Unsupervised Embedding Quality Evaluation

Anton Tsitsulin 1  Marina Munkhoeva 2  Bryan Perozzi 1

Abstract

Unsupervised learning has recently significantly gained in popularity, especially with deep learning-based approaches. Despite numerous successes and approaching supervised-level performance on a variety of academic benchmarks, it is still hard to train and evaluate SSL models in practice due to the unsupervised nature of the problem. Even with networks trained in a supervised fashion, it is often unclear whether they will perform well when transferred to another domain.

Past works are generally limited to assessing the amount of information contained in embeddings, which is most relevant for self-supervised learning of deep neural networks. This work chooses to follow a different approach: can we quantify how easy it is to linearly separate the data in a stable way? We survey the literature and uncover three methods that could be potentially used for evaluating quality of representations. We also introduce one novel method based on recent advances in understanding the high-dimensional geometric structure of self-supervised learning.

We conduct extensive experiments and study the properties of these metrics and ones introduced in the previous work. Our results suggest that while there is no free lunch, there are metrics that can robustly estimate embedding quality in an unsupervised way.

1. Introduction

With proliferation of unsupervised and self-supervised deep learning methods in the recent years, there is an increasing need to quantify the quality of representations produced by such methods. Across different domains, this is commonly done with training linear classifiers (probes) against known labels (Perozzi et al., 2014; Chen et al., 2020). However, in unsupervised settings there are no labels to begin with. How can we do model selection, optimize methods’ hyperparameters, or even verify the method worked at all?

In search of such metrics, we turn our attention to different sub-fields of numerical linear algebra, machine learning and optimization, and high-dimensional probability. We identify three promising candidate metrics and introduce one based on the expected distribution of embedding distances. We then proceed to test them on two conceptually novel domains: supervised model selection and shallow single-layer graph embedding learning.

Our experimental results indicate there is no “free lunch”—a metric that is universally dominating—thus calling for a comprehensive suite of evaluation metrics. Despite that, metrics introduced in this work exhibit, like stable rank and coherence, display stronger correlation to downstream task performance of the supervised models, are more computationally stable, and suit shallow embedding models much better than state-of-the-art ones.

We summarize our key contributions as follows:
- We identify three different perspectives on evaluation of embedding quality in unsupervised manner and introduce four metrics based on these perspectives.
- We experimentally study two novel settings for embedding quality evaluation, showing that standard metrics often fail when shallow models are being studied.
- We conduct a study on computational stability of all metrics and identify the minimum viable sample sizes.
- We demonstrate that the proposed metrics are at least as effective as state-of-the-art ones in terms of downstream quality prediction while having more intuitive behavior for shallow embedding models.

2. Related Work

The literature on evaluating representations in unsupervised way is still sparse. Arguably, dimensional collapse (Hua et al., 2021) has sparked initial interest in the area. In dimensional collapse, some dimensions become non-meaningful (collapse) during training. Because of that problem, three concurrent metrics, which we introduce below, all study the problem of measuring such collapse from different angles.

---

1Google Research, New York, USA 2Max Planck Institute for Intelligent Systems, Tübingen, Germany. Correspondence to: Anton Tsitsulin <tsitsulin@google.com>.

Proceedings of the 2nd Annual Workshop on Topology, Algebra, and Geometry in Machine Learning (TAG-ML) at the 40th International Conference on Machine Learning, Honolulu, Hawaii, USA. 2023. Copyright 2023 by the author(s).
We now study three different perspectives on estimating $\alpha$ with convention of $
abla_i \propto 1^\alpha$. Logarithmic decay of the spectrum with slope $\alpha = 1$ was recently proven to provide the best generalization in infinite-dimensional analysis of linear regression (Bartlett et al., 2020). In practice, a simple linear regression estimator on a log-log scale is used to estimate the value of $\alpha$. This approach for estimating the power-law exponent is considered inaccurate (Clauset et al., 2009).

3. Three Perspectives on Embedding Quality

How hard it is to find a suitable transformation from representations to the targets of the downstream task asks: Do embedding carry as much information as their size allows? Do embedding carry as much information as different representations: Do embedding carry as much information as their size allows? However, there are different questions worth answering. This paper introduces four novel metrics for embedding quality evaluation based on different perspectives on the embedding quality.

The following section pursues the linear classifier perspective on representation quality (Mohri & Talwalkar, 2011). It asks: How hard it is to find a suitable transformation from the representations to the targets of the downstream task? We show that this is an inherent property of the representations themselves (and the target matrix too, if it’s not a classification task).

3.1. Linear Classifier Perspective

Let our downstream task be a classification with a target matrix $Y \in \{0, 1\}^{n \times c}$ and a linear probe $h = XW + b$ with weight matrix $W$ and bias vector $b$. In what follows, we argue that it is easier to find $h$ that yields high accuracy when applied to the input matrix $X$ with higher coherence.

Without loss of generality, we can drop the bias term. For the ease of exposition, we will adopt the Mean-Squared Error loss ($\mathcal{L} = \|Y - XW\|_F^2$) for a downstream task. The optimal weight matrix will then depend on the target and representation matrices, i.e. from the derivative condition $X^\top Y = X^\top XW$. Given some $A \in \ker(X)$, i.e. a matrix comprised of vectors from the null space of $X$, we rewrite the condition as $X^\top Y = X^\top (A + X^\top Y)$ and get $W^* = X^\top Y + A$ for any $A \in \ker(X)$.

Assuming we can always find an optimal weight matrix, to minimize the loss $\mathcal{L}$, the representations $X$ should be aligned with the target matrix $Y$, i.e. the left singular vectors $U$ of $X = U \Sigma V^\top$ should span $\ker(Y)$, i.e. the left singular vectors $U$ of $X = U \Sigma V^\top$ should span $\ker(Y)$, i.e. $U_Y$. In what follows, we will use $V = I_c$ when $Y$ is a classification target matrix.

Plugging in the optimal $W^*$ into the loss, we get:

$$||Y - X(X^\top Y + A)||_F^2 = ||Y - U \Sigma \Sigma U^\top Y||_F^2 = \| (I - U_U U^\top) Y \|_F^2 = \| Y \|_F^2 - \| U_U^\top U_Y \Sigma Y \|_F^2,$$

where $I_d \in \mathbb{R}^{n \times n}$ with $d$ ones on the diagonal, and the minimum is reached whenever columns in $U$ are aligned with columns in $U_Y$.

Intuitively, if the representation dimensionality is larger than number of classes in the downstream task, i.e. $d > c$, and $X$ has full rank (a consequence of most methods being spectral embedding), then the representation basis covers the target basis with high probability. However, to quantify the extent of this coverage, we will need to introduce a notion of incoherence.

Definition 3.1 ($\mu_0$-incoherence). Given matrix $M \in \mathbb{R}^{n_1 \times n_2}$ with rank-$r$ and SVD $M = U \Sigma V^\top$, $M$ is said to satisfy the standard incoherence condition with parameter $\mu_0$ if

$$\frac{\max_{1 \leq i \leq n_1} \| U^\top e_i \|_2}{\sqrt{\frac{\mu_0 r}{n_1}}} \leq \sqrt{\frac{\mu_0 r}{n_1}}, \quad \frac{\max_{1 \leq i \leq n_2} \| V^\top e_j \|_2}{\sqrt{\frac{\mu_0 r}{n_2}}} \leq \sqrt{\frac{\mu_0 r}{n_2}},$$

where $e_i$ is the $i$-th standard basis vector of a respective dimension. Note that $1 \leq \mu_0 \leq \max(n_1, n_2)/r$.

Informally, standard incoherence characterizes the extent of alignment of the singular vectors to the standard basis.
Incoherence is typically used in low-rank matrix completion problems to estimate a complexity of matrix recovery (Mohri & Talwalkar, 2011). In our setting, lower incoherence will be indicative of high alignment with target matrix and, thus, better performance.

Ideally, if we had access to the targets, we could use joint incoherence $\mu_1(Z, Y)$ to measure the alignment directly. However, we will need to rely on the standard coherence $\mu_0(Z)$ which measures alignment to the standard basis. Our experiments show that there is indeed a correlation between standard incoherence of the representations and performance on the downstream tasks (almost perfect in some cases).

### 3.2. Numerical Linear Algebra Perspective

Numerical linear algebra provides us with more tools for analysing behaviors of linear classifiers. One of the classic ones is the condition number, or, in the case of non-square matrices, its generalized version (Ben-Israel, 1966). For example, $\kappa_2$ is used to detect multicollinearity in linear and logistic regression (Belsley et al., 2005).

**Definition 3.2.** Pseudo-condition number number of a matrix $M$ with SVD $M = U\Sigma V^T$ is defined as $\kappa_p(M) = \|M\|_p \|M^T\|_p^{p/2} \sigma_1 / \sigma_n$.

We are particularly interested in $\kappa_2$, since it is easily computable with SVD, as the pseudo-inverse of $M$ is $(M^T M)^{-1} = U\Sigma^{-1} V^T$, meaning $\|M^T\|_2 = 1/\sigma_n$.

In the analysis of linear regression, $\kappa_2$ can be used to bound the sensitivity of the system to the change in the input. Consider a linear system $(A + \Delta A)x = b + \Delta b$. Then,

$$\frac{\|\hat{x} - x\|}{\|x\|} \leq \kappa(A) \frac{1}{1 - \kappa(A)} \left( \frac{\|\Delta A\|}{\|A\|} + \frac{\|\Delta b\|}{\|b\|} \right).$$

We use $\kappa_2$ to measure stability of learned representations.

#### 3.2.1. Stable Rank

Stable rank (also called effective rank or intrinsic dimension of a matrix) is another fundamental quality in numerical analysis of random matrices.

**Definition 3.3.** Numerical rank of a matrix $M$ is defined as $r(M) = \|M\|_F / \|M\|_2$.

Note that $r(M) \leq \text{rank}(M)$, and that bound is sharp. Stable rank is a useful tool that guides fundamental numerical problems, including matrix sampling and covariance estimation.

Let us restate Theorem 1.1 from Rudelson & Vershynin (2007):

**Theorem 3.4.** Let $A$ be an $n \times d$ matrix with stable rank $r$. Let $\varepsilon, \delta \in (0, 1)$, and let $m \leq n$ be an integer such that

$$m \geq C \left( \frac{d}{\varepsilon^2 \delta} \right) \log \left( \frac{d}{\varepsilon^2 \delta} \right).$$

Consider a $m \times d$ matrix $\tilde{A}$, which consists of $m$ normalized rows of $A$ picked independently with replacement, with probabilities proportional to the squares of their Euclidean lengths. Then with probability at least $1 - 2\exp(-c/\delta)$ the following holds. For a positive integer $k$, let $P_k$ be the orthogonal projection onto the top $k$ left singular vectors of $A$. Then,

$$\|A - AP_k\| = \sigma_{k+1}(A) + \varepsilon \|A\|_2.$$

This suggests that the numerical rank determines how hard it is to estimate the matrix by subsampling its rows. Intuitively, a well-distributed representations should be hard to estimate; we will observe that this is indeed the case in practice.

### 3.3. High-dimensional Probability Perspective

In self-supervised learning, Assran et al. (2023) shows that several contrastive learning methods try to distribute representations equally in the space. High-dimensional probability can provide us with an estimate of pairwise distances when embeddings are distributed uniformly on a $d$-dimensional unit sphere $S^d$.

Given $L_2$ normalized embeddings $W \in \mathbb{R}^{n \times d}$, a measure of clustering can be defined using the norm of the pairwise dot product matrix $Q = \|WW^T\|_F$. Since the expected dot product of high-dimensional isotropic random vectors $(x, y) \sim \mathcal{N}(0, 1)$ (Vershynin, 2018, Remark 3.2.5), we can estimate $\mathbb{E}[Q] = n + n(n-1)/d$. The maximum metric value $Q = n^2$ can only be achieved in the collapsed case. Combining all normalizations to get a metric upper-bounded that is upper-bounded by 1, we get:

**Definition 3.5.**

$$\text{SelfCluster}(W) = \frac{\|WW^T\|_F - n - \frac{n(n-1)}{d}}{n^2 - n - \frac{n(n-1)}{d}}$$

$$= \frac{d\|WW^T\|_F - n(d + n - 1)}{(d-1)(n-1)n}.$$
4. Experiments

In contrast to previous work (Agrawal et al., 2022; Garrido et al., 2022), we shift our attention from self-supervised learning to novel, more generally applicable settings. We experimentally study proposed metrics on two novel use-cases: (i) supervised representation learning with deep neural networks and (ii) unsupervised graph embeddings. Supervised representation learning allows us to gain insights into performance of semi-supervised learning systems. Graph embedding, on the other hand, has very different architecture—shallow single-layer network—and optimization.

Section 4.1.2 further provides a novel study on computational stability of different embedding quality evaluation metrics. Stability is important for many practical application, since the most computationally stable metrics can be even computed during training for monitoring purposes.

4.1. Supervised Network Performance Prediction

We used Wightman (2019) repository of supervised PyTorch models, accessed May 2023. (Deng et al., 2009) We ran inference of all available models, as permitted by GPU memory, on the validation set, and a subset of models\(^1\)—on the full training set. Inference was performed on a single 16-core machine with NVIDIA RTX 4090 and 64Gb RAM.

4.1.1. Downstream Quality Correlation

Figures 1 and 2 present rank correlation of the different embedding quality metrics to downstream prediction quality on ImageNet, measured for training and validation set embeddings respectively. We do not report SelfCluster metric results on the training set because of its quadratic time complexity. Since RankMe is dependent on the dimensionality of the data, we normalize its values and call the metric RankMe\(^*\). This new metric has the range between 0 and 1, and represents relative utilization of the embedding space.

On the training set evaluation, \(\alpha\)-ReQ, NESum, pseudo-condition number, and coherence all show significant correlation to the test set performance. Out of these metrics, \(\alpha\)-ReQ is the only metric with significant outliers, possibly due to the power law estimation issues (Clauset et al., 2009). High stable rank, NESum, and coherence seem to indicate good test test performance of the model. Note that the models we selected for training set evaluation are pareto-optimal in terms of either parameter size or inference speed. This allowed us to significantly restrict the model set size without affecting representativeness of selected models.

On the validation set performance with expanded model set, the correlation between many metrics and test set performance drops to near-zero. This can be attributed to both expanded model set, which has many under-performing models as well as the general instability of the computation on the smaller example set. We further examine the computational stability considerations in the next section. Only NESum, stable rank and self clustering achieve significant correlation to the test set performance. Across both training and validation sets, NESum demonstrates strong downstream performance correlation while both variants of RankMe are not able to successfully predict supervised task performance.

4.1.2. Metric Stability

It is important to have stable metrics for embedding quality evaluation, especially in low-data regimes. Moreover, if a metric is stable up to very small batch sizes, it can be evaluated during training, greatly enhancing its usability.
Unsupervised Embedding Quality Evaluation

To do that, we sample embeddings for ImageNet training set with batch sizes from 128 to 65536, log-space \((2^7−2^{16})\) and compare the sampled metric value to the value computed on the whole dataset. The results are presented in Table 1. Numerical rank-based methods are among the most stable, followed by NESum. One advantage of RankMe over its numerical rank estimation counterpart is that it offers a strong lower-bound in terms of the sample size. Coherence appears to be strongly data-dependent and least stable.

4.2. Graph Embedding Quality Prediction

Graph embedding is a common way to solve many tasks arising in the graph mining domain from node classification, link prediction, and community detection. In the graph embedding process, each node in a graph is mapped to a vector in \(\mathbb{R}^d\), and distances in the embedding space should resemble some similarity metric defined between the nodes.

![Figure 2: Representation quality metrics on the ImageNet validation set of over 1000 pre-trained models. Spearman rank correlation \(\rho\) to the test set accuracy displayed per metric in the title. Methods introduced in this work are highlighted in colored bold.](image)

To understand the quality of the graph embeddings, we can use various metrics. The accuracy on the test set can be used to evaluate the quality of the embeddings. The following table shows the accuracy for different methods and batch sizes.

<table>
<thead>
<tr>
<th>Method</th>
<th>Batch Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Coherence</td>
<td>512</td>
</tr>
<tr>
<td>SelfCluster</td>
<td>1024</td>
</tr>
<tr>
<td>NESum</td>
<td>2048</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Method</th>
<th>Batch Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Coherence</td>
<td>4096</td>
</tr>
<tr>
<td>SelfCluster</td>
<td>4096</td>
</tr>
</tbody>
</table>

Table 1. Batch sizes needed to achieve constant multiplicative approximation factors compared to evaluation on the full ImageNet training set on XX networks. Additionally, we check that each metric lower-bounds the true value. The result can be either yes, no, or approximately.

<table>
<thead>
<tr>
<th>Approximation factor</th>
<th>0.5</th>
<th>0.7</th>
<th>0.9</th>
<th>0.95</th>
</tr>
</thead>
<tbody>
<tr>
<td>(\alpha)-ReQ</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NESum</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RankMe</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Stable rank</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cond. number</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Coherence</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 2: Dataset statistics. We report total number of nodes \(|V|\), average node degree \(\bar{d}\), number of labels \(|Y|\).

| Dataset      | \(|V|\) | \(|\bar{d}|\) | \(|Y|\) |
|--------------|-------|-----------|-------|
| Cora         | 19793 | 3.20      | 7     |
| Citeseer     | 3327  | 1.37      | 6     |
| PubMed       | 19717 | 2.25      | 3     |
| Amazon PC    | 13752 | 17.88     | 10    |
| Amazon Photo | 7650  | 15.57     | 8     |
| MSA-Physics  | 34493 | 7.19      | 5     |
| OGB-arXiv    | 169343| 6.84      | 40    |
| CIFAR-10     | 50000 | 99        | 10    |
| MNIST        | 60000 | 99        | 10    |

in the original graph (Tsitsulin et al., 2018). For an in-depth review of modern graph embedding approaches, readers are referred to Chami et al. (2022) survey.

For our experiments, we study representations of the DeepWalk (Perozzi et al., 2014) model as it is a de-facto standard in the field of unsupervised embedding of graphs with no features. We use 10 different graph datasets that include both natural and constructed graphs. We report the dataset statistics in Table 2 and provide a brief description below:

- Cora, Citeseer, and Pubmed (Sen et al., 2008) are citation networks; nodes represent papers connected by citation edges; features are bag-of-word abstracts, and labels represent paper topics. We use a re-processed version of Cora from (Shchur et al., 2018) due to errors in the processing of the original dataset.
- Amazon {PC, Photo} (Shchur et al., 2018) are two subsets of the Amazon co-purchase graph for the computers and photo sections of the website, where nodes represent goods with edges between ones frequently purchased together; node features are bag-of-word reviews, and class labels are product category.
- OGB-ArXiv (Hu et al., 2020) is a paper co-citation dataset based on arXiv papers indexed by the Microsoft Academic graph. Nodes are papers; edges are citations, and class labels indicate the main category of the paper.
Unsupervised Embedding Quality Evaluation

Table 3. Average Spearman rank correlation on two dataset corruption types: naïve (N) and component-preserving (C). We highlight datasets where there is a consistent correlation pattern, meaning the same sign and approximately the same magnitude of correlation. Methods proposed in this work exhibit stronger and more consistent correlation patterns across all datasets.

<table>
<thead>
<tr>
<th>metric</th>
<th>Cora</th>
<th>Citeseer</th>
<th>Pubmed</th>
<th>Amazon PC</th>
<th>Amazon Photo</th>
</tr>
</thead>
<tbody>
<tr>
<td>α-ReQ</td>
<td>-1.00</td>
<td>-1.00</td>
<td>-1.00</td>
<td>-1.00</td>
<td>-1.00</td>
</tr>
<tr>
<td>NESum</td>
<td>1.00</td>
<td>0.03</td>
<td>1.00</td>
<td>0.10</td>
<td>0.94</td>
</tr>
<tr>
<td>RankMe</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
</tr>
<tr>
<td>Stable rank</td>
<td>1.00</td>
<td>0.66</td>
<td>1.00</td>
<td>0.30</td>
<td>1.00</td>
</tr>
<tr>
<td>Cond. number</td>
<td>1.00</td>
<td>0.83</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
</tr>
<tr>
<td>SelfCluster</td>
<td>-1.00</td>
<td>-1.00</td>
<td>-1.00</td>
<td>-0.60</td>
<td>1.00</td>
</tr>
<tr>
<td>Coherence</td>
<td>1.00</td>
<td>1.00</td>
<td>0.90</td>
<td>1.00</td>
<td>0.94</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>metric</th>
<th>MSA-Physics</th>
<th>OGB-arXiv</th>
<th>MNIST</th>
<th>CIFAR-10</th>
</tr>
</thead>
<tbody>
<tr>
<td>α-ReQ</td>
<td>-0.70</td>
<td>0.94</td>
<td>-0.81</td>
<td>1.00</td>
</tr>
<tr>
<td>NESum</td>
<td>0.51</td>
<td>-0.98</td>
<td>0.84</td>
<td>-1.00</td>
</tr>
<tr>
<td>RankMe</td>
<td>0.59</td>
<td>-0.92</td>
<td>0.85</td>
<td>-1.00</td>
</tr>
<tr>
<td>Stable rank</td>
<td>0.52</td>
<td>-0.97</td>
<td>0.99</td>
<td>-0.99</td>
</tr>
<tr>
<td>Cond. number</td>
<td>0.52</td>
<td>-0.97</td>
<td>0.92</td>
<td>-1.00</td>
</tr>
<tr>
<td>SelfCluster</td>
<td>0.96</td>
<td>0.98</td>
<td>1.00</td>
<td>1.00</td>
</tr>
<tr>
<td>Coherence</td>
<td>0.97</td>
<td>0.99</td>
<td>0.90</td>
<td>1.00</td>
</tr>
</tbody>
</table>

We sparsify to a fixed number of edges corresponding to a target average node degree from the range $[1.1, 10]$. Some graphs in our studies have an average node degree $< 10$ naturally (cf. Table 2), in this case, we stop at that number. We embed each graph 10 times, run a downstream node classification 100 times, and average the result. We report Spearman rank correlation coefficient $\rho$ (Spearman, 1904) between the classification accuracy and each quality metric.

First, we report aggregated results across all datasets in Table 4. Surprisingly, most metrics completely revert the correlation sign between two sparsification strategies. Only SelfCluster and Coherence are aligned with the downstream evaluation, and between them, Coherence displays a near-perfect correlation with the downstream task performance.

Table 4. Average Spearman rank correlation on two dataset corruption types: naïve and component-preserving. We highlight rows where there is a consistent correlation pattern. Two methods introduced in this work strongly and consistently correlate with the downstream classification performance.

<table>
<thead>
<tr>
<th>metric</th>
<th>Naïve</th>
<th>Connected</th>
</tr>
</thead>
<tbody>
<tr>
<td>α-ReQ</td>
<td>-0.50</td>
<td>0.48</td>
</tr>
<tr>
<td>NESum</td>
<td>0.49</td>
<td>-0.71</td>
</tr>
<tr>
<td>RankMe</td>
<td>0.45</td>
<td>-0.47</td>
</tr>
<tr>
<td>Stable rank</td>
<td>0.56</td>
<td>-0.46</td>
</tr>
<tr>
<td>Cond. number</td>
<td>0.53</td>
<td>-0.43</td>
</tr>
<tr>
<td>SelfCluster</td>
<td>0.55</td>
<td>0.60</td>
</tr>
<tr>
<td>Coherence</td>
<td>0.95</td>
<td>0.99</td>
</tr>
</tbody>
</table>

Unsupervised Embedding Quality Evaluation

- CIFAR and MNIST (Krizhevsky et al., 2009; LeCun et al., 1998) are $\varepsilon$-nearest neighbor graphs with $\varepsilon$ such that the average node degree is 100.

Instead of changing the parameters of the model, we controllably change the quality of data itself. We sparsify each graph in two different ways:

- **Naïve sparsification**: we randomly pick $nd$ edges from the original edge set. This method may produce disconnected components, which are known to be difficult to embed correctly.

- **Component-preserving sparsification**: we first ensure the resulting graph is connected by sampling a random spanning tree. Then, we sample $n(d - 1)$ edges randomly and output the combined graph.

It is easy to see both versions create a controllably worse version of the data. As such, one could expect that representation quality degrades with the sparsity of the input graph, perhaps faster for the naïve algorithm, since it does not preserve the component information. As we will observe later, surprisingly, this is very much not the case for many embedding quality metrics we study.

We sparsify to a fixed number of edges corresponding to a target average node degree from the range $[1.1, 10]$. Some graphs in our studies have an average node degree $< 10$ naturally (cf. Table 2), in this case, we stop at that number. We embed each graph 10 times, run a downstream node classification 100 times, and average the result. We report Spearman rank correlation coefficient $\rho$ (Spearman, 1904) between the classification accuracy and each quality metric.
4.3. Metric Similarity

Since there are no clear winners in the experiments, it is important to use multiple metrics in real-world applications. Figure 3 presents pairwise correlations and kernel densities of different metrics on the training and validation sets of ImageNet. Overall, there are two clusters of the metrics: NESum and Stable rank as one and Coherence, α-ReQ, RankMe and condition number in another.

We conducted a large-scale study on two novel domains for unsupervised embedding quality evaluation: prediction of supervised test set performance and predicting performance of much simpler single-layer graph embedding methods. In case of supervised models, there seem to be no one-size-fits-all dominant solution, however, we identify numerically stable metrics that have strong correlation with downstream task performance. In the shallow model case, metrics introduced in this work show favorable downstream performance correlation consistently across 9 different datasets.

5. Conclusions

Is it possible to estimate embedding quality based on its statistical properties? This paper demonstrates it is possible in two scenarios outside of the known one of self-supervised learning. We introduced four new metrics based on ideas from numerical linear algebra, analysis of linear regression and high-dimensional probability.

References


A. Appendix.

Here we present the list of models we used for experimenting on the training and validation sets of ImageNet.

Training set models
beitv2_base_patch16_224.in1k_ft_in22k_in1k
covt_tiny
convnext_base.fb_in22k_ft_in1k_384
convnext_femto_ols.d1_in1k
dla46x_c
edgenext_base
edgenext_small
edgenext_x_small
edgenext_xx_small
eva_giant_patch14_560.m30m_ft_in22k_in1k
eva_large_patch14_196.in22k_ft_in22k_in1k
eva_large_patch14_336.in22k_ft_in22k_in1k
lcnet_050.ra2_in1k
lcnet_075.ra2_in1k
lcnet_100.ra2_in1k
levit_128s
maxvit_base_tf_512.in21k_ft_in1k
maxvit_large_tf_512.in21k_ft_in1k
mobilenetv3_large_100.miil_in21k_ft_in1k
mobilenetv3_small_075.lamb_in1k
mobilenetv3_small_100.lamb_in1k
mobilevit_xs
mobilevit_xxs
mobilevitv2_100
mobilevitv2_150_384_in22ft1k
regnetz_d8
rexnet_100
swin_large_patch4_window12_384
tf_efficientnet_b0.ns_jft_in1k
tf_efficientnet_b3.ns_jft_in1k
tf_efficientnet_b4.ns_jft_in1k
tf_efficientnet_b5.ns_jft_in1k
tf_efficientnet_b6.ns_jft_in1k
tf_efficientnetb7.ns_jft_in1k
tf_efficientnetv2_b0.in1k
tf_mobilenetv3_small_100.in1k
tinynet_e.in1k
vit_base_patch16_clip_224.laion2b_ft_in12k_in1k
vit_base_patch16_clip_384.laion2b_ft_in12k_in1k
vit_base_patch32_clip_224.laion2b_ft_in12k_in1k
vit_base_patch32_clip_384.laion2b_ft_in12k_in1k
volo_d1_384
volo_d2_384
volo_d3_448
volo_d4_448
xcit_nano_12_p8_384_dist
xcit_small_12_p8_384_dist
xcit_small_24_p8_384_dist
xcit_tiny_12_p8_384_dist
xcit_tiny_24_p8_384_dist
### Validation set models

<table>
<thead>
<tr>
<th>Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>adv_inception_v3</td>
</tr>
<tr>
<td>beit_base_patch16_224.in22k_ft_in22k</td>
</tr>
<tr>
<td>beit_base_patch16_384.in22k_ft_in22k_in1k</td>
</tr>
<tr>
<td>beit_large_patch16_224.in22k_ft_in22k_in1k</td>
</tr>
<tr>
<td>beit_large_patch16_384.in22k_ft_in22k_in1k</td>
</tr>
<tr>
<td>beitv2_base_patch16_224.in1k_ft_in22k_in1k</td>
</tr>
<tr>
<td>beitv2_large_patch16_224.in1k_ft_in22k_in1k</td>
</tr>
<tr>
<td>beitn36_384</td>
</tr>
<tr>
<td>beit_s24_224</td>
</tr>
<tr>
<td>beit_s36_384</td>
</tr>
<tr>
<td>beit_xxs24_224</td>
</tr>
<tr>
<td>beit_xxs36_224</td>
</tr>
<tr>
<td>coat_lite_mini</td>
</tr>
<tr>
<td>coat_lite_tiny</td>
</tr>
<tr>
<td>coatnet_1_rw_224_sw_in1k</td>
</tr>
<tr>
<td>coatnet_2_rw_224_sw_in1k2k_ft_in1k</td>
</tr>
<tr>
<td>coatnet_bn_0_0_224_sw_in1k</td>
</tr>
<tr>
<td>coatnet_mlp_1_rw_224_sw_in1k</td>
</tr>
<tr>
<td>coatnet_mlp_2_rw_224_sw_in1k</td>
</tr>
<tr>
<td>coatnet_mlp_3_rw_224_sw_in1k</td>
</tr>
<tr>
<td>coatnet_nano_rw_224_sw_in1k</td>
</tr>
<tr>
<td>convit_small</td>
</tr>
<tr>
<td>convit_1024_20_kx9_p14</td>
</tr>
<tr>
<td>convit_1024_20_kx9_p14</td>
</tr>
<tr>
<td>convnext_atto_ols.a2_in1k</td>
</tr>
<tr>
<td>convnext_base.clip_laion2b_augreg</td>
</tr>
<tr>
<td>convnext_base.clip_laion2b_augreg_ft_in12k_in1k</td>
</tr>
<tr>
<td>convnext_base.clip_laion2b_augreg_ft_in1k</td>
</tr>
<tr>
<td>convnext_base.clip_laiona_320</td>
</tr>
<tr>
<td>convnext_base.clip_laiona_augreg_ft_in1k_384</td>
</tr>
<tr>
<td>convnext_base.fb_in22k</td>
</tr>
<tr>
<td>convnext_base.fb_in22k_ft_in1k</td>
</tr>
<tr>
<td>convnext_femto.ols.d1_in1k</td>
</tr>
<tr>
<td>convnext_femto.ols.d1_in1k</td>
</tr>
<tr>
<td>convnext_large.fb_in22k</td>
</tr>
<tr>
<td>convnext_large.fb_in22k_ft_in1k</td>
</tr>
<tr>
<td>convnext_large.mlp.clip_laion2b_augreg_ft_in12k_384</td>
</tr>
<tr>
<td>convnext_large.mlp.clip_laion2b_augreg_ft_in1k_384</td>
</tr>
<tr>
<td>convnext_large.mlp.clip_laion2b_augreg_ft_soup_320</td>
</tr>
<tr>
<td>convnext_large.mlp.clip_laion2b_soup_ft_in12k_384</td>
</tr>
<tr>
<td>convnext_large.mlp.clip_laion2b_soup_ft_in1k2k_384</td>
</tr>
<tr>
<td>convnext_nano.in12k</td>
</tr>
<tr>
<td>convnext_nano.ols.d1h_in1k</td>
</tr>
<tr>
<td>convnext_pico.ols.d1_in1k</td>
</tr>
<tr>
<td>convnext_small.fb_in22k</td>
</tr>
<tr>
<td>convnext_small.fb_in22k_ft_in1k</td>
</tr>
<tr>
<td>convnext_small.in12k_ft_in1k</td>
</tr>
<tr>
<td>convnext_tiny.fb_in1k</td>
</tr>
<tr>
<td>convnext_tiny.fb_in22k_ft_in1k</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>bat_resnext26ts.ch_in1k</td>
</tr>
<tr>
<td>beit_base_patch16_224.in22k_ft_in22k_in1k</td>
</tr>
<tr>
<td>beit_large_patch16_224.in22k_ft_in22k_in1k</td>
</tr>
<tr>
<td>beit_large_patch16_384.in22k_ft_in22k_in1k</td>
</tr>
<tr>
<td>beitv2_base_patch16_224.in1k_ft_in22k_in1k</td>
</tr>
<tr>
<td>beitv2_large_patch16_224.in1k_ft_in22k_in1k</td>
</tr>
<tr>
<td>botnet26t_256</td>
</tr>
<tr>
<td>coat_m48_448</td>
</tr>
<tr>
<td>coat_s24_384</td>
</tr>
<tr>
<td>coat_xxs24_384</td>
</tr>
<tr>
<td>coat_xxs36_384</td>
</tr>
<tr>
<td>coat_lite_small</td>
</tr>
<tr>
<td>coat_mini</td>
</tr>
<tr>
<td>coatnet_0_rw_224.sw_in1k</td>
</tr>
<tr>
<td>coatnet_2_rw_224.sw_in12k</td>
</tr>
<tr>
<td>coatnet_3_rw_224.sw_in12k</td>
</tr>
<tr>
<td>coatnet_nano_rw_224.sw_in1k</td>
</tr>
<tr>
<td>coatnet_mlp_1_rw_224.sw_in1k</td>
</tr>
<tr>
<td>coatnet_mlp_2_rw_224.sw_in1k</td>
</tr>
<tr>
<td>coatnet_mlp_3_rw_224.sw_in1k</td>
</tr>
<tr>
<td>coatnet_nano_rw_224.sw_in1k</td>
</tr>
<tr>
<td>convit_base</td>
</tr>
<tr>
<td>convit_tiny</td>
</tr>
<tr>
<td>convnext_1536_20</td>
</tr>
<tr>
<td>convnext_attio2.in1k</td>
</tr>
<tr>
<td>convnext_base.clip_laion2b</td>
</tr>
<tr>
<td>convnext_base.clip_laion2b_augreg_ft_in12k</td>
</tr>
<tr>
<td>convnext_base.clip_laion2b_augreg_ft_in1k</td>
</tr>
<tr>
<td>convnext_base.clip_laiona</td>
</tr>
<tr>
<td>convnext_base.clip_laiona_augreg_320</td>
</tr>
<tr>
<td>convnext_base.fb_in1k</td>
</tr>
<tr>
<td>convnext_base.fb_in22k_ft_in1k</td>
</tr>
<tr>
<td>convnext_femto.d1_in1k</td>
</tr>
<tr>
<td>convnext_large.fb_in22k_ft_in1k</td>
</tr>
<tr>
<td>convnext_large.mlp.clip_laion2b_augreg_ft_in12k_384</td>
</tr>
<tr>
<td>convnext_large.mlp.clip_laion2b_augreg_ft_in1k_384</td>
</tr>
<tr>
<td>convnext_large.mlp.clip_laion2b_augreg_ft_soup_320</td>
</tr>
<tr>
<td>convnext_large.mlp.clip_laion2b_soup_ft_in12k_384</td>
</tr>
<tr>
<td>convnext_large.mlp.clip_laion2b_soup_ft_in1k2k_384</td>
</tr>
<tr>
<td>convnext_nano.in12k</td>
</tr>
<tr>
<td>convnext_nano.ols.d1h_in1k</td>
</tr>
<tr>
<td>convnext_pico.ols.d1_in1k</td>
</tr>
<tr>
<td>convnext_small.fb_in22k</td>
</tr>
<tr>
<td>convnext_small.fb_in22k_ft_in1k</td>
</tr>
<tr>
<td>convnext_small.in12k_ft_in1k</td>
</tr>
<tr>
<td>convnext_tiny.fb_in1k</td>
</tr>
<tr>
<td>convnext_tiny.fb_in22k_ft_in1k</td>
</tr>
</tbody>
</table>
Unsupervised Embedding Quality Evaluation

convnext_tiny.in12k
convnext_tiny.in12k_ft_in1k_384
convnext_xlarge.fb_in22k
convnext_xlarge.fb_in22k_ft_in1k_384
convnext_xlarge.clip_laion2b
convnextv2_atto.fcmae
convnextv2_base.fcmae
convnextv2_base.fcmae_ft_in22k_in1k
convnextv2_feento.fcmae
convnextv2_huge.fcmae
convnextv2_huge.fcmae_ft_in22k_in1k_384
convnextv2_large.fcmae
convnextv2_large.fcmae_ft_in22k_in1k
convnextv2_nano.fcmae
convnextv2_nano.fcmae_ft_in22k_in1k
convnextv2_pico.fcmae
convnextv2_tiny.fcmae
convnextv2_tiny.fcmae_ft_in22k_in1k

crossvit_15_240
crossvit_15_dagger_408
crossvit_18_240
crossvit_9_240
crossvit_base_240
crossvit_tiny_240
cs3darknet_focus_m
cs3darknet_m
cs3edgenet_x
cs3eddarknet_1
cspdarknet53
cspresnext50
darknetaa53
davit_small.msft_in1k
davit_base.msft_in1k
davit_tiny.msft_in1k
deit3_base_patch16_224.fb_in1k
deit3_base_patch16_384.fb_in1k
deit3_huge_patch14_224.fb_in1k
deit3_large_patch16_224.fb_in1k
deit3_large_patch16_384.fb_in1k
deit3_medium_patch16_224.fb_in1k
deit3_medium_patch16_384.fb_in1k
deit3_small_patch16_224.fb_in1k
deit3_small_patch16_384.fb_in1k
deit_base_distilled_patch16_224.fb_in1k
deit_base_patch16_224.fb_in1k
deit_small_distilled_patch16_224.fb_in1k
deit_tiny_distilled_patch16_224.fb_in1k
densenet121
densenet169
densenetblur121d
dla102k
dla169
dla46_c
dla60
dla60_res2next
dla60x_c
dm_nfnet_f1.dm_in1k
dm_nfnet_f0.dm_in1k
Unsupervised Embedding Quality Evaluation

dm_nfnet_f3.dm_in1k
dm_nfnet_f5.dm_in1k
dpn107
dpn68
dpn92
eca_botnext26ts.a256
eca_nfnet.10.ra2.in1k
eca_nfnet.12.ra3.in1k
eca_resnext26ts.rb.in1k
ecaresnet101d.pruned.miil_in1k
ecaresnet26t.ra2.in1k
ecaresnet50d.pruned.miil_in1k
ecaresnet50t.ra2.in1k
ecaresnet50t.ra2_in1k
edgenext_base
edgenext_small
edgenext_x_s

efficientformer_13.snap_dist.in1k
efficientformerv2.1.snap_dist.in1k
efficientformerv2.5.snap_dist.in1k
efficientnet_b0.ra_in1k
efficientnet_b1.pruned.in1k
efficientnet_b2.pruned.in1k
efficientnet_b3.pruned.in1k
efficientnet_b5.in1k
efficientnet_e1.ra_in1k
efficientnet_e4.ra_in1k
efficientnetv2_rw._m.ago_in1k
efficientnetv2_rw.t.ra2_in1k
ese_vovnet19b_dv
eva02_base_patch14.224.mim_in22k
eva02_base_patch14.448.mim_in22k_ft_in22k
eva02_large_patch14.224.mim_in22k
eva02_large_patch14.448.mim_in22k_ft_in22k
eva02_large_patch14.448.mim_m38m_ft_in22k
neva02_small_patch14.224.mim_in22k
eva02_tiny_patch14.224.mim_in22k
eva_giant_patch14.224.clip_ft_in1k
neva_giant_patch14.336.m30m_ft_in22k_in1k
eva_large_patch14.196.in22k_ft_in1k
eva_large_patch14.336.in22k_ft_in1k
rfnetc.100.rmp_in1k
rfnetv3.d.ra2_in1k
flexivit_base.100ep_in21k
flexivit_base.300ep_in1k
flexivit_base.600ep_in1k
flexivit_base.patch30_in21k
flexivit_large.300ep_in1k
flexivit_small.1200ep_in1k
flexivit_small.600ep_in1k
focalnet_base.srf.mi_in1k
focalnet_huge_fl4.mi_in22k
Unsupervised Embedding Quality Evaluation

regnety_080.ra3_in1k
regnety_120.ra3_in1k
regnety_120.sw_in12k_ft_in1k
regnety_1280.seer_ft_in1k
regnety_160.11m_in12k_ft_in1k
regnety_160.sw_in12k
regnety_160.swag_ft_in1k
regnety_160.tv2_in1k
regnety_320.pycls_in1k
regnety_320.seer_ft_in1k
regnety_640.seer
regnety_c16_evos.ch_in1k
regnety_d8
regnety_dr_evos.ch_in1k
repvgg_a2.rvgg_in1k
repvgg_b1.rvgg_in1k
repvgg_b2.rvgg_in1k
repvgg_b3.rvgg_in1k
res2net101_26w_4s
res2net50_26w_4s
res2next50
resmlp_12_224.fb_distilled_in1k
resmlp_24_224.fb_distilled_in1k
resmlp_24_224.fb_in1k
resmlp_36_224.fb_distilled_in1k
resmlp_36_224.fb_in1k
resmlp_48_224.fb_distilled_in1k
resnest101e
resnest200e
resnest26d
resnest50d_1s4x24d
resnet101.a1_in1k
resnet101.a2_in1k
resnet101.gluon_in1k
resnet101.tv2_in1k
resnet101d.gluon_in1k
resnet101a.gluon_in1k
resnet14t.o3_in1k
resnet152.alh_in1k
resnet152.a2_in1k
resnet152.tv2_in1k
resnet152c.gluon_in1k
resnet152d.ra2_in1k
resnet18.a1_in1k
resnet18.a3_in1k
resnet18.fb_ssl_yfcc100m_ft_in1k
resnet18.tv_in1k
resnet200d.ra2_in1k
resnet26d.bt_in1k
resnet32ts.ra2_in1k
Unsupervised Embedding Quality Evaluation

selecsls60
selecsls60b
sennasnet_075.rmsp_in1k
densenet150.gluon_in1k
sequencer2d_m
seresnet152d.ra2_in1k
seresnet50.a1_in1k
seresnet50.a3_in1k
seresnext101_32x4d.gluon_in1k
seresnext101d_32x8d.ah_in1k
seresnext26d_32x4d.bt_in1k
seresnext2ts.ch_in1k
seresnext50_32x4d.racm_in1k
seresnexta101d_32x8d.sw_in12k
seresnexta101d_32x8d.sw_in12k_ft_in1k
skresnet18
skresnet50_32x4d
spnasnet_100.rmsp_in1k
swin_base_patch4_window12_384.ms_in1k
swin_base_patch4_window7_224.ms_in1k
swin_base_patch4_window7_224.me_in22k
swin_large_patch4_window12_384.ms_in22k
swin_large_patch4_window7_224.ms_in22k
swin_s3_base_224.ms_in1k
swin_s3_tiny_224.ms_in1k
swin_small_patch4_window7_224.ms_in22k
swin_tiny_patch4_window7_224.ms_in22k
swin_tiny_patch4_window7_224.ms_in22k_ft_in1k
swinv2_base_window12to16_192to256.ms_in22k
swinv2_base_window16_256.ms_in1k
swinv2_cr_small_224.sw_in1k
swinv2_cr_tiny_224.sw_in1k
swinv2_large_window12to16_192to256.ms_in22k
swinv2_tiny_window16_256.ms_in1k
tf_efficientnet_b0.a_enable
tf_efficientnet_b0.ca_in1k
tf_efficientnet_b0.na_jft_in1k
tf_efficientnet_b1.ap_in1k
tf_efficientnet_b2.aa_in1k
tf_efficientnet_b2.na_jft_in1k
tf_efficientnet_b3.ap_in1k
tf_efficientnet_b4.a Enable
tf_efficientnet_b4.na_jft_in1k
tf_efficientnet_b5.na_jft_in1k
tf_efficientnet_b6.a Enable
tf_efficientnet_b6.na_jft_in1k
tf_efficientnet_b7.na_jft_in1k
tf_efficientnet_b8.ap_in1k
tf_efficientnet_cc_b0_4e.in1k
tf_efficientnet_cc_b0_8e.in1k
tf_efficientnet_cc_b1_8e.in1k
tf_efficientnet_el.in1k
tf_efficientnet_el.in1k
tf_efficientnet_lite0.in1k
tf_efficientnet_lite2.in1k
tf_efficientnet_lite4.in1k
tf_efficientnetv2_b1.in1k
tf_efficientnetv2_b3.in1k

17
Unsupervised Embedding Quality Evaluation

tf_efficientnetv2_b3.in21k_ft_in1k
tf_efficientnetv2_l.in1k

tf_efficientnetv2_m.in21k

tf_efficientnetv2_s.in21k

tf_inception_v3
tf_mixnet_n.in1k

tf_mobilevit_large_075.in1k

tf_mobilevit_large_minimal_100.in1k

tf_mobilevit_small_100.in1k

tinynet_a.in1k
tinynet_c.in1k
tinynet_e.in1k

tv_densenet121

twins_pcpvt_large
twins_svt_base

twins_svt_small
twinclip

vgg11_bn

twins_xl

twins_xl_s

vit_base_patch16_224.augreg2_in21k_ft_in1k

tvit_base_patch16_224.augreg2_in21k

tvit_base_patch16_224.dino

vit_base_patch16_224.sam

tvit_base_patch16_224.miil.in21k_ft_in1k

vit_base_patch16_384.augreg_in21k_ft_in1k

vit_base_patch16_clip_224.laion2b

tvit_base_patch16_clip_224.laion2b_ft_in1k

tvit_base_patch16_clip_224.openai

vit_base_patch16_clip_384.laion2b_ft_in1k

vit_base_patch16_rpn_224.in1k

vit_base_patch32_224.augreg_in21k

vit_base_patch32_224.sam

vit_base_patch32_384.augreg_in21k_ft_in1k

vit_base_patch32_clip_224.laion2b

vit_base_patch32_clip_224.laion2b_ft_in1k

vit_base_patch32_clip_448.laion2b_ft_in1k

vit_base_patch8_224.augreg_in21k

vit_base_patch8_224.dino

vit_giant_patch14_clip_224.laion2b

vit_huge_patch14_clip_224.laion2b

vit_large_patch14_clip_224.laion2b_ft_in1k

vit_large_patch14_clip_336.laion2b_ft_in1k

vit_large_patch14_clip_224.openai

vit_large_patch14_clip_336.openai
Unsupervised Embedding Quality Evaluation

vit_large_patch16_224.augreg_in21k
vit_large_patch16_384.augreg_in21k_ft_in1k
vit_large_patch32_384.orig_in21k_ft_in1k
vit_medium_patch16_gap_240.in12k
vit_medium_patch16_gap_384.in12k_ft_in1k
vit_repos_base_patch16.clsseg_224.swt_in1k
vit_repos_medium_patch16_224.swt_in1k
vit_repos_medium_patch16.rpn_224.swt_in1k
vit_small_patch16_224.augreg_in1k
vit_small_patch16_224.augreg_in21k
vit_small_patch16_384.augreg_in21k_ft_in1k
vit_small_patch32_384.augreg_in21k_ft_in1k
vit_repos_small_patch16_224.swt_in1k
vit_repos_medium_patch16.rpn_224.swt_in1k
vit_repos_medium_patch16_224.swt_in1k
vit_repos_medium_patch16_224.swt_in1k
vit_tiny_patch16_224.augreg_in21k_ft_in1k
vit_tiny_patch16_224.augreg_in21k
vit_tiny_patch16_384.augreg_in21k_ft_in1k
vit_tiny_patch16_384.augreg_in21k
vit_tiny_patch16_p8_in12k
volod1_384
volod2_384
volod3_448
volod4_448
volod5_448
volod6_448
wide_resnet101_2.tv2_in1k
wide_resnet50_2.sacm_in1k
wide_resnet50_2.tv_in1k
xception
xception41
xception65
xception71
xcit_large_24_p16_224_dist
xcit_large_24_p8_224
xcit_large_24_p8_384_dist
xcit_medium_24_p16_224_dist
xcit_medium_24_p8_224
xcit_medium_24_p8_384_dist
xcit_nano_12_p16_224_dist
xcit_nano_12_p8_224
xcit_nano_12_p8_384_dist
xcit_small_12_p16_224_dist
xcit_small_12_p8_224
xcit_small_12_p8_384_dist
xcit_small_24_p16_224_dist
xcit_small_24_p8_224
xcit_small_24_p8_384_dist
xcit_tiny_12_p16_224_dist
xcit_tiny_12_p8_224
xcit_tiny_12_p8_384_dist
xcit_tiny_24_p16_224_dist
xcit_tiny_24_p8_224
xcit_tiny_24_p8_384_dist

19