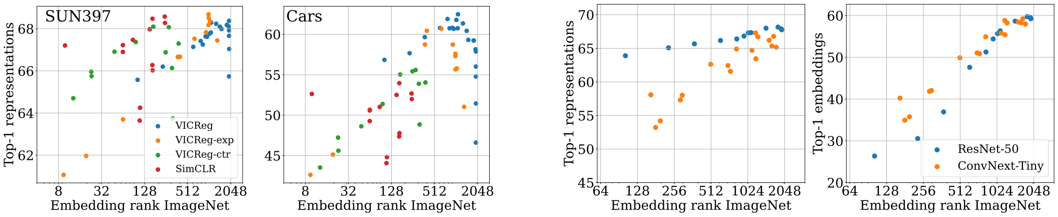


Figure 4: Impact of rank on performance on other architectures and evaluation protocols. **(Left)** ConvNexts are also sensitive to dimensional collapse, showing that rank-deficiency is not an artifact of ResNets. **(Right)** Using a 3 layer MLP as classification head does not alter the performance before or after the projector, showing that RankMe can go beyond linear evaluation.

Algorithm 1 Hyperparameter selection wiht RankMe

```
Require: Models $f_1, \dots, f_N$ to compare, in increasing value of the hyperparameter
Require: Corresponding ranks $r_1, \dots, r_N$
$f_{best} \leftarrow f_1,\ r_{best} \leftarrow r_1$
for $i = 2$ to $N$ do
    if $r_i > r_{best}$ then
        $f_{best} \leftarrow f_i,\ r_{best} \leftarrow r_i$
    else if $r_i = r_{best}$ and $(r_i > r_{i-1}$ or $r_i > r_{i+1})$ then
        $f_{best} \leftarrow f_i,\ r_{best} \leftarrow r_i$
return $f_{best}$
```

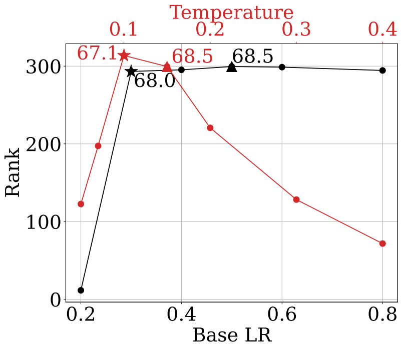


Figure 5: **(Left)** Algorithm describing how to use RankMe for hyperparameter selection. We select either the highest rank model, or if there are multiple ones the one with the minimal/maximal value achieving it.**(Right)** Visual example of the hyperparameter selection applied to SimCLR's temperature and learning rate. The star indicates the value that is selected using RankMe and the triangle the one with the ImageNet oracle. Notice the high rank of oracle selected models.

Dimensional collapse on different architectures. Our results so far have only focused on ResNet-50s, and a concern could be that the architecture played a significant role in the introduction of collapse. As such we trained VICReg in the same setting as before but using ConvNext-T (Liu et al., 2022) as the backbone architecture. As we can see in fig. 4, collapse still appears in this case, with an even stronger impact on performance on ImageNet. This reinforces the findings of Jing et al. (2022); He & Ozay (2022), which study collapse through the used loss function independently of the architecture of the backbone.

4 RankMe for Label-Free hyperparameter selection in SSL

We previously focused on validating RankMe by comparing overall performance compared to linear evaluation. In this section we focus on the evolution of rank and performance when varying one hyperparameter at a time in order to demonstrate how RankMe can be used for hyperparameter selection. We focus on loss specific hyperparameters such as the loss weights or temperature as well as hyperparameters related to optimization, such as the learning rate and weight decay.

4.1 Using RankMe to choose the correct hyperparameter value

As we have shown before, having a higher rank leads to better performance, and using RankMe to find the correct value of an hyperparameter is as simple as choosing the value that leads to the highest rank, as illustrated in fig. 5. Certain hyperparemeters will lead to plateaus of equal rank, and in those the value that first achieves the maximal value should be selected. This second part is only applicable when hyperparameter values can be ordered, and may be omitted for categorical values. This is analogous to the elbow method that is used in clustering.

Table 1: Top-1 accuracies obtained by doing hyperparameter selection using ImageNet validation performance or RankMe. OOD indicates the average performance over all the considered datasets other than ImageNet. The performance is computed on the embeddings.

Dataset	Method	VICReg				SimCLR		
		cov.	inv.	LR	WD	temp.	LR.	WD.
ImageNet	ImageNet Oracle	59.7	59.7	59.7	59.7	56.9	56.9	57.1
	RankMe	59.6	59.7	59.7	59.5	56.5	56.0	57.1
	Difference	−0.1	=	=	−0.2	−0.4	−0.9	=
OOD	ImageNet Oracle	43.3	43.6	43.3	43.4	42.5	42.5	42.0
	RankMe	43.5	43.6	43.3	43.1	45.8	41.7	42.0
	Difference	+0.2	=	=	−0.3	+3.3	−0.8	=

Even in cases where the values cannot be compared and equal ranks are found in a different setting, this still makes it possible to discard some runs and only focus on the one that achieve the maximal rank. This further highlight how maximal rank is only a necessary condition for good performance, however when the hyperparameters are ordered we can go one step further and use the rank alone to find a good hyperparameter value.

4.2 Experiments

In order to demonstrate the effectiveness of RankMe for hyperparameter selection, we apply the algorithm presented in fig. 5 to find the best values for a given set of hyperparameters for VICReg and SimCLR. Our focus is on the covariance and invariance weights in VICReg, the temperature in SimCLR, and on learning rate and weight decay for both. We compare the performance on ImageNet as well as the average performance on the previously discussed OOD datasets to models selected by their ImageNet top-1 accuracy on its validation set. For detailed performance on every dataset, confer supplementary section F.

As can be seen in table 1, using RankMe we are able to retrieve most of the performance on ImageNet, with gaps being lower than half a point. It is not possible to beat the selection using ImageNet's validation since this is the metric we are evaluating on. However, on OOD datasets we are able to improve the performance in certain settings, and match it in the others. Thus, when comparing performance after the projector, RankMe is the better approach of the two to select the hyperparameters that will generalize best to unseen datasets. As pointed out in Girish et al. (2022), using ImageNet performance to select models can lead to suboptimal performance in downstream tasks, which our results further confirm and reinforces the need for a new way of selecting hyperparameters. When looking at performance before the projector in fig. 1, we can see that here RankMe does not beat the models selected with ImageNet's validation set, even on OOD datasets. Nonetheless the gaps are on average of less than half a point, which shows how competitive RankMe can be for hyperparameter selection, despite using no labeled data, having no parameters to tune, and being able to be computed in a couple of minutes.

5 Conclusion

We have shown how the phenomenon of dimensional collapse in self-supervised learning can be used as a powerful metric to evaluate models. By using a theoretically motivated analogue of the rank of embeddings, we show that the performance on downstream datasets can easily be assessed by only looking at the training dataset, without any labels, training or parameters. While our work focuses on linear classification, we show promising results in non-linear classification that raise the question of how general this simple metric can be. Furthermore, its competitiveness with traditional oracle based hyperparameter selection methods makes it a promising tool in settings where labels are scarce, such as in the case of large uncurated datasets. As such, this work makes a step towards completely label-less self-supervised learning, as most existing approaches' hyperparameters are tuned with the help of ImageNet's validation set. Further work will explore the use of RankMe in more varied scenarios, to further legitimize its use in designing better self-supervised approaches.

6 Reproducibility statement

While reproducing the pretrainings is prohibitively expensive since each training takes around a day on 8 V100 GPUs, we provide all of the hyperparameters used in supplementary section G. We also provide all of the pretraining details in supplementary section E, along with the hyperparameters used for the linear evaluations in the same section. We also provide all the performance and rank values to reproduce our main figures in supplementary section G. While the implementation of RankMe is straightforward, we prove an example algorithm to use it in fig. 5. All of these efforts should make our results reproducible and verifiable.

References

Randall Balestriero and Yann LeCun. Contrastive and non-contrastive self-supervised learning recover global and local spectral embedding methods. *arXiv preprint arXiv:2205.11508*, 2022.

Adrien Bardes, Jean Ponce, and Yann LeCun. Vicreg: Variance-invariance-covariance regularization for self-supervised learning. *arXiv preprint arXiv:2105.04906*, 2021.

Florian Bordes, Randall Balestriero, and Pascal Vincent. High fidelity visualization of what your self-supervised representation knows about. *arXiv preprint arXiv:2112.09164*, 2021.

Jane Bromley, Isabelle Guyon, Yann LeCun, Eduard Sackinger, and Roopak Shah. Signature verification using a "siamese" time delay neural network. In *NeurIPS*, 1994.

Mathilde Caron, Piotr Bojanowski, Armand Joulin, and Matthijs Douze. Deep clustering for unsupervised learning. In *ECCV*, 2018.

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

Mathilde Caron, Hugo Touvron, Ishan Misra, Herve Jegou, and Julien Mairal Piotr Bojanowski Armand Joulin. Emerging properties in self-supervised vision transformers. In *ICCV*, 2021.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *ICML*, pp. 1597–1607. PMLR, 2020a.

Xinlei Chen and Kaiming He. Exploring simple siamese representation learning. In *CVPR*, 2020.

Xinlei Chen, Haoqi Fan, Ross Girshick, and Kaiming He. Improved baselines with momentum contrastive learning. *arXiv preprint arXiv:2003.04297*, 2020b.

Xinlei Chen, Saining Xie, and Kaiming He. An empirical study of training self-supervised vision transformers. In *ICCV*, 2021.

Thomas M Cover. Geometrical and statistical properties of systems of linear inequalities with applications in pattern recognition. *IEEE transactions on electronic computers*, (3):326–334, 1965.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009.

Aleksandr Ermolov, Aliaksandr Siarohin, Enver Sangineto, and Nicu Sebe. Whitening for self-supervised representation learning, 2021.

Octavian Ganea, Sylvain Gelly, Gary Bécigneul, and Aliaksei Severyn. Breaking the softmax bottleneck via learnable monotonic pointwise non-linearities. In *International Conference on Machine Learning*, pp. 2073–2082. PMLR, 2019.

Quentin Garrido, Yubei Chen, Adrien Bardes, Laurent Najman, and Yann Lecun. On the duality between contrastive and non-contrastive self-supervised learning. *arXiv preprint arXiv:2206.02574*, 2022.

Sharath Girish, Debadeepta Dey, Neel Joshi, Vibhav Vineet, Shital Shah, Caio Cesar Teodoro Mendes, Abhinav Shrivastava, and Yale Song. One network doesn't rule them all: Moving beyond handcrafted architectures in self-supervised learning. *arXiv preprint arXiv:2203.08130*, 2022.

Priya Goyal, Piotr Dollár, Ross Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, and Kaiming He. Accurate, large minibatch sgd: Training imagenet in 1 hour. *arXiv preprint arXiv:1706.02677*, 2017.

Priya Goyal, Quentin Duval, Jeremy Reizenstein, Matthew Leavitt, Min Xu, Benjamin Lefaudeux, Mannat Singh, Vinicius Reis, Mathilde Caron, Piotr Bojanowski, Armand Joulin, and Ishan Misra. Vissl. `https://github.com/facebookresearch/vissl`, 2021.

Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre H. Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Daniel Guo, Mohammad Gheshlaghi Azar, Bilal Piot, Koray Kavukcuoglu, Rémi Munos, and Michal Valko. Bootstrap your own latent: A new approach to self-supervised learning. In *NeurIPS*, 2020.

Jeff Z HaoChen, Colin Wei, Adrien Gaidon, and Tengyu Ma. Provable guarantees for self-supervised deep learning with spectral contrastive loss. *NeurIPS*, 34, 2021.

Bobby He and Mete Ozay. Exploring the gap between collapsed & whitened features in self-supervised learning. In *International Conference on Machine Learning*, pp. 8613–8634. PMLR, 2022.

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

Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *CVPR*, 2020.

Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 16000–16009, 2022.

Patrick Helber, Benjamin Bischke, Andreas Dengel, and Damian Borth. Eurosat: A novel dataset and deep learning benchmark for land use and land cover classification. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 12(7):2217–2226, 2019.

Grant Van Horn, Oisin Mac Aodha, Yang Song, Yin Cui, Chen Sun, Alex Shepard, Hartwig Adam, Pietro Perona, and Serge Belongie. The inaturalist species classification and detection dataset. In *CVPR*, 2018.

Tianyu Hua, Wenxiao Wang, Zihui Xue, Sucheng Ren, Yue Wang, and Hang Zhao. On feature decorrelation in self-supervised learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 9598–9608, 2021.

Li Jing, Pascal Vincent, Yann LeCun, and Yuandong Tian. Understanding dimensional collapse in contrastive self-supervised learning. In *International Conference on Learning Representations*, 2022. URL `https://openreview.net/forum?id=YevsQ05DEN7`.

Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. 3d object representations for fine-grained categorization. In *Proceedings of the IEEE international conference on computer vision workshops*, pp. 554–561, 2013.

Kuang-Huei Lee, Anurag Arnab, Sergio Guadarrama, John Canny, and Ian Fischer. Compressive visual representations. In *NeurIPS*, 2021.

Chunyuan Li, Jianwei Yang, Pengchuan Zhang, Mei Gao, Bin Xiao, Xiyang Dai, Lu Yuan, and Jianfeng Gao. Efficient self-supervised vision transformers for representation learning. In *ICLR*, 2022a.

Zengyi Li, Yubei Chen, Yann LeCun, and Friedrich T Sommer. Neural manifold clustering and embedding. *arXiv preprint arXiv:2201.10000*, 2022b.

Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A convnet for the 2020s. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11976–11986, 2022.

Ishan Misra and Laurens van der Maaten. Self-supervised learning of pretext-invariant representations. In *CVPR*, 2020.

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

William H Press, Saul A Teukolsky, William T Vetterling, and Brian P Flannery. *Numerical recipes 3rd edition: The art of scientific computing*. Cambridge university press, 2007.

Colorado J Reed, Sean Metzger, Aravind Srinivas, Trevor Darrell, and Kurt Keutzer. Selfaugment: Automatic augmentation policies for self-supervised learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 2674–2683, 2021.

Olivier Roy and Martin Vetterli. The effective rank: A measure of effective dimensionality. In *2007 15th European signal processing conference*, pp. 606–610. IEEE, 2007.

Shibani Santurkar, Dimitris Tsipras, Andrew Ilyas, and Aleksander Madry. How does batch normalization help optimization? *Advances in neural information processing systems*, 31, 2018.

Claude Elwood Shannon. A mathematical theory of communication. *The Bell system technical journal*, 27(3):379–423, 1948.

Nenad Tomasev, Ioana Bica, Brian McWilliams, Lars Buesing, Razvan Pascanu, Charles Blundell, and Jovana Mitrovic. Pushing the limits of self-supervised resnets: Can we outperform supervised learning without labels on imagenet? *arXiv preprint arXiv:2201.05119*, 2022.

Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. Deep image prior. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 9446–9454, 2018.

Zhirong Wu, Yuanjun Xiong, Stella Yu, , and Dahua Lin. Unsupervised feature learning via non-parametric instance discrimination. In *CVPR*, 2018.

Jianxiong Xiao, James Hays, Krista A Ehinger, Aude Oliva, and Antonio Torralba. Sun database: Large-scale scene recognition from abbey to zoo. In *2010 IEEE computer society conference on computer vision and pattern recognition*, pp. 3485–3492. IEEE, 2010.

Chun-Hsiao Yeh, Cheng-Yao Hong, Yen-Chi Hsu, Tyng-Luh Liu, Yubei Chen, and Yann LeCun. Decoupled contrastive learning. *arXiv preprint arXiv:2110.06848*, 2021.

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

Jure Zbontar, Li Jing, Ishan Misra, Yann LeCun, and Stéphane Deny. Barlow twins: Self-supervised learning via redundancy reduction. In *ICML*, pp. 12310–12320. PMLR, 2021.

Bolei Zhou, Agata Lapedriza, Jianxiong Xiao, Antonio Torralba, and Aude Oliva. Learning deep features for scene recognition using places database. In *NeurIPS*, 2014.

Jinghao Zhou, Chen Wei, Huiyu Wang, Wei Shen, Cihang Xie, Alan Yuille, and Tao Kong. ibot: Image bert pre-training with online tokenizer. In *ICLR*, 2022a.

Pan Zhou, Yichen Zhou, Chenyang Si, Weihao Yu, Teck Khim Ng, and Shuicheng Yan. Mugs: A multi-granular self-supervised learning framework. 2022b.

Chengxu Zhuang, Alex Lin Zhai, and Daniel Yamins. Local aggregation for unsupervised learning of visual embeddings. In *ICCV*, 2019.

A Background

In order to make our work as self-contained as possible, we recall the loss functions of the methods we study. For conciseness we refer to the outputs of the encoder as *representations* and the outputs of the projection head as *embeddings*, which we denote by $z_i \in \mathbb{R}^d$. We first briefly recall that the SimCLR loss is given by

$$\mathcal{L} = -\sum_{(i,j)\in\mathbb{P}} \frac{e^{CoSim(\boldsymbol{z}_i,\boldsymbol{z}_j)}}{\sum_{k=1}^{N} \mathbf{1}_{\{k\neq i\}} e^{CoSim(\boldsymbol{z}_i,\boldsymbol{z}_k)}},$$

with $\mathbb{P}$ the set of all positive pairs in the current mini-batch or dataset that comprise N exemplars.

VICReg's loss is defined with three components. The variances loss v acts as a norm regularizer for the dimensions, and the covariance loss aims at decorrelating dimensions in the embeddings. They are respectively defined as

$$v(\boldsymbol{Z}) = \frac{1}{d}\sum_{i=1}^{d} \max\left(0, 1 - \sqrt{\text{Var}(Z_{\cdot,i})}\right) \text{ and } c(\boldsymbol{Z}) = \frac{1}{d}\sum_{i\neq j} \text{Cov}(\boldsymbol{Z})^2.$$

Both of these loss are combined with an invariance loss that matches positives pairs, giving a final loss of

$$\mathcal{L} = \lambda \sum_{(i,j)\in\mathbb{P}} \|z_i - z_j\|_2^2 + \mu\, c(\boldsymbol{Z}) + \nu\, v(\boldsymbol{Z}).$$

VICReg-exp is definied similarly, but with the exponential covariance loss defined as

$$c_{exp}(\boldsymbol{Z}) = \frac{1}{d}\sum_{i} \log\left(\sum_{j\neq i} e^{\text{Cov}(\boldsymbol{Z})_{i,j}/\tau}\right). \tag{3}$$

VICReg-ctr is then VICReg-exp but applied to $\boldsymbol{Z}^T$, making it a contrastive approach and conceptually similar to SimCLR. These methods give us different scenarios of collapse and allow us to make a more general study of the rank of representations as a powerful metric.

B Comparison of the rank estimators

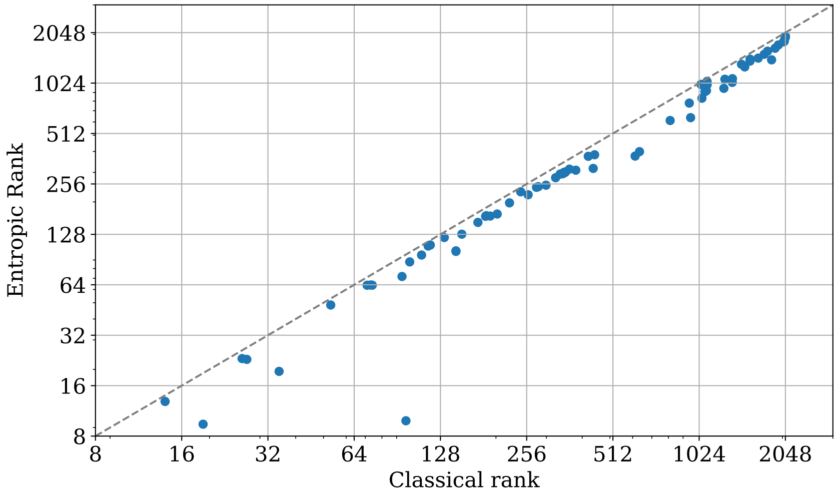


Figure S1: Relationship between the two rank estimators, Pearson correlation coefficient of 0.99. Outliers indicate embeddings with singular values to the threshold, showing how the entropic rank takes into account this information.

Since we do not rely on the classical threshold-based rank estimator, it is important to verify how well our entropy based one correlates with it. As we can see in fig. S1, both estimates discussed previously correlate extremely well, showing that using one or the other should not lead to significant differences, as validated in supplementary section D. Nonetheless the entropic estimator takes into account the degree of whitening of the embeddings, which links better to theoretical results.

C Convergence of the rank estimators

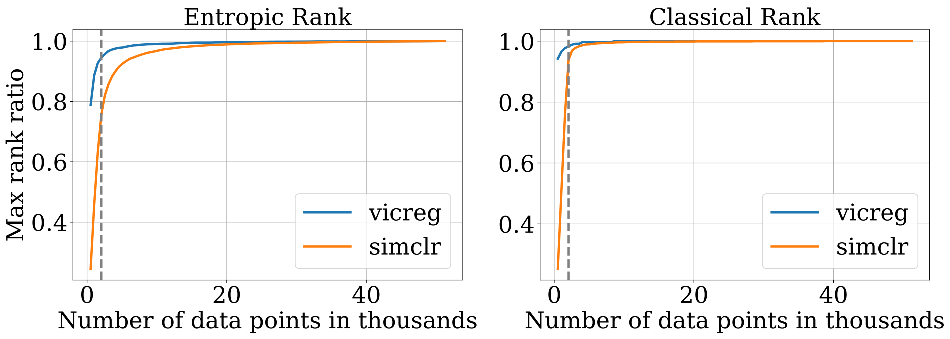


Figure S2: Convergence of the rank estimators on ImageNet as a function of the number of samples for 2048 dimensional outputs, as indicated by the vertical line.

As we can see in fig. S2, the rank estimates converge extremely quickly, especially for VICReg. For both VICReg and SimCLR, 10000 samples are enough to obtain more than 95% of the final rank. It is worth noting that the entropic rank estimator converges more slowly that the classical rank estimator as it is more sensitive to the singular values. The fact that the rank can be approximated with few samples is encouraging for its use during training and not only as a measure of performance after pretraining.

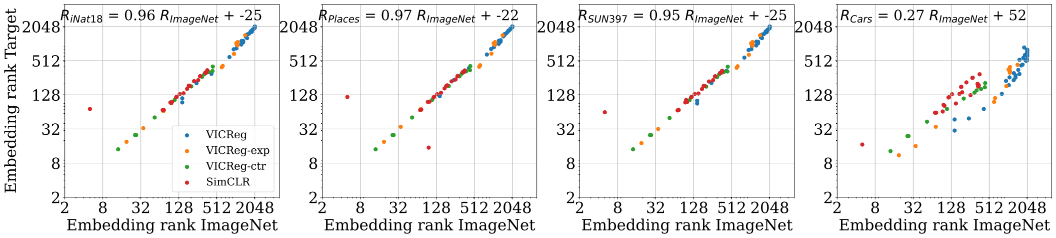


Figure S3: Reproduction of the top of fig. 2 with the classical rank estimator. Embeddings' rank transfers from source to target datasets. The estimates used 25600 images from the respective datasets.

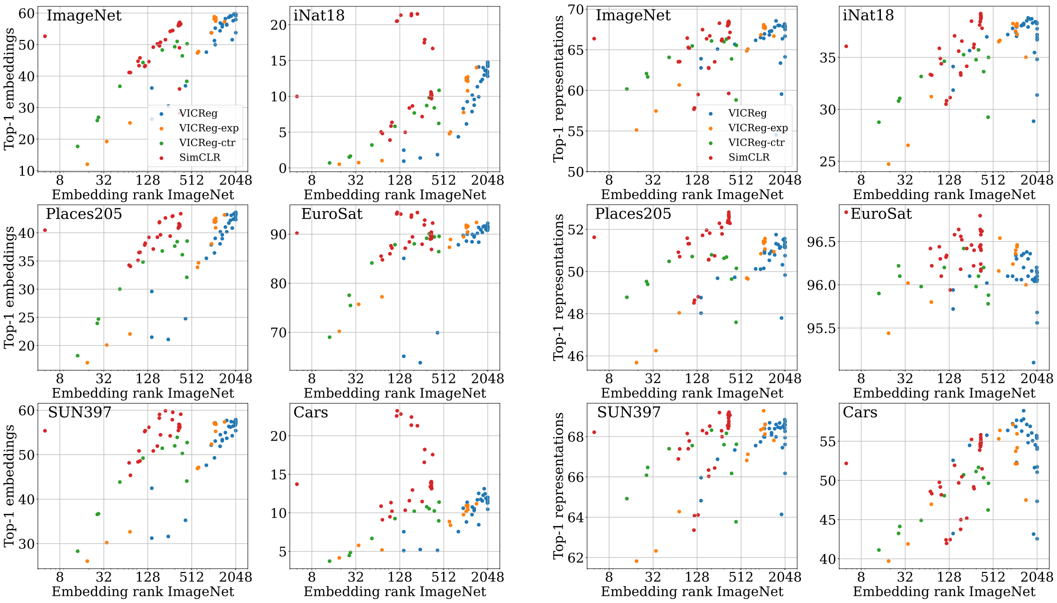


Figure S4: Reproduction of fig. 3 with the classical rank estimator.**(Left)** Validation of RankMe on embeddings, a higher ImageNet rank leads to improved performance across methods and datasets.**(Right)** Validation of RankMe on representations, where the link is even clearer, reinforcing RankMe's practical use.

D Reproduction of figures with the classical rank estimator

As can be seen in figs. S3 and S4, the results that we obtain using the classical threshold-based rank estimator are extremely similar to the ones with the entropic estimator. The exact values do differ, but the behaviours stay the same. One of the main differences is illustrated in fig. S4, where we can see that the target rank is almost identical to the source one when we previously saw a drop of around 50%. This can be explained by the fact that some features may be less present in the target dataset, reducing the associated singular values, and thus the entropic rank.
All of this shows that using one or the other will lead to similar results in practical scenarios.

E Detailed training and evaluation procedures

E.1 Pretraining

All pretraining were done with ResNet-50 backbones. The projector used is a MLP with intermediate dimensions $8192, 8192, 2048$. They were trained with the LARS optimizer using a momentum of 0.9, weight decay 10^{-6} and varying learning rates depending on the method. VICReg used 0.3 base learning rate, SimCLR 0.3 or 0.5 depending on the experiment, VICReg-exp 0.6 and VICReg-ctr 0.6. The learning rate is then computed as $lr = base\_lr * batch\_size/256$. We do a 10-epochs linear

Table S1: Image augmentation parameters, taken from Grill et al. (2020).

Parameter	View 1	View 2
Random crop probability	1.0	1.0
Horizontal flip probability	0.5	0.5
Color jittering probability	0.8	0.8
Brightness adjustment max intensity	0.4	0.4
Contrast adjustment max intensity	0.4	0.4
Saturation adjustment max intensity	0.2	0.2
Hue adjustment max intensity	0.1	0.1
Grayscale probability	0.2	0.2
Gaussian blurring probability	1.0	0.1
Solarization probability.	0.0	0.2

warmup and then use cosine annealing. We used batch sizes of 2048 for SimCLR and 1024 for other methods. SimCLR and VICReg-ctr also use a default temperature of 0.15, and 0.1 for VICReg-exp. We used the image augmentation strategy from Grill et al. (2020) illustrated in table S1.

E.2 Evaluation

Table S2: Optimization parameters used to evaluate on downstream datasets

Dataset	Optimizer	Weight decay	Momentum	Learning rate	Epochs
iNaturalist18	SGD (w/ Nesterov)	0.0005	0.9	0.01	84
Places205	SGD (w/ Nesterov)	0.0005	0.9	0.01	14
EuroSat	SGD (w/ Nesterov)	0.0005	0.9	0.01	28
Sun397	SGD (w/ Nesterov)	0.0005	0.9	0.01	28
StanfordCars	SGD (w/ Nesterov)	0.0005	0.9	0.1	28

For all datasets except StanfordCars, we use the standard protocol in VISSL. On StanfordCars we mostly tuned the learning rate. The parameters that we used are described in table S2. For data augmentation, we used random resized crops and random horizontal flips during training, and center crop for evaluation.

F Detailed tables for hyperparameter selection

Table S3: Top-1 accuracies computed on representations when tuning hyperparameters with ImageNet validation performance or with RankMe.

Dataset	Method	VICReg				SimCLR		
		cov.	inv.	LR	WD	temp.	LR.	WD.
ImageNet	ImageNet Oracle	68.2	68.2	68.6	68.0	68.5	68.5	68.3
	RankMe	67.8	67.9	68.2	67.8	67.1	68.0	68.3
	Difference	−0.4	−0.3	−0.4	−0.2	−1.4	−0.5	=
iNat18	ImageNet Oracle	38.4	38.4	38.8	38.3	39.2	39.2	38.9
	RankMe	36.7	37.2	38.4	38.3	37.8	38.0	38.9
	Difference	−1.7	−1.2	−0.4	=	−1.4	−1.2	=
Places205	ImageNet Oracle	51.2	51.2	51.8	51.3	52.4	52.4	52.6
	RankMe	51.2	51.4	51.2	51.6	52.3	52.3	52.6
	Difference	=	+0.2	−0.6	+0.3	−0.1	−0.1	=
EuroSat	ImageNet Oracle	96.2	96.2	96.3	96.2	96.5	96.5	96.4
	RankMe	96.1	96.1	96.2	96.0	96.6	96.4	96.4
	Difference	−0.1	−0.1	−0.1	−0.2	+0.1	−0.1	=
SUN397	ImageNet Oracle	68.4	68.4	68.6	68.6	68.9	68.9	69.2
	RankMe	68.6	68.3	68.4	68.8	69.1	68.5	69.2
	Difference	+0.2	−0.1	−0.2	+0.2	+0.2	−0.4	=
Cars	ImageNet Oracle	55.7	55.7	55.8	55.6	54.4	54.4	54.9
	RankMe	51.1	54.0	55.7	55.4	51.5	53.9	54.9
	Difference	−4.6	−1.7	−0.1	−0.2	−2.9	−0.5	=
Average	ImageNet Oracle	63.0	63.0	63.3	63.0	63.3	63.3	63.4
	RankMe	61.9	62.5	63.0	63.0	62.4	62.9	63.4
	Difference	−1.1	−0.5	−0.3	=	−0.9	−0.4	=

Table S4: Top-1 accuracies computed on embeddings when tuning hyperparameters with ImageNet validation performance or with RankMe.

Dataset	Method	VICReg				SimCLR		
		cov.	inv.	LR	WD	temp.	LR.	WD.
ImageNet	ImageNet Oracle	59.7	59.7	59.7	59.7	56.9	56.9	57.1
	RankMe	59.6	59.7	59.7	59.5	56.5	56.0	57.1
	Difference	−0.1	=	=	−0.2	−0.4	−0.9	=
iNat18	ImageNet Oracle	13.5	14.2	13.5	13.6	10.3	10.3	9.8
	RankMe	14.2	14.2	13.5	13.4	16.7	9.8	9.8
	Difference	+0.7	=	=	−0.2	+6.4	−0.5	=
Places205	ImageNet Oracle	42.7	43.3	42.7	43.4	41.2	41.2	41.2
	RankMe	43.2	43.3	42.7	42.7	43.4	40.8	41.2
	Difference	+0.5	=	=	−0.7	+2.2	−0.4	=
EuroSat	ImageNet Oracle	91.3	91.7	91.3	91.0	90.4	90.4	89.5
	RankMe	91.0	91.7	91.3	91.3	92.3	89.0	89.5
	Difference	−0.3	=	=	+0.3	+1.9	−1.4	=
SUN397	ImageNet Oracle	57.3	57.0	57.3	57.3	56.4	56.4	56.2
	RankMe	57.4	57.0	57.3	56.7	59.1	55.4	56.2
	Difference	+0.1	=	=	−0.6	+2.7	−1.0	=
Cars	ImageNet Oracle	12.0	12.0	12.0	11.9	14.0	14.0	13.2
	RankMe	11.6	12.0	12.0	11.5	17.6	13.4	13.2
	Difference	−0.4	=	=	−0.4	+3.6	−0.6	=
Average	ImageNet Oracle	46.1	46.3	46.1	46.1	44.9	44.9	44.5
	RankMe	46.2	46.3	46.1	45.9	47.6	44.1	44.5
	Difference	+0.1	=	=	−0.2	+2.7	−0.8	=

G Complete performance tables

Table S5: Top-1 on representations in all settings.

Method	Hyperparameters	INet	iNat18	Places	EuroSat	SUN	Cars
VICReg	$\lambda : 25, \mu : 25, \nu : 0.3$	63.90	34.12	48.77	95.94	65.96	52.56
	$\lambda : 25, \mu : 25, \nu : 0.4$	65.08	35.65	49.68	96.10	66.87	54.47
	$\lambda : 25, \mu : 25, \nu : 0.5$	65.67	36.97	49.73	96.02	67.33	55.76
	$\lambda : 25, \mu : 25, \nu : 0.6$	66.17	37.20	50.13	96.10	67.55	56.37
	$\lambda : 25, \mu : 25, \nu : 0.7$	66.40	37.42	50.15	96.34	67.99	56.86
	$\lambda : 25, \mu : 25, \nu : 0.8$	66.83	38.05	50.53	96.06	68.40	57.63
	$\lambda : 25, \mu : 25, \nu : 0.9$	67.30	38.13	50.96	96.20	68.08	57.83
	$\lambda : 25, \mu : 25, \nu : 1$	67.34	38.26	50.96	96.36	68.19	58.89
	$\lambda : 25, \mu : 25, \nu : 2$	68.00	38.68	51.28	96.36	68.46	56.90
	$\lambda : 25, \mu : 25, \nu : 4$	68.16	38.36	51.17	96.20	68.42	55.70
	$\lambda : 25, \mu : 25, \nu : 8$	67.91	37.75	51.14	96.14	68.75	54.21
	$\lambda : 25, \mu : 25, \nu : 16$	67.77	36.70	51.20	96.06	68.57	51.05
	$\lambda : 5, \mu : 25, \nu : 4$	64.12	31.37	49.83	95.56	66.17	42.56
	$\lambda : 10, \mu : 25, \nu : 4$	66.67	34.81	50.76	95.68	67.61	47.33
	$\lambda : 15, \mu : 25, \nu : 4$	67.49	36.91	51.40	96.10	67.95	51.72
	$\lambda : 20, \mu : 25, \nu : 4$	67.87	37.18	51.40	96.06	68.26	54.00
	$\lambda : 30, \mu : 25, \nu : 4$	67.99	38.71	51.11	96.16	68.68	56.05
	$\lambda : 35, \mu : 25, \nu : 4$	67.78	38.52	50.79	96.38	68.39	57.13
	$\lambda : 40, \mu : 25, \nu : 4$	67.25	38.08	50.85	96.34	68.69	56.29
	$\lambda : 45, \mu : 25, \nu : 4$	66.95	37.93	50.88	96.06	67.98	57.67
	$\lambda : 50, \mu : 25, \nu : 4$	66.51	37.79	50.11	96.10	67.74	57.23
	$\lambda : 25, \mu : 25, \nu : 4$	59.54	28.85	47.80	95.10	64.14	43.15
	$\lambda : 25, \mu : 25, \nu : 4$	66.36	35.47	50.32	96.04	67.45	51.64
	$\lambda : 25, \mu : 25, \nu : 4$	68.16	38.36	51.17	96.20	68.42	55.70
	$\lambda : 25, \mu : 25, \nu : 4$	68.56	38.80	51.75	96.30	68.60	55.75
	$\lambda : 25, \mu : 25, \nu : 4$	62.77	31.85	48.02	95.72	64.82	43.23
	$\lambda : 25, \mu : 25, \nu : 4$	67.79	38.25	51.57	96.04	68.84	55.38
	$\lambda : 25, \mu : 25, \nu : 4$	67.97	38.26	51.29	96.16	68.62	55.57
	$\lambda : 25, \mu : 25, \nu : 4$	67.87	38.43	51.51	96.08	68.52	54.53
	$\lambda : 25, \mu : 25, \nu : 4$	63.36	38.31	51.17	96.06	68.43	55.06
	$\lambda : 25, \mu : 25, \nu : 4$	54.52	37.92	51.32	96.10	67.99	54.82
	$\lambda : 25, \mu : 25, \nu : 4$	40.73	37.03	50.97	96.30	68.40	52.28
VICReg-exp	$\lambda : 1, \mu : 1, \nu : 2, \tau : 0.05$	67.74	37.53	51.44	96.36	68.41	52.12
	$\lambda : 1, \mu : 1, \nu : 2, \tau : 0.07$	67.64	38.00	51.42	96.46	68.60	54.16
	$\lambda : 1, \mu : 1, \nu : 2, \tau : 0.1$	67.84	38.25	51.07	96.44	68.12	55.94
	$\lambda : 1, \mu : 1, \nu : 2, \tau : 0.2$	65.09	36.64	49.65	96.54	67.12	56.37
	$\lambda : 1, \mu : 1, \nu : 2, \tau : 0.3$	60.67	31.22	48.04	95.80	64.28	46.96
	$\lambda : 1, \mu : 1, \nu : 2, \tau : 0.4$	57.46	26.54	46.25	96.02	62.33	41.90
	$\lambda : 1, \mu : 1, \nu : 0.1, \tau : 0.1$	55.12	24.73	45.68	95.44	61.82	39.71
	$\lambda : 1, \mu : 1, \nu : 0.5, \tau : 0.1$	64.87	36.51	49.69	96.16	66.82	55.30
	$\lambda : 1, \mu : 1, \nu : 1, \tau : 0.1$	66.84	38.25	50.85	96.24	68.34	57.12
	$\lambda : 1, \mu : 1, \nu : 4, \tau : 0.1$	68.08	38.03	51.34	96.40	69.28	53.72
	$\lambda : 1, \mu : 1, \nu : 8, \tau : 0.1$	67.80	37.20	51.57	96.46	68.61	52.15
	$\lambda : 1, \mu : 1, \nu : 16, \tau : 0.1$	66.68	35.02	50.94	96.00	67.81	47.49

Table S6: Top-1 on representations in all settings, continued.

Method	Hyperparameters	INet	iNat18	Places	EuroSat	SUN	Cars
VICReg-ctr	$\lambda : 1, \mu : 1, \nu : 1, \tau : 0.05$	65.54	35.00	50.15	95.88	67.62	49.63
	$\lambda : 1, \mu : 1, \nu : 1, \tau : 0.07$	66.32	35.72	50.69	96.10	68.16	51.66
	$\lambda : 1, \mu : 1, \nu : 1, \tau : 0.1$	66.09	35.26	50.80	96.42	68.32	50.72
	$\lambda : 1, \mu : 1, \nu : 1, \tau : 0.2$	64.06	33.16	50.48	95.98	67.40	44.91
	$\lambda : 1, \mu : 1, \nu : 1, \tau : 0.3$	62.06	30.80	49.53	96.22	66.08	43.24
	$\lambda : 1, \mu : 1, \nu : 1, \tau : 0.4$	60.17	28.76	48.78	95.90	64.92	41.13
	$\lambda : 1, \mu : 1, \nu : 0.1, \tau : 0.1$	61.66	31.05	49.40	96.10	66.47	44.12
	$\lambda : 1, \mu : 1, \nu : 0.5, \tau : 0.1$	65.47	34.63	50.71	96.20	67.55	48.05
	$\lambda : 1, \mu : 1, \nu : 2, \tau : 0.1$	65.99	34.77	50.63	95.98	67.60	51.14
	$\lambda : 1, \mu : 1, \nu : 4, \tau : 0.1$	63.87	33.63	49.64	96.20	66.17	50.35
	$\lambda : 1, \mu : 1, \nu : 8, \tau : 0.1$	58.81	29.24	47.60	95.78	63.77	46.23
SimCLR	$d : 512, \tau : 0.05$	57.68	30.50	48.51	96.32	63.36	42.42
	$d : 512, \tau : 0.07$	62.79	33.50	50.56	96.18	66.34	43.76
	$d : 512, \tau : 0.1$	66.13	35.94	52.10	96.22	68.29	49.17
	$d : 512, \tau : 0.2$	66.35	35.60	51.96	96.64	68.17	49.68
	$d : 512, \tau : 0.3$	65.17	34.38	51.32	96.10	67.78	48.17
	$d : 512, \tau : 0.4$	63.54	33.29	50.71	96.22	67.39	48.31
	$d : 2048, \tau : 0.05$	57.84	30.82	48.64	96.34	64.07	41.97
	$d : 2048, \tau : 0.07$	62.73	33.30	50.57	96.56	66.03	44.99
	$d : 2048, \tau : 0.1$	66.30	36.25	51.79	96.40	67.99	48.95
	$d : 2048, \tau : 0.2$	66.71	36.56	51.82	96.52	68.52	50.47
	$d : 2048, \tau : 0.3$	65.29	34.90	51.32	96.30	67.40	49.16
	$d : 2048, \tau : 0.4$	63.52	33.35	50.92	96.42	66.89	48.59
	$d : 2048, \tau : 0.05$	59.49	31.13	48.80	95.94	64.11	42.46
	$d : 2048, \tau : 0.07$	63.51	34.14	50.75	96.42	66.44	45.18
	$d : 2048, \tau : 0.1$	67.14	37.80	52.29	96.62	69.06	51.47
	$d : 2048, \tau : 0.15$	68.48	39.20	52.37	96.46	68.92	54.43
	$d : 2048, \tau : 0.2$	68.27	38.48	52.29	96.46	69.19	55.22
	$d : 2048, \tau : 0.3$	67.48	37.07	51.72	96.58	68.30	51.92
	$d : 2048, \tau : 0.4$	66.44	35.87	51.58	96.44	68.15	49.76
	$d : 2048, \tau : 0.15$	68.33	38.93	52.56	96.40	69.21	54.86
	$d : 2048, \tau : 0.15$	68.13	39.09	52.42	96.42	69.15	54.83
	$d : 2048, \tau : 0.15$	66.47	38.80	52.81	96.58	69.03	55.19
	$d : 2048, \tau : 0.15$	59.62	38.86	52.69	96.62	69.07	55.47
	$d : 2048, \tau : 0.15$	47.58	39.03	52.70	96.16	68.77	54.96
	$d : 2048, \tau : 0.15$	32.27	38.70	52.62	96.18	68.53	54.67
	$d : 2048, \tau : 0.15$	66.37	36.06	51.62	96.84	68.22	52.17
	$d : 2048, \tau : 0.15$	67.96	38.12	52.33	96.44	68.54	53.86
	$d : 2048, \tau : 0.15$	68.32	38.44	52.42	96.80	69.08	54.63
	$d : 2048, \tau : 0.15$	68.48	39.20	52.37	96.46	68.92	54.43
	$d : 2048, \tau : 0.15$	68.41	38.77	52.42	96.24	68.65	55.81
	$d : 2048, \tau : 0.15$	68.12	38.45	52.33	96.64	68.41	54.30

Table S7: Top-1 on embeddings in all settings.

Method	Hyperparameters	INet	iNat18	Places	EuroSat	SUN	Cars
VICReg	$\lambda: 25, \mu: 25, \nu: 0.3$	26.35	0.95	21.48	65.10	31.23	5.10
	$\lambda: 25, \mu: 25, \nu: 0.4$	30.54	1.39	21.07	63.80	31.60	5.24
	$\lambda: 25, \mu: 25, \nu: 0.5$	36.92	1.85	24.77	69.92	35.24	5.12
	$\lambda: 25, \mu: 25, \nu: 0.6$	47.60	4.34	35.48	87.84	47.62	7.56
	$\lambda: 25, \mu: 25, \nu: 0.7$	51.26	6.14	36.43	88.58	49.29	8.82
	$\lambda: 25, \mu: 25, \nu: 0.8$	54.39	7.87	38.04	88.44	51.88	9.76
	$\lambda: 25, \mu: 25, \nu: 0.9$	55.66	8.76	38.97	89.42	53.14	10.31
	$\lambda: 25, \mu: 25, \nu: 1$	56.33	9.56	39.65	89.76	53.39	10.53
	$\lambda: 25, \mu: 25, \nu: 2$	58.65	12.08	41.73	90.88	56.35	11.75
	$\lambda: 25, \mu: 25, \nu: 4$	59.71	13.47	42.72	91.28	57.26	11.95
	$\lambda: 25, \mu: 25, \nu: 8$	59.58	14.18	43.22	90.96	57.43	11.58
	$\lambda: 25, \mu: 25, \nu: 16$	59.22	14.63	43.48	91.34	57.75	11.99
	$\lambda: 5, \mu: 25, \nu: 4$	53.78	12.80	42.35	92.22	55.41	10.46
	$\lambda: 10, \mu: 25, \nu: 4$	57.94	14.36	43.61	91.60	57.89	11.30
	$\lambda: 15, \mu: 25, \nu: 4$	59.20	14.77	43.63	91.40	57.36	12.00
	$\lambda: 20, \mu: 25, \nu: 4$	59.73	14.20	43.31	91.66	57.04	11.95
	$\lambda: 30, \mu: 25, \nu: 4$	59.09	12.35	42.21	90.34	56.21	11.55
	$\lambda: 35, \mu: 25, \nu: 4$	58.23	11.34	41.02	90.16	54.97	11.69
	$\lambda: 40, \mu: 25, \nu: 4$	56.82	10.15	40.19	89.94	54.50	10.76
	$\lambda: 45, \mu: 25, \nu: 4$	55.22	9.26	39.02	90.00	53.07	10.98
	$\lambda: 50, \mu: 25, \nu: 4$	53.75	8.29	37.87	89.76	52.16	10.60
	$\lambda: 25, \mu: 25, \nu: 4$	51.53	12.97	40.84	91.64	54.26	13.13
	$\lambda: 25, \mu: 25, \nu: 4$	57.57	13.60	42.40	91.64	56.43	12.47
	$\lambda: 25, \mu: 25, \nu: 4$	59.71	13.47	42.72	91.28	57.26	11.95
	$\lambda: 25, \mu: 25, \nu: 4$	56.22	9.92	40.18	88.36	53.69	8.46
	$\lambda: 25, \mu: 25, \nu: 4$	36.22	2.48	29.59	85.06	42.46	7.55
	$\lambda: 25, \mu: 25, \nu: 4$	59.33	13.22	42.86	90.76	57.21	11.28
	$\lambda: 25, \mu: 25, \nu: 4$	59.51	13.37	42.69	91.34	56.66	11.53
	$\lambda: 25, \mu: 25, \nu: 4$	59.70	13.64	43.37	90.96	57.32	11.89
	$\lambda: 25, \mu: 25, \nu: 4$	59.03	14.00	43.10	91.50	57.44	12.27
	$\lambda: 25, \mu: 25, \nu: 4$	56.37	14.10	43.23	91.36	57.69	12.52
	$\lambda: 25, \mu: 25, \nu: 4$	49.96	12.36	41.93	91.52	57.13	11.37
VICReg-exp	$\lambda: 1, \mu: 1, \nu: 2, \tau: 0.05$	58.19	12.56	41.93	91.82	57.13	10.19
	$\lambda: 1, \mu: 1, \nu: 2, \tau: 0.07$	58.53	12.10	42.03	91.64	56.85	10.96
	$\lambda: 1, \mu: 1, \nu: 2, \tau: 0.1$	57.41	10.78	40.89	90.44	55.24	10.73
	$\lambda: 1, \mu: 1, \nu: 2, \tau: 0.2$	47.80	5.02	34.67	88.92	47.14	8.39
	$\lambda: 1, \mu: 1, \nu: 2, \tau: 0.3$	25.14	1.02	22.04	77.22	32.66	5.17
	$\lambda: 1, \mu: 1, \nu: 2, \tau: 0.4$	19.24	0.75	20.08	75.70	30.22	5.77
	$\lambda: 1, \mu: 1, \nu: 0.1, \tau: 0.1$	12.03	0.51	16.95	70.18	26.06	4.15
	$\lambda: 1, \mu: 1, \nu: 0.5, \tau: 0.1$	47.33	4.78	33.88	87.32	46.90	8.87
	$\lambda: 1, \mu: 1, \nu: 1, \tau: 0.1$	53.72	7.74	38.04	89.46	52.37	9.76
	$\lambda: 1, \mu: 1, \nu: 4, \tau: 0.1$	58.87	12.22	42.27	91.34	56.97	10.93
	$\lambda: 1, \mu: 1, \nu: 8, \tau: 0.1$	58.09	12.70	42.56	91.58	56.90	10.42
	$\lambda: 1, \mu: 1, \nu: 16, \tau: 0.1$	57.24	14.01	43.18	92.38	57.36	11.18

Table S8: Top-1 on embeddings in all settings, continued.

Method	Hyperparameters	INet	iNat18	Places	EuroSat	SUN	Cars
VICReg-ctr	$\lambda: 1, \mu: 1, \nu: 1, \tau: 0.05$	50.26	10.83	38.54	89.54	52.73	11.39
	$\lambda: 1, \mu: 1, \nu: 1, \tau: 0.07$	50.99	9.81	38.43	90.06	53.90	10.53
	$\lambda: 1, \mu: 1, \nu: 1, \tau: 0.1$	48.27	7.67	36.78	88.02	51.45	10.21
	$\lambda: 1, \mu: 1, \nu: 1, \tau: 0.2$	36.77	3.20	30.01	84.12	43.85	6.68
	$\lambda: 1, \mu: 1, \nu: 1, \tau: 0.3$	25.92	1.51	23.93	77.54	36.57	4.46
	$\lambda: 1, \mu: 1, \nu: 1, \tau: 0.4$	17.69	0.70	18.19	69.00	28.30	3.73
	$\lambda: 1, \mu: 1, \nu: 0.1, \tau: 0.1$	26.90	1.65	24.69	75.46	36.72	4.86
	$\lambda: 1, \mu: 1, \nu: 0.5, \tau: 0.1$	44.31	5.81	34.82	87.84	49.23	9.25
	$\lambda: 1, \mu: 1, \nu: 2, \tau: 0.1$	49.31	8.71	37.64	89.16	51.98	10.78
	$\lambda: 1, \mu: 1, \nu: 4, \tau: 0.1$	46.43	8.38	36.11	89.44	50.27	10.25
	$\lambda: 1, \mu: 1, \nu: 8, \tau: 0.1$	38.33	6.21	32.10	86.44	44.07	8.97
SimCLR	$d: 512, \tau: 0.05$	43.05	20.48	37.83	94.12	55.20	22.57
	$d: 512, \tau: 0.07$	49.69	21.28	41.82	93.48	58.23	21.38
	$d: 512, \tau: 0.1$	54.45	17.49	42.92	91.88	57.86	16.55
	$d: 512, \tau: 0.2$	50.24	8.36	39.22	88.42	51.94	11.58
	$d: 512, \tau: 0.3$	45.77	6.36	36.55	87.16	48.59	10.20
	$d: 512, \tau: 0.4$	41.14	4.81	34.04	84.76	45.36	9.10
	$d: 2048, \tau: 0.05$	43.31	20.51	38.16	94.44	55.40	23.24
	$d: 2048, \tau: 0.07$	49.60	21.51	41.93	93.88	59.05	22.43
	$d: 2048, \tau: 0.1$	54.48	17.92	43.00	92.86	59.53	18.22
	$d: 2048, \tau: 0.2$	50.72	8.65	39.64	89.44	54.47	12.96
	$d: 2048, \tau: 0.3$	43.32	5.84	36.51	87.62	50.85	11.33
	$d: 2048, \tau: 0.4$	41.15	5.02	34.26	85.52	48.16	10.87
	$d: 2048, \tau: 0.05$	44.61	21.33	39.20	94.08	56.15	22.82
	$d: 2048, \tau: 0.07$	51.54	21.50	42.64	94.40	59.87	21.30
	$d: 2048, \tau: 0.1$	56.51	16.68	43.39	92.26	59.10	17.56
	$d: 2048, \tau: 0.15$	56.89	10.35	41.21	90.38	56.37	14.04
	$d: 2048, \tau: 0.2$	54.18	7.16	39.42	87.94	54.24	11.47
	$d: 2048, \tau: 0.3$	49.19	4.98	37.12	87.14	50.85	10.33
	$d: 2048, \tau: 0.4$	44.72	3.89	35.11	86.00	48.38	9.49
	$d: 2048, \tau: 0.15$	57.06	10.12	41.18	89.52	56.20	13.19
	$d: 2048, \tau: 0.15$	56.72	10.17	41.38	90.08	56.40	13.23
	$d: 2048, \tau: 0.15$	56.14	10.26	41.47	89.12	56.74	13.18
	$d: 2048, \tau: 0.15$	48.98	10.03	41.48	89.84	56.15	13.47
	$d: 2048, \tau: 0.15$	35.92	10.03	41.33	89.74	56.38	13.87
	$d: 2048, \tau: 0.15$	28.26	9.68	41.22	86.88	56.02	13.13
	$d: 2048, \tau: 0.15$	52.65	9.98	40.47	90.20	55.38	13.72
	$d: 2048, \tau: 0.15$	55.98	9.88	40.84	89.00	55.43	13.41
	$d: 2048, \tau: 0.15$	56.43	10.03	41.04	89.60	56.23	13.89
	$d: 2048, \tau: 0.15$	56.89	10.35	41.21	90.38	56.37	14.04
	$d: 2048, \tau: 0.15$	56.65	10.30	41.40	89.22	56.42	13.79
	$d: 2048, \tau: 0.15$	56.56	10.60	41.61	90.14	56.82	13.90

Table S9: Rank after projector in all settings

Method	Hyperparameters	INet	iNat18	Places	EuroSat	SUN	Cars
VICReg	$\lambda: 25, \mu: 25, \nu: 0.3$	102	38	44	15	32	7
	$\lambda: 25, \mu: 25, \nu: 0.4$	230	93	129	89	98	13
	$\lambda: 25, \mu: 25, \nu: 0.5$	374	136	206	120	163	20
	$\lambda: 25, \mu: 25, \nu: 0.6$	612	261	336	229	266	39
	$\lambda: 25, \mu: 25, \nu: 0.7$	831	383	468	367	374	59
	$\lambda: 25, \mu: 25, \nu: 0.8$	953	449	539	429	436	77
	$\lambda: 25, \mu: 25, \nu: 0.9$	1034	493	587	478	478	88
	$\lambda: 25, \mu: 25, \nu: 1$	1088	531	631	515	517	100
	$\lambda: 25, \mu: 25, \nu: 2$	1443	726	849	693	724	162
	$\lambda: 25, \mu: 25, \nu: 4$	1809	948	1111	856	955	210
	$\lambda: 25, \mu: 25, \nu: 8$	1921	1055	1248	871	1076	258
	$\lambda: 25, \mu: 25, \nu: 16$	1938	1087	1276	925	1119	307
	$\lambda: 5, \mu: 25, \nu: 4$	1938	1101	1338	963	1172	382
	$\lambda: 10, \mu: 25, \nu: 4$	1945	1096	1308	969	1156	353
	$\lambda: 15, \mu: 25, \nu: 4$	1940	1096	1281	910	1127	325
	$\lambda: 20, \mu: 25, \nu: 4$	1942	1050	1241	893	1070	270
	$\lambda: 30, \mu: 25, \nu: 4$	1521	782	920	725	772	170
	$\lambda: 35, \mu: 25, \nu: 4$	1279	637	757	607	627	129
	$\lambda: 40, \mu: 25, \nu: 4$	1080	532	635	528	525	111
	$\lambda: 45, \mu: 25, \nu: 4$	910	447	526	454	431	89
	$\lambda: 50, \mu: 25, \nu: 4$	778	376	448	378	360	74
	$\lambda: 25, \mu: 25, \nu: 4$	1409	891	996	814	890	353
	$\lambda: 25, \mu: 25, \nu: 4$	1652	936	1070	838	932	275
	$\lambda: 25, \mu: 25, \nu: 4$	1809	948	1111	856	955	210
	$\lambda: 25, \mu: 25, \nu: 4$	1422	649	813	533	650	91
	$\lambda: 25, \mu: 25, \nu: 4$	101	44	46	21	37	11
	$\lambda: 25, \mu: 25, \nu: 4$	1822	960	1130	840	962	222
	$\lambda: 25, \mu: 25, \nu: 4$	1815	948	1107	856	947	218
	$\lambda: 25, \mu: 25, \nu: 4$	1729	913	1066	814	911	216
	$\lambda: 25, \mu: 25, \nu: 4$	1587	860	1009	808	864	245
	$\lambda: 25, \mu: 25, \nu: 4$	1385	758	882	716	767	230
	$\lambda: 25, \mu: 25, \nu: 4$	975	509	613	508	526	144
VICReg-exp	$\lambda: 1, \mu: 1, \nu: 2, \tau: 0.05$	1007	531	637	501	552	143
	$\lambda: 1, \mu: 1, \nu: 2, \tau: 0.07$	1002	521	626	515	535	133
	$\lambda: 1, \mu: 1, \nu: 2, \tau: 0.1$	923	473	564	476	472	120
	$\lambda: 1, \mu: 1, \nu: 2, \tau: 0.2$	399	192	233	203	190	37
	$\lambda: 1, \mu: 1, \nu: 2, \tau: 0.3$	64	30	37	21	31	8
	$\lambda: 1, \mu: 1, \nu: 2, \tau: 0.4$	19	12	10	6	8	4
	$\lambda: 1, \mu: 1, \nu: 0.1, \tau: 0.1$	9	7	5	4	5	3
	$\lambda: 1, \mu: 1, \nu: 0.5, \tau: 0.1$	375	181	217	192	177	32
	$\lambda: 1, \mu: 1, \nu: 1, \tau: 0.1$	637	314	380	341	312	66
	$\lambda: 1, \mu: 1, \nu: 4, \tau: 0.1$	1002	529	629	518	537	139
	$\lambda: 1, \mu: 1, \nu: 8, \tau: 0.1$	1049	556	673	547	581	158
	$\lambda: 1, \mu: 1, \nu: 16, \tau: 0.1$	1326	734	876	707	771	209

Table S10: Rank after projector in all settings, continued.

Method	Hyperparameters	INet	iNat18	Places	EuroSat	SUN	Cars
VICReg-ctr	$\lambda : 1, \mu : 1, \nu : 1, \tau : 0.05$	382	225	252	208	221	69
	$\lambda : 1, \mu : 1, \nu : 1, \tau : 0.07$	279	164	183	155	161	50
	$\lambda : 1, \mu : 1, \nu : 1, \tau : 0.1$	169	101	114	98	100	35
	$\lambda : 1, \mu : 1, \nu : 1, \tau : 0.2$	48	32	35	33	32	13
	$\lambda : 1, \mu : 1, \nu : 1, \tau : 0.3$	23	17	18	18	17	7
	$\lambda : 1, \mu : 1, \nu : 1, \tau : 0.4$	13	10	10	11	10	5
	$\lambda : 1, \mu : 1, \nu : 0.1, \tau : 0.1$	23	17	18	17	17	8
	$\lambda : 1, \mu : 1, \nu : 0.5, \tau : 0.1$	96	62	68	61	60	23
	$\lambda : 1, \mu : 1, \nu : 2, \tau : 0.1$	252	146	166	139	144	46
	$\lambda : 1, \mu : 1, \nu : 4, \tau : 0.1$	309	177	204	171	176	54
	$\lambda : 1, \mu : 1, \nu : 8, \tau : 0.1$	317	185	214	175	186	59
SimCLR	$d : 512, \tau : 0.05$	109	106	105	76	106	93
	$d : 512, \tau : 0.07$	164	149	150	100	148	114
	$d : 512, \tau : 0.1$	244	184	203	130	188	106
	$d : 512, \tau : 0.2$	151	95	117	84	102	41
	$d : 512, \tau : 0.3$	88	58	67	55	60	25
	$d : 512, \tau : 0.4$	64	42	48	41	43	18
	$d : 2048, \tau : 0.05$	111	107	106	77	107	94
	$d : 2048, \tau : 0.07$	165	150	150	103	149	114
	$d : 2048, \tau : 0.1$	247	185	204	129	190	107
	$d : 2048, \tau : 0.2$	165	103	128	95	112	43
	$d : 2048, \tau : 0.3$	10	30	3	55	65	26
	$d : 2048, \tau : 0.4$	64	42	48	41	43	19
	$d : 2048, \tau : 0.05$	123	119	117	85	118	103
	$d : 2048, \tau : 0.07$	197	173	176	117	173	129
	$d : 2048, \tau : 0.1$	314	220	240	161	223	112
	$d : 2048, \tau : 0.15$	299	173	209	141	184	61
	$d : 2048, \tau : 0.2$	221	122	151	107	130	40
	$d : 2048, \tau : 0.3$	128	72	90	66	79	26
	$d : 2048, \tau : 0.4$	72	49	64	49	55	19
	$d : 2048, \tau : 0.15$	302	173	211	147	185	61
	$d : 2048, \tau : 0.15$	300	173	209	142	182	62
	$d : 2048, \tau : 0.15$	300	174	209	144	182	61
	$d : 2048, \tau : 0.15$	301	174	209	148	184	61
	$d : 2048, \tau : 0.15$	301	174	207	143	184	61
	$d : 2048, \tau : 0.15$	301	175	207	146	182	61
	$d : 2048, \tau : 0.15$	12	16	32	145	14	4
	$d : 2048, \tau : 0.15$	293	173	212	140	185	65
	$d : 2048, \tau : 0.15$	295	173	208	140	181	62
	$d : 2048, \tau : 0.15$	299	173	209	141	184	61
	$d : 2048, \tau : 0.15$	299	172	207	143	181	61
	$d : 2048, \tau : 0.15$	294	170	202	141	177	59