ContraSim – Analyzing Neural Representations Based on Contrastive Learning

Adir Rahamim
adir.rahamim@cs.technion.ac.il

Yonatan Belinkov
belinkov@technion.ac.il

Technion – Israel Institute of Technology

Abstract

Recent work has compared neural network representations via similarity-based analyses to improve model interpretation. The quality of a similarity measure is typically evaluated by its success in assigning a high score to representations that are expected to be matched. However, existing similarity measures perform mediocrely on standard benchmarks. In this work, we develop a new similarity measure, dubbed ContraSim, based on contrastive learning. In contrast to common closed-form similarity measures, ContraSim learns a parameterized measure by using both similar and dissimilar examples. We perform an extensive experimental evaluation of our method, with both language and vision models, on the standard layer prediction benchmark and two new benchmarks that we introduce: the multilingual benchmark and the image–caption benchmark. In all cases, ContraSim achieves much higher accuracy than previous similarity measures, even when presented with challenging examples. Finally, ContraSim is more suitable for the analysis of neural networks, revealing new insights not captured by previous measures.[1]

1 Introduction

Representation learning has been key in the success of deep neural networks (NNs) on many tasks. However, the resulting representations are opaque and not easily understood. A recent line of work analyzes internal representations by comparing two sets of representations, for instance from two different models. The choice of similarity measure is crucial and much work has been devoted to developing various such measures (Raghu et al., 2017; Morcos et al., 2018; Kornblith et al., 2019; Wu et al., 2020). Similarity-based analyses may shed light on how different datasets, architectures, etc., change the model's learned representations, and improve their interpretability. For example, a similarity analysis showed that lower layers in different language models are more similar to each other, while fine-tuning affects mostly the top layers (Wu et al., 2020).

Various similarity measures have been proposed for comparing representations, among them the most popular ones are based on centered kernel alignment (CKA) (Kornblith et al., 2019) and canonical correlation analysis (CCA) (Hotelling, 1936; Morcos et al., 2018). They all share a similar methodology: given a pair of feature representations *of the same input*, they estimate the similarity between them, without considering other examples. However, they all perform mediocrely on standard benchmarks. Thus, we might question the reliability of their similarity scores, as well as the validity of interpretability insights derived from them.

Motivated by this, we introduce ContraSim, a new similarity measure for interpreting NNs, based on contrastive learning (CL) (Chen et al., 2020; He et al., 2020). Contrary to prior work (e.g., Raghu et al., 2017; Kornblith et al., 2019), which defines closed-form general-purpose similarity measures, ContraSim is a learnable similarity measure that uses examples with a high similarity (the *positive* set) and examples that have a low similarity (the *negative* set), to train an encoder that maps representations to the space where similarity is measured. In the projected space, representation similarity is maximized with positive examples and minimized with negative examples. Our approach allows specializing the similarity measure to a particular domain, to obtain a more reliable and specific analysis. The similarity between projected representations is determined using a simpler closed-form measure.

We experimentally evaluate ContraSim on standard benchmark for similarity measures – the layer prediction benchmark (Kornblith et al., 2019), and two new benchmarks we introduce in this paper:

[1] Code is available at: https://github.com/technion-cs-nlp/ContraSim.

the multilingual benchmark and the image–caption benchmark. In experiments with both language and vision models and multiple datasets, ContraSim outperforms common similarity measures. In addition, we investigate a more challenging scenario, where during evaluation instead of choosing a random sentence, we retrieve a highly similar sentences as confusing examples, using the Facebook AI Similarity Search (FAISS) library (Johnson et al., 2019). While other similarity measures are highly affected by this change, our method maintains a high accuracy with a very small degradation. We attribute this to the highly separable representations that our method learns. Even when ContraSim is trained on data from one domain/task and evaluated on data from another domain/task, it achieves superior performance.

Through ablations, we demonstrate that the CL procedure is crucial to the success of the method and only maximizing the similarity of positive examples is not sufficient. Furthermore, we demonstrate that ContraSim reveals new insights not captured by previous similarity measures. For instance, CKA suggests that the BERT language model (Devlin et al., 2019) is more similar to vision models than GPT-2 (Radford et al., 2019). Our analysis indicates that both BERT and GPT-2 create representations that are equally similar to the vision ones.

2 Related Work

Many studies have used similarity measures for the interpretability of NNs. For instance, Kornblith et al. (2019) showed that adding too many layers to a convolutional neural network, trained for image classification, hurts its performance. Using CKA, they found that more than half of the network's layers are very similar to the last. They further found that two models trained on different image datasets (CIFAR-10 and CIFAR-100, Krizhevsky et al. 2009) learn representations that are similar in the shallow layers. Similar findings were noted for language models by Wu et al. (2020). The latter also evaluated the effect of fine-tuning on language models and found that the top layers are most affected by fine-tuning. Kornblith et al. (2019) and Morcos et al. (2018) found that increasing the model's layer width results in more similar representations between models. Raghu et al. (2017) provided an interpretation of the learning process by examining how similar representations were during the training process compared to final representations. They found that networks converge from bottom to top. In all this line of work, the similarity is computed using only similar examples, using functional closed-form measures. In contrast, we use both positive and negative samples in a learnable similarity measure, which allows adaptation to specific tasks.

Separate work employs contrastive learning for representation learning (He et al., 2020; Chen et al., 2020) and data retrieval (Karpathy et al., 2014; Huang et al., 2008). In contrast to that line of work, we borrow the contrastive learning formulation for similarity-based interpretations of deep networks.

3 Problem Setup

Let $\mathbb{X} = \{(x_1^{(i)}, x_2^{(i)})\}_{i=1}^N$ denote a set of N examples, and $\mathbb{A} = \{(\boldsymbol{a}_1^{(i)}, \boldsymbol{a}_2^{(i)})\}_{i=1}^N$ the set of representations generated for the examples in $\mathbb{X}$. A representation is a high-dimensional vector of neuron activations. Representations may be created by different models, different layers of the same model, etc. For instance, $x_1^{(i)}$ and $x_2^{(i)}$ may be the same input, with $\boldsymbol{a}_1^{(i)}$ and $\boldsymbol{a}_2^{(i)}$ representations of that input in different layers.

Our goal is to obtain a scalar similarity score, which represents the similarity between the two sets of representations, $\boldsymbol{a}_1^{(i)}$ and $\boldsymbol{a}_2^{(i)}$, and ranges from 0 (no similarity) to 1 (identical representations). That is, we define $\boldsymbol{X}_1 \in \mathbb{R}^{N \times p_1}$ as a matrix of p_1-dim activations of N data points, and $\boldsymbol{X}_2 \in \mathbb{R}^{N \times p_2}$ as another matrix of p_2-dim activations of N data points. We seek a similarity measure, $s(\boldsymbol{X}_1, \boldsymbol{X}_2)$.

4 ContraSim

In this section we introduce ContraSim, a similarity index for measuring the similarity of neural network representations. Our method uses a trainable encoder, which first maps representations to a new space and then measures the similarity of the projected representations. Formally, let e_θ denote an encoder with trainable parameters θ, and assume two representations $\boldsymbol{a}_1$ and $\boldsymbol{a}_2$. In order to obtain a similarity score between 0 and 1, we first apply L2 normalization to the encoder outputs: $\boldsymbol{z}_1 = e_\theta(\boldsymbol{a}_1)/\|e_\theta(\boldsymbol{a}_1)\|$ (and similarly for $\boldsymbol{a}_2$). Then their similarity is calculated as:

$$s(\boldsymbol{z}_1, \boldsymbol{z}_2) \tag{1}$$

where s is a simple closed-form similarity measure for two vectors. Throughout this work we use dot product for s.

For efficiency reasons, we calculate the similarity between batches of the normalized encoder representations, dividing by the batch size n:

$$\frac{1}{n}\sum_{i=1}^{n}\left(\boldsymbol{z}_1^i \cdot \boldsymbol{z}_2^i\right) \tag{2}$$

If the representations a_1 and a_2 have the same dimensionality, ContraSim can be trained with a single encoder shared for both representations. In the case that a_1 and a_2 have different dimensions, two different encoders are trained, $e_{\theta 1}$ and $e_{\theta 2}$, one for each representation set. By that, ContraSim can calculate the similarity of representations with different dimensionality, as each encoder has a different input dimension, but both encoders share the same output dimension. Experiments with representations with different dimensionality are in Section 6.3.3.

Training. None of the similarity measures commonly used in NNs analysis uses negative examples to estimate the similarity of a given pair (Section 2). Given two examples, these measures output a scalar that represents the similarity between them, without leveraging data from other examples. However, based on knowledge from other examples, we can construct a better similarity measure. In particular, for a given example $x^{(i)} \in \mathbb{X}$ with its encoded representation $\boldsymbol{z}_i$, we construct a set of *positive* example indices, $P(i) = \{p_1, ..., p_q\}$, and a set of *negative* example indices, $N(i) = \{n_1, ..., n_t\}$. The choice of these sets is task-specific and allows one to add inductive bias to the training process.

We train the encoder to maximize the similarity of $\boldsymbol{z}_i$ with all the positive examples, while at the same time making it dis-similar from the negative examples. We leverage ideas from contrastive learning (Chen et al., 2020; He et al., 2020), and minimize the following objective:

$$\mathcal{L} = \sum_{i \in I} \frac{-1}{|P(i)|} \log \frac{\sum_{p \in P(i)} \exp(\boldsymbol{z}_i \cdot \boldsymbol{z}_p / \tau)}{\sum_{n \in N(i)} \exp(\boldsymbol{z}_i \cdot \boldsymbol{z}_n / \tau)} \tag{3}$$

with scalar temperature parameter $\tau > 0$. Here $\boldsymbol{z}_p$ and $\boldsymbol{z}_n$ are normalized encoder outputs of representations from the positive and negative groups, respectively. It is important to notice that we do not alter the original model representations, and the only trainable parameters are the encoder e parameters, θ. Our work uses negative examples and a trainable encoder for constructing a similarity measure. We evaluate these two aspects in the experimental section and show that using negative examples is an important aspect of our method. Combining the two leads to a similarity measure that outperforms current measures.

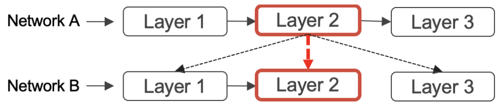

Figure 1: Layer prediction benchmark. Given two models differing only in weight initialization, A and B, for each layer in the first model, among all layers of the second model, the highest similarity should be assigned to the architecturally-corresponding layer.

5 Similarity Measure Evaluation

For evaluation, we use the known layer prediction benchmark and two new benchmarks we design: the multilingual benchmark and the image–caption benchmark. We further propose a strengthened version of the last two using the FAISS software.

5.1 Layer prediction benchmark

Proposed by Kornblith et al. (2019), a basic and intuitive benchmark is to assess the invariance of a similarity measure against changes to a random seed. Given two models differing only in their weight initializations, for each layer in the first model, among all layers of the second model, one can expect that a good similarity measure assigns the highest similarity for the architecturally-corresponding layer. Formally, let f and g be two models with k layers, and define f_i and g_i as the i^{th} layer of f and g models, respectively. After calculating the similarity of f_i to each layer of g $(g_1, \ldots, g_k)$, the pair with the highest similarity is expected to be (f_i, g_i). This benchmark counts the number of layers for which this pair was indeed the most similar, and divides by the total number of pairs. An illustration is found in Figure 1.

The intuition behind this benchmark is that each layer captures different information about the input data. For example, Jawahar et al. (2019) showed that different layers of the BERT model capture different semantic information.

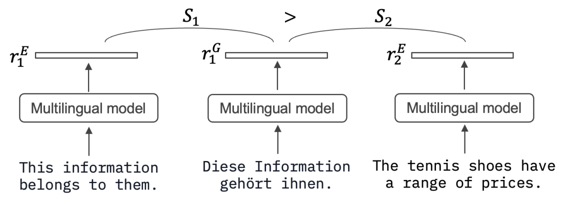

Figure 2: The multilingual benchmark. r_1^E and r_1^G denote the representations of the same sentence in different languages, and S_1 is their similarity. r_2^E represents the random sentence representation, and S_2 is the similarity between it and r_1^G. We expect S_1 to be higher than S_2.

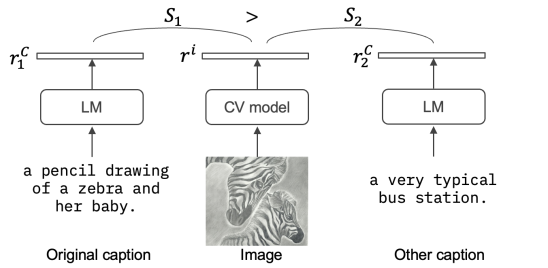

Figure 3: The image–caption benchmark. r_1^C and r^i denote the representations of the caption and the image pair, respectively, and S_1 is their similarity. r_2^C denotes the random caption representation, and S_2 is the similarity between it and r^i. S_1 should be greater than S_2.

5.2 Multilingual benchmark

Multilingual models, such as Multilingual-BERT (Devlin et al., 2019), learn to represent texts in different languages in the same representation space. Interestingly, these models show cross-lingual zero-shot transferability, where a model is fine-tuned in one language and evaluated in a different language (Pires et al., 2019). Muller et al. (2021) analyzed this transferability and found that lower layers of the Multilingual-BERT align the representations between sentences in different languages.

Since multilingual models share similarities between representations of different languages, we expect that a good similarity measure should assign a high similarity to two representations of a sentence in two different languages. In other words, we expect similarity measures to be invariant to the sentence source language. Consider a multilingual model f and dataset $\mathbb{X}$, where each entry consists of the same sentence in different languages. Let $(x_1^{(i)}, x_2^{(i)}) \in \mathbb{X}$ be a sentence written in two languages – language A and language B. The similarity between $f(x_1^{(i)})$ and $f(x_2^{(i)})$ should be higher than the similarity between $f(x_1^{(i)})$ and the representation of a sentence in language B randomly chosen from $\mathbb{X}$, i.e., $f(x_2^{(j)})$, where $(x_1^{(j)}, x_2^{(j)}) \in \mathbb{X}$ is a randomly chosen example from $\mathbb{X}$. The benchmark calculates the fraction of cases for which the correct translation was assigned the highest similarity. An illustration is found in Figure 2.

Additionally, we suggest a strengthened version of the multilingual benchmark, using FAISS. Instead of sampling random sentences in language B, we use FAISS to find the pair $(x_1^{(j)}, x_2^{(j)}) \in \mathbb{X}$, where $x_2^{(j)} \neq x_2^{(i)}$, with the representation $f(x_2^{(j)})$ that is most similar to $f(x_2^{(i)})$, out of a large set of vectors pre-indexed by FAISS. This leads to a more challenging scenario, as the similarity between $x_1^{(i)}$ and FAISS-sampled $x_2^{(j)}$ is expected to be higher than the similarity between $x_1^{(i)}$ and randomly chosen $x_2^{(j)}$, increasing the difficulty of identifying the pair $(x_1^{(i)}, x_2^{(i)})$ as the highest-similarity pair. Note that FAISS only affects the evaluation step, and is not used during ContraSim's training.

5.3 Image-caption benchmark

Let $\mathbb{X}$ be a dataset of images and their textual descriptions (captions), f be a computer vision model and g a language model. Given a pair of an image and its caption, $(m^{(i)}, c^{(i)}) \in \mathbb{X}$, a good similarity measure is expected to assign a high similarity to their representations – $f(m^{(i)}), g(c^{(i)})$. In particular, this similarity should be higher than that of the pair of the same image representation $f(m^{(i)})$ and some random caption's representation $g(c^{(j)})$, where $c^{(j)}$ is a randomly chosen caption from $\mathbb{X}$.

The intuition behind this benchmark is that an image and its caption represent the same scene in a different way. Thus, their representations should have a higher similarity than that of the same image and some random caption. An illustration is found in Figure 3. As in the multilingual benchmark, we also propose a strengthened variant for the image–caption benchmark using FAISS. Rather than sampling a random caption $c^{(j)}$, we use FAISS to find the pair $(m^{(j)}, c^{(j)}) \in \mathbb{X}$, where $c^{(j)} \neq c^{(i)}$, with the representation $g(c^{(j)})$ that is most similar to $g(c^{(i)})$.

6 Experiments

Baselines. We compare ContraSim with the following standard baselines.

- **Centered Kernel Alignment (CKA)**: Proposed by Kornblith et al. (2019), CKA computes a kernel matrix for each matrix representation input, and defines the scalar similarity index as the two kernel matrices' alignment. We use a linear kernel for CKA evaluation, as the original paper reveals similar results for both linear and RBF kernels. CKA is our main point of comparison due to its success in prior work and wide applicability.

- **PWCCA**: Proposed by Morcos et al. (2018), PWCCA is an extension of Canonical Correlation Analysis (CCA). Given two matrices, CCA finds bases for those matrices, such that after projecting them to those bases the correlation between the projected matrices is maximized. While in CCA the scalar similarity index is computed as the mean correlation coefficient, in PWCCA that mean is weighted by the importance each canonical correlation has on the representation.[2]

See Appendix A.4 for more details on the current methods. In the main body we report results of the more successful methods: CKA and PWCCA. Additional baseline are reported in Appendix A.5.

Ablations. In addition, we report the results of two new similarity measures, which use an encoder to map representations to the space where similarity is measured. However, in both methods we train e_θ to only maximize the similarity between positive pairs:

$$\mathcal{L}_{max} = -s(\boldsymbol{z}_1, \boldsymbol{z}_2) \tag{4}$$

where $\boldsymbol{z}_1$ and $\boldsymbol{z}_2$ are representations whose similarity we wish to maximize. We experiment with two functions for s—dot-product and CKA—and accordingly name these similarity measures DeepDot and DeepCKA. These methods provide a point of comparison where the similarity measure is trained, but *without negative examples*, in order to assess the importance of contrastive learning.

Encoders details. In all experiments, the encoder e_θ is a two-layer multi-layered perceptron with hidden layer dimensions of 512 and 256, and output dimension of 128. We trained the encoder for 50 epochs for the layer prediction and 30 epochs for the multilingual and image–caption benchmarks. We used the Adam optimizer (Kingma and Ba, 2015) with a learning rate of 0.001 and a batch size of 1024 representations. We used $\tau = 0.07$ for ContraSim training.

[2]PWCCA and SVCCA require the number of examples to be larger than the feature vector dimension, which is not possible to achieve in all benchmarks. Therefore, we compare with them in a subset of our experiments.

6.1 Layer prediction benchmark

6.1.1 Setup

Recall that this benchmark evaluates whether a certain layer from one model is deemed most similar to its architecturally-corresponding layer from another model, where the two models differ only in their weight initialization. We repeat this process for all layers and 5 different model pairs, and report average accuracy. We experiment with both language and vision setups.

Models. For language experiments, we use the MultiBERTs (Sellam et al., 2021), a set of 25 BERT models, differing only in their weights initialization. For vision experiments, we pre-train 10 visual transformer (ViT) models (Dosovitskiy et al., 2020) on the ImageNet-1k dataset (Russakovsky et al., 2015). Then we fine-tune them on CIFAR-10 and CIFAR-100 datasets (Krizhevsky et al., 2009). Further details are available in Appendix A.3.

Datasets. In language experiments, we use word-level contextual representations generated on two English text datasets: the Penn Treebank (Marcus et al., 1993) and WikiText (Merity et al., 2016). For Penn TreeBank we generate 5005/10019 test/training representations, respectively; for WikiText we generate 5024/10023 test/training representations. Vision experiments are conducted using representations generated on CIFAR-10 and CIFAR-100. For both we generate 5000 and 10000 test and training representations, respectively.

Positive and Negative sets. Given a batch of representations of some model i at layer j, we define its positive set as the representations at the same layer j of all models that differ from i. The negative set is all representations from layers that differ from j (including from model i).

6.1.2 Results

The results are shown in Table 1. In both language and vision evaluations, CKA achieves better results than PWCCA, consistent with the findings by Ding et al. (2021). DeepDot and DeepCKA perform poorly, with much lower results than PWCCA and CKA, revealing that maximizing the similarity is not satisfactory for similarity measure purposes. Our method, ContraSim, achieves excellent results.

Language			Vision		
	Penn TreeBank	WikiText		CIFAR-10	CIFAR-100
PWCCA	38.33	55.00	PWCCA	47.27	45.45
CKA	71.66	76.66	CKA	78.18	74.54
DeepDot	15.55 ± 1.69	14.00 ± 2.26	DeepDot	14.90 ± 1.78	14.18 ± 2.67
DeepCKA	16.66 ± 3.16	19.66 ± 1.63	DeepCKA	17.09 ± 2.95	13.09 ± 4.20
ContraSim			ContraSim		
Penn	100 ± 0	85.45 ± 1.62	CIFAR-10	100 ± 0	90.54 ± 2.90
Wiki	94.00 ± 4.66	100 ± 0	CIFAR-100	85.81 ± 5.68	100 ± 0

Table 1: Layer prediction benchmark accuracy results for language and vision cases. For encoder-based methods we report mean and std over 5 random initializations. For ContraSim, we experiment with training with different datasets (rows) and evaluating on same or different datasets (columns).

When trained on one dataset's training set and evaluated on the same dataset's test set, ContraSim achieves perfect accuracy under this benchmark, with a large margin over CKA results. This holds for both language and vision cases. Even when trained on one dataset and evaluated over another dataset, ContraSim surpasses other similarity measures, showing the transferability of the learned encoder projection between datasets. This is true both when transferring across domains (in text, between news texts from the Penn Treebank and Wikipedia texts), and when transferring across classification tasks (in images, between the 10-label CIFAR-10 and the 100-label CIFAR-100).

6.2 Multilingual benchmark

6.2.1 Setup

This benchmark assesses whether a similarity measure assigns a high similarity to multilingual representations of the same sentence in different languages. Given a batch of (representations of) sentences $b^{(i)}$ in language L_i and their translations $b^{(j)}$ in language L_j, we compute the similarity between $b^{(i)}$ and $b^{(j)}$, and the similarities between $b^{(i)}$ and 10 randomly chosen batches of representations in language L_j. If $b^{(i)}$ is more similar to $b^{(j)}$ than to all other batches, we mark success. (Alternatively, in a more challenging scenario, we use FAISS to find for each representation in each layer the 10 most similar representations in that layer.) We repeat this process separately for representations from different layers of a multilingual model, over many sentences and multiple language pairs, and report average accuracy per layer.[3] Appendix A.1 gives more details.

Model and Data. We use two multilingual models: multilingual BERT (Devlin et al., 2019)[4] and XLM-R (Conneau et al., 2020a). We use the XNLI dataset (Conneau et al., 2018), which has natural language inference examples, parallel in multiple languages. Each example in our dataset is a sentence taken from either the premise or hypothesis sets. We experiment with 5 typologically-different languages: English, Arabic, Chinese, Russian, and Turkish. We created sentence-level representations, with 5000 test 10000 training representations. As a sentence representation, we experiment with [CLS] token representations and with mean pooling of token representations, since Del and Fishel (2021) noted a difference in similarity in these two cases. We report results with [CLS] representations in the main body and with mean pooling in Appendix A.1; the trends are similar.

Positive and Negative sets. Given a pair of languages and a batch of representations at some layer, for each representation we define its positive pair as the representation of the sentence in the different language, and its negative set as all other representations in the batch.

6.2.2 Results

Results with multilingual BERT representations in Table 2 show our method's effectiveness. (Trends with XLM-R are consistent; Appendix A.1.2). Under random sampling evaluation (left block), ContraSim shows superior results over other similarity measures, despite being evaluated on language

[3] For deep similarity measures (DeepCKA, DeepDot, and ContraSim), upon training the encoder on examples from a pair of languages, $(L_r, L_q), r \neq q$, we evaluate it over all other distinct pairs of languages.

[4] https://huggingface.co/bert-base-multilingual-cased

	Random				FAISS			
Layer	CKA	DeepCKA	DeepDot	ContraSim	CKA	DeepCKA	DeepDot	ContraSim
1	71.7 ± 5.3	82.0 ± 6.4	63.3 ± 10.4	95.5 ± 5.4	20.1 ± 4.0	10.7 ± 2.6	29.9 ± 8.7	36.0 ± 10.7
2	78.7 ± 4.4	86.4 ± 4.1	68.5 ± 9.9	95.0 ± 7.2	27.2 ± 5.5	12.3 ± 2.9	46.9 ± 9.8	33.0 ± 14.8
3	86.8 ± 3.0	87.1 ± 3.2	70.4 ± 9.7	96.4 ± 6.7	41.9 ± 8.7	17.6 ± 4.2	51.5 ± 10.3	45.4 ± 20.5
4	92.6 ± 1.4	91.5 ± 2.4	95.4 ± 3.4	99.9 ± 0.2	33.4 ± 7.0	15.2 ± 3.7	52.2 ± 8.6	72.4 ± 9.8
5	88.3 ± 3.2	83.5 ± 5.2	94.7 ± 4.8	99.9 ± 0	49.3 ± 4.3	36.9 ± 6.3	42.4 ± 12.9	99.1. ± 0.8
6	88.6 ± 3.4	86.4 ± 5.2	92.5 ± 5.4	100 ± 0	51.4 ± 5.5	39.9 ± 7.2	42.1 ± 12.3	99.5. ± 0.4
7	88.8 ± 3.7	86.9 ± 5.0	92.6 ± 5.0	100 ± 0	53.0 ± 5.8	41.1 ± 7.7	45.7 ± 11.7	99.6. ± 0.3
8	89.3 ± 3.6	85.2 ± 5.7	91.4 ± 7.0	100 ± 0	56.1 ± 5.8	45.0 ± 8.7	43.8 ± 13.4	99.7. ± 0.3
9	88.1 ± 3.8	82.4 ± 5.6	89.1 ± 9.5	100 ± 0	53.3 ± 4.9	42.7 ± 8.5	39.2 ± 12.9	99.6. ± 0.3
10	87.0 ± 3.5	80.3 ± 5.9	85.3± 10.3	100 ± 0	51.5 ± 5.3	42.4 ± 7.8	34.3 ± 12.2	99.5. ± 0.4
11	86.7 ± 4.2	76.6 ± 6.4	79.7± 13.9	99.9 ± 0	52.4 ± 5.3	43.3 ± 8.5	31.4 ± 12.8	99.3. ± 0.5
12	86.4 ± 3.4	63.8 ± 7.9	64.3± 19.7	99.9 ± 0	52.8 ± 4.5	32.3 ± 8.7	26.1 ± 21.9	98.9. ± 0.8

Table 2: Multilingual benchmark accuracy results. With random sampling (left block), ContraSim outperforms other similarity measures. Using FAISS (right block) further extends the gaps.

pairs it hasn't seen at training. Using FAISS sampling (right block) further extends the gaps. While CKA results dropped by $\approx 45\%$, DeepCKA dropped by $\approx 51\%$, and DeepDot dropped by $\approx 40\%$, ContraSim was much less affected by FAISS sampling ($\approx 17\%$ drop on average and practically no drop in most layers). This demonstrates the high separability between examples of ContraSim, enabling it to distinguish even very similar examples. For all other methods, mid-layers have the highest accuracy, whereas for our method almost all layers are near 100% accuracy, except for the first 3 or 4 layers.

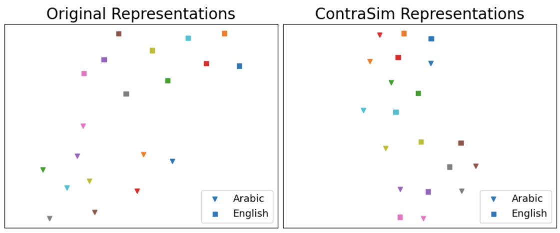

Figure 4: Original representations (left) are clustered by the source language (by shape). ContraSim (right) projects representations of the same sentence in different languages close by (by color).

To further analyze this, we compare the original multilingual representations from the last layer with their projections by ContraSim's trained encoder. Figure 4 shows UMAP (McInnes et al., 2018) projections for 10 English sentences and 10 Arabic sentences, before and after ContraSim encoding. The ContraSim encoder was trained on Arabic and English languages. The original representations are organized according to the source language (by shape), whereas ContraSim projects translations of the same sentence close to each other (clustered by color).

6.3 Image–caption benchmark

6.3.1 Setup

Given a test set $\mathbb{X}$, consisting of pairs of an image representation generated by a CV model and its caption representation from a LM, we split $\mathbb{X}$ to batches of size 64. For each batch, we compute the similarity between the image representations and their corresponding caption representations. We then sample 10 different caption batches, either randomly or using FAISS (as before), and compute the similarity between the image representation and each random/FAISS-retrieved caption representation. If the highest similarity is between the image representation and the original caption representation, we mark a success. For trainable similarity measures, we train with 5 different random seeds and average the results.

Models and Data. We use two vision models for image representations: ViT and ConvNext (Liu et al., 2022); and two language models for text representations: BERT and GPT2 (Radford et al., 2019). We use the Conceptual Captions dataset (Sharma et al., 2018). We use 5000 and 10000 pairs as test and training sets, respectively.

Positive and Negative sets. Given a batch of image representations with their corresponding caption representations, for each image representation

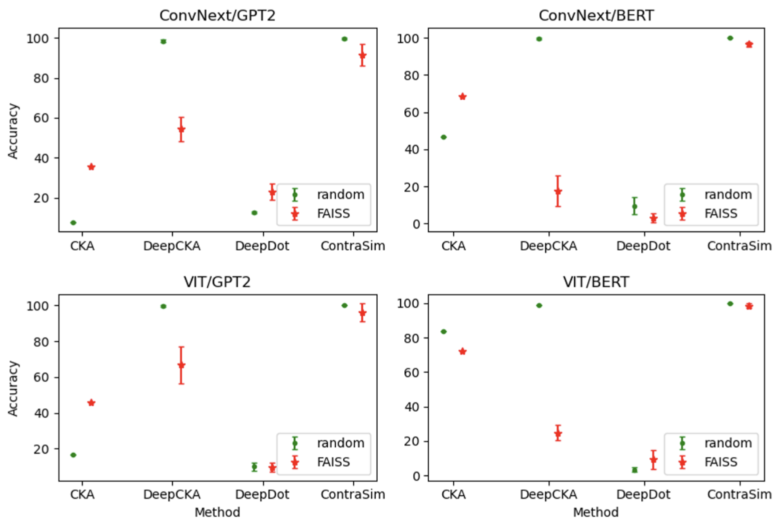

Figure 5: Image–caption benchmark results for 4 different model pairs. ContraSim works best, and is the only measure robust to FAISS sampling.

we define as a positive set its corresponding caption representation, and as a negative set all other representations in the batch.

6.3.2 Results

Figure 5 demonstrates the strength of ContraSim. Under random sampling (green boxes), DeepCKA achieves comparable results to ContraSim, while DeepDot and CKA achieve lower results. However, using FAISS (red boxes) causes a big decrease in DeepCKA accuracy, while ContraSim maintains high accuracy. Furthermore, in 3 of 4 pairs we tested, FAISS sampling yielded better CKA accuracy than random sampling. This contradicts the intuition that similar examples at the sampling stage should make it harder for similarity measures to distinguish between examples. This might indicate that CKA suffers from stability issues. Finally, we report results with the multi-modal CLIP model (Radford et al., 2021) in Table 6 (Appendix A.2). Because the model was pre-trained with contrastive learning, simple dot-product similarity works very well, so there is no need to learn a similarity measure in this case.

6.3.3 ContraSim on different dimensions

We further evaluated ContraSim with different representation dimensions. We performed the same image–caption benchmark with the exception that we used different vision model sizes: ViT-large and ConvNext-base, both with 1024-dimensional representation vectors. As language models we used the same GPT2 and BERT models, with a 768-dimensional representation vector.

	CKA	ContraSim
ViT-large/GPT2	45.57	98.73
ViT-large/BERT	83.54	98.73
ConvNext-base/GPT2	39.24	100
ConvNext-base/BERT	74.68	98.73

Table 3: Image–caption benchmark accuracy results for model pairs with different dimensions. We report results using FAISS sampling. Despite different model dimensions, ContraSim consistently works best.

We trained a different encoder for each model, as opposed to the single encoder we trained in all other experiments. This enables ContraSim to be used with representations with different dimensions. Results are summarized in Table 3. We report results with FAISS sampling. Across all pairs, ContraSim achieves superior results.

7 Interpretability insights

Having shown the superiority of our method, we now discuss a few interpretability insights that arise from our evaluations, and are not revealed by previous similarity measures.

In the multilingual benchmark (Table 2, FAISS results), we found a much greater difference in accuracy between shallow and deep layers in ContraSim compared to previous similarity measures. Using previous similarity measures we might infer that there is no difference in the ability to detect the correct pairs across different layers. However, ContraSim shows that the difference in the ability to detect the correct pair dramatically changes from shallow to deep layers. This raises an interesting insight regarding the evolution of representations across layers. For instance, Conneau et al. (2020b) used CKA to measure the similarity of representations of bilingual models of different languages and the same language (using back-translation). From their results it can be observed that there is no much difference in similarity of sentences from different languages and from the same language at the shallow and deep layers. Our results show that this difference is higher than found before.

In the image–caption benchmark (Figure 5), from the CKA results we might infer that BERT representations are more similar to computer vision representations than GPT2 representations. That is because with CKA, it is easier to detect the matching image–caption pair with BERT than it is with GPT2. However, ContraSim achieves a high accuracy in both BERT pairs and GPT2 pairs, which means that both language models about as similar to vision models, in contrast to what we may infer from previous similarity measures. This reveals a new understanding regarding the relationship between language and vision models. To the best of our knowledge, no prior work has done such a similarity analysis.

8 Conclusion

We proposed a new similarity measure for interpreting neural networks, ContraSim. By defining positive and negative sets we learn an encoder that maps representation to a space where similarity is measured. Our method outperformed other similarity measures under the common layer prediction benchmark and two new benchmarks we proposed: the multilingual benchmark and the image–caption benchmark. It particularly shines in strengthened versions of said benchmarks, where random sampling is replaced with finding the most similar examples using FAISS. Moreover, we show that even when ContraSim is trained on data from one domain/task and evaluated on data from another domain/task, it achieves superior performance. Considering ContraSim's superiority in all evaluations, we believe it is a better tool for the interpretability of neural networks, and have discussed a few insights revealed by ContraSim and not captured by previous methods.

Our new similarity measure benchmarks can facilitate work on similarity-based analyses of deep networks. The multilingual benchmark is useful for work on multilingual language models, while the image–caption benchmark may help in multimodal settings. In addition, since our method learns a parameterized measure, it may help train models with similarity objectives.

9 Limitations

Compared to existing methods, ContraSim needs access to a training set for the encoder training procedure. The training procedure itself is efficient, typically a matter of minutes.

10 Ethics Statement

Our work adds to the body of literature on the interpretability of neural networks and may mitigate their opacity. We do not foresee major risks associated with this work. However, a malicious actor could train ContraSim adversarially, assign poor similarity estimates, and lead to false analyses.

Acknowledgements

This work was supported by an AI alignment grant from Open Philanthropy, the Israel Science Foundation (grant No. 448/20), and an Azrieli Foundation Early Career Faculty Fellowship.

References

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. 2020. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2020a. Unsupervised cross-lingual representation learning at scale. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8440–8451, Online. Association for Computational Linguistics.

Alexis Conneau, Ruty Rinott, Guillaume Lample, Adina Williams, Samuel R. Bowman, Holger Schwenk, and Veselin Stoyanov. 2018. Xnli: Evaluating cross-lingual sentence representations. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

Alexis Conneau, Shijie Wu, Haoran Li, Luke Zettlemoyer, and Veselin Stoyanov. 2020b. Emerging cross-lingual structure in pretrained language models. In *Proceedings of the 58th Annual Meeting of* the Association for Computational Linguistics, pages 6022–6034.

Maksym Del and Mark Fishel. 2021. Establishing interlingua in multilingual language models. *arXiv preprint arXiv:2109.01207*.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

F Ding, J-S Denain, and J Steinhardt. 2021. Grounding representation similarity with statistical testing. *Advances in neural information processing systems*.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. 2020. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*.

Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. 2020. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9729–9738.

Harold Hotelling. 1936. Relations between two sets of variates. *Biometrika*, 28:321–377.

Thomas S Huang, Charlie K Dagli, Shyamsundar Rajaram, Edward Y Chang, Michael I Mandel, Graham E Poliner, and Daniel PW Ellis. 2008. Active learning for interactive multimedia retrieval. *Proceedings of the IEEE*, 96(4):648–667.

Ganesh Jawahar, Benoît Sagot, and Djamé Seddah. 2019. What does bert learn about the structure of language? In *ACL 2019-57th Annual Meeting of the Association for Computational Linguistics*.

Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2019. Billion-scale similarity search with GPUs. *IEEE Transactions on Big Data*, 7(3):535–547.

Andrej Karpathy, Armand Joulin, and Li F Fei-Fei. 2014. Deep fragment embeddings for bidirectional image sentence mapping. *Advances in neural information processing systems*, 27.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Simon Kornblith, Mohammad Norouzi, Honglak Lee, and Geoffrey Hinton. 2019. Similarity of neural network representations revisited. In *International Conference on Machine Learning*, pages 3519–3529. PMLR.

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

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

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.

Leland McInnes, John Healy, Nathaniel Saul, and Lukas Großberger. 2018. Umap: Uniform manifold approximation and projection. *Journal of Open Source Software*, 3(29):861.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2016. Pointer sentinel mixture models. In *International Conference on Learning Representations*.

Ari Morcos, Maithra Raghu, and Samy Bengio. 2018. Insights on representational similarity in neural networks with canonical correlation. *Advances in Neural Information Processing Systems*, 31.

Benjamin Muller, Yanai Elazar, Benoît Sagot, and Djamé Seddah. 2021. First align, then predict: Understanding the cross-lingual ability of multilingual BERT. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 2214–2231, Online. Association for Computational Linguistics.

Telmo Pires, Eva Schlinger, and Dan Garrette. 2019. How multilingual is multilingual BERT? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4996–5001, Florence, Italy. Association for Computational Linguistics.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. 2021. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning*, pages 8748–8763. PMLR.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Maithra Raghu, Justin Gilmer, Jason Yosinski, and Jascha Sohl-Dickstein. 2017. Svcca: Singular vector canonical correlation analysis for deep learning dynamics and interpretability. *Advances in neural information processing systems*, 30.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. 2015. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252.

Thibault Sellam, Steve Yadlowsky, Jason Wei, Naomi Saphra, Alexander D'Amour, Tal Linzen, Jasmijn Bastings, Iulia Turc, Jacob Eisenstein, Dipanjan Das, et al. 2021. The multiberts: Bert reproductions for robustness analysis. *arXiv preprint arXiv:2106.16163*.

Piyush Sharma, Nan Ding, Sebastian Goodman, and Radu Soricut. 2018. Conceptual captions: A cleaned, hypernymed, image alt-text dataset for automatic image captioning. In *Proceedings of ACL*.

John Wu, Yonatan Belinkov, Hassan Sajjad, Nadir Durrani, Fahim Dalvi, and James Glass. 2020. Similarity analysis of contextual word representation models. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4638–4655.

A Appendix

A.1 Multilingual benchmark

A.1.1 Evaluation parameters

We split the test set, $\mathbb{X}$, into equally sized batches of size 8, $\{b^{(1)}, b^{(2)}, ..., b^{(n)}\}$, where each batch consists of multilingual BERT representations of the same sentence in 5 different languages: $L = \{L_1, ..., L_5\}$. Given a pair of different languages, $(L_i, L_j), i \neq j$, and a batch of representations, b, we consider the representation of those languages in the batch, $(b[i], b[j])$, and compute the similarity between $b[i]$ and $b[j]$ as $s_0 \equiv s(b[i], b[j])$. We also compute the similarity between $b[i]$ and 10 randomly chosen batches (or, 10 batches chosen using FAISS) of representations in language L_j as $\{s_t \equiv s(b[i], b_t[j])\}_{t=1}^{10}$. If $\arg\max_t\{s_t\}_{t=0}^{10} = 0$, we count it as a correct prediction. Each layer's accuracy is defined as the number of successful predictions over the number of batches, n. We average results over all possible pairs of different languages.

A.1.2 Further evaluations

In addition to using the [CLS] token representation as a sentence representation, we also evaluated the multilingual benchmark using mean pooling sentence representation. We used the same evaluation process as described in Section 6.2. The results, summarized in Table 4, are consistent with the results in the main paper (Table 2). Under random sampling, ContraSim outperforms all other similarity measures. Using FAISS causes a big degradation in all other methods' accuracy, while ContraSim maintains a high accuracy across all layers.

In addition, we evaluated the multilingual benchmark with another multilingual model – the XLM-R (Conneau et al., 2020a) model. Results, summarized in Table 5, show a similar pattern to Tables 2 and 4, with ContraSim achieving the highest accuracies across all layers, in both random sampling and FAISS sampling scenarios.

A.2 Image–caption

In addition to the four model pairs we evaluated in Figure 5, we assessed the multi-modal vision and language CLIP model (Radford et al., 2021), which was trained using contrastive learning on pairs of images and their captions. Results in Table 6 show interesting findings. Under random sampling, dot product, DeepCKA and ContraSim achieve perfect accuracy. However, using FAISS causes significant degradation in DeepCKA accuracy, and only a small degradation in dot product and ContraSim results, with equal accuracy for both. We attribute this high accuracy for simple dot product to the fact that CLIP training was done using contrastive learning, thus observing high separability between examples.

A.3 ViT training details

We used the ViT-base (Dosovitskiy et al., 2020) architecture. We pretrained 10 models on the ImgaeNet-1K dataset (Deng et al., 2009), differing only in their weight initializations by using random seeds from 0 to 9. We used the AdamW optimizer (Kingma and Ba, 2015) with lr $= 0.001$, weight decay $= 1e - 3$, batch size $= 128$, and a cosine learning scheduler. We trained each model for 150 epochs and used the final checkpoint.

Then, we fine-tuned the pretrained models on CIFAR-10 and CIFAR-100 datasets (Krizhevsky et al., 2009). We used AdamW optimizer with lr $= 2e{-}5$, weight decay $= 0.01$, batch size $= 10$, and a linear scheduler. For models fine-tuned on CIFAR-10, the average accuracy on the CIFAR-10 test set is 96.33%. For models fine-tuned on CIFAR-100, the average accuracy on the CIFAR-100 test set is 78.87%.

A.4 Details of Prior Similarity Measures

Canonical Correlation Analysis (CCA). Given two matrices, CCA finds bases for those matrices, such that after projecting them to those bases the projected matrices' correlation is maximized. For $1 \leq i \leq p_1$, the i^{th} canonical correlation coefficient ρ_i is given by:

$$\begin{aligned} \rho_i = \max_{w_X^i, w_Y^i} &\operatorname{corr}(Xw_X^i, Yw_Y^i) \\ \text{s.t.} \quad &\forall_{j<i} \; Xw_X^i \perp Xw_X^j \\ &\forall_{j<i} \; Yw_Y^i \perp Yw_Y^j. \end{aligned} \tag{5}$$

where $\operatorname{corr}(X, Y) = \frac{\langle X, Y \rangle}{\|X\| \cdot \|Y\|}$. Given the vector of correlation coefficients $\text{corrs} = (\rho_1, ..., \rho_{p_1})$, the final scalar similarity index is computed as the mean correlation coefficient:

$$S_{\text{CCA}}(X, Y) = \overline{\rho}_{CCA} = \frac{\sum_{i=1}^{p_1} \rho_i}{p_1} \tag{6}$$

as previously used in (Raghu et al., 2017; Kornblith et al., 2019).

	Random				FAISS			
Layer	CKA	DeepCKA	DeepDot	ContraSim	CKA	DeepCKA	DeepDot	ContraSim
1	87.7 ± 6.9	86.3 ± 9.7	43.4 ± 17.7	98.7 ± 2.2	67.6 ± 14.3	54.1 ± 10.9	41.7 ± 19.1	94.2 ± 10.7
2	89.0 ± 6.3	88.7 ± 7.0	51.5 ± 20.0	99.5 ± 0.8	68.2 ± 13.2	49.0 ± 8.3	38.9 ± 17.2	96.6 ± 14.8
3	91.8 ± 4.4	90.7 ± 6.0	63.3 ± 20.8	99.9 ± 0.1	72.2 ± 11.5	55.4 ± 8.1	44.8 ± 16.6	98.8 ± 20.5
4	93.7 ± 3.3	91.3 ± 5.0	73.1 ± 19.4	99.9 ± 0.0	74.3 ± 10.0	55.1 ± 8.0	45.7 ± 16.5	99.5 ± 7.1
5	95.3 ± 3.0	92.1 ± 4.0	83.9 ± 15.6	99.9 ± 0.0	78.2 ± 8.2	56.7 ± 8.1	53.2 ± 17.5	99.8 ± 4.4
6	95.9 ± 2.4	91.8 ± 3.9	91.2 ± 10.6	100 ± 0	77.6 ± 7.9	54.2 ± 8.1	60.1 ± 18.2	99.8 ± 1.7
7	95.4 ± 2.5	90.6 ± 4.1	93.1 ± 9.2	100 ± 0	77.9 ± 7.8	53.3 ± 7.2	63.5 ± 18.5	99.9 ± 0.7
8	94.8 ± 3.2	89.7 ± 4.3	90.3 ± 12.0	100 ± 0	76.7 ± 8.1	52.4 ± 7.4	61.0 ± 19.8	99.9 ± 0.3
9	94.0 ± 3.4	88.5 ± 5.0	86.4 ± 15.1	100 ± 0	73.9 ± 8.8	51.4 ± 7.8	55.5 ± 20.0	99.9 ± 0.1
10	92.6 ± 4.2	85.6 ± 5.9	80.7 ± 18.8	100 ± 0	72.2 ± 8.4	49.3 ± 8.4	49.2 ± 20.6	99.9 ± 0.1
11	91.1 ± 5.1	81.0 ± 6.5	72.2 ± 23.7	99.9 ± 0	70.6 ± 10.1	48.8 ± 9.1	43.2 ± 20.7	99.8 ± 0.1
12	90.8 ± 5.8	71.3 ± 7.6	71.0 ± 21.0	99.9 ± 0	72.7 ± 11.3	40.3 ± 8.7	42.7 ± 17.0	99.4 ± 0.1

Table 4: Multilingual benchmark results with mean pooling.

	Random				FAISS			
Layer	CKA	DeepCKA	DeepDot	ContraSim	CKA	DeepCKA	DeepDot	ContraSim
1	89.7 ±. 4.6	88.2 ± 3.1	45.0 ± 22.1	99.89 ± 0.31	60.6 ± 11.4	24.6 ± 4.7	24.8 ± 12.4	98.1 ± 2.5
2	90.6 ± 4.1	92.3 ± 2.0	63.1 ± 23.4	99.99 ± 0.02	57.1 ± 11.4	30.7 ± 5.7	31.1 ± 13.5	99.5 ± 0.7
3	92.8 ± 3.1	93.8 ± 1.7	79.5 ±18.9	99.99 ± 0	55.5 ±10.1	33.8 ± 6.4	39.3 ±15.5	99.9 ± 0.1
4	94.7 ± 2.6	94.3 ± 1.7	91.4 ±11.7	100 ± 0	62.3 ± 9.5	36.3 ± 6.8	55.1 ±16.2	99.9 ± 0
5	95.9 ± 2.1	94.6 ± 1.6	94.0 ±10.0	100 ± 0	66.2 ± 8.4	37.2 ± 7.6	64.2 ±16.2	99.9 ± 0
6	95.9 ± 2.1	94.9 ± 1.6	94.6 ± 8.9	100 ± 0	66.7 ± 8.3	41.2 ± 7.6	66.0 ±17.5	99.9 ± 0
7	96.6 ± 2.0	94.9 ± 1.6	94.9 ± 8.5	100 ± 0	71.7 ± 8.5	44.1 ± 8.2	68.8 ±17.5	99.9 ± 0
8	96.1 ± 2.1	94.8 ± 1.7	94.0 ± 9.3	100 ± 0	68.1 ± 8.3	43.6 ± 7.9	65.3 ±18.0	99.9 ± 0
9	94.8 ± 2.2	94.9 ± 1.6	93.3 ± 9.1	100 ± 0	58.5 ± 8.7	42.8 ± 8.0	61.7 ±18.6	99.9 ± 0
10	93.9 ± 2.3	94.3 ± 1.7	92.7 ±10.7	100 ± 0	46.3 ± 8.2	39.1 ± 7.5	56.6 ± 18.9	99.9 ± 0
11	92.0 ± 2.8	93.3 ± 2.2	92.7 ± 10.9	100 ± 0	35.5 ± 7.0	39.1 ± 7.3	57.2 ± 18.3	99.9 ± 0
12	80.7 ± 4.7	89.5 ± 3.0	81.6 ± 13.8	100 ± 0	26.5 ± 5.8	32.3 ± 7.0	34.7 ± 15.1	99.9 ± 0

Table 5: Multilingual benchmark results on XLM-R model.

	Random	FAISS
CKA	93.67	25.31
Dot Product	100	98.73
DeepCKA	100	13.92
DeepDot	29.11	25.31
ContraSim	100	98.73

Table 6: Image–caption benchmark accuracy results using CLIP model

Projection-Weighted CCA (PWCCA). Morcos et al. (2018) proposed a different approach to transform the vector of correlation coefficients, corrs, into a scalar similarity index. Instead of defining the similarity as the mean correlation coefficient, PWCCA uses a weighted mean and the similarity is defined as:

$$S_{PW} = \frac{\sum_{i=1}^{p_1} \alpha_i \rho_i}{\sum_i \alpha_i} \quad \alpha_i = \sum_j |\langle h_i, x_j \rangle| \quad (7)$$

where x_j is the j^{th} column of X, and $h_i = X w_X^i$ is the vector observed upon projecting X to the i^{th} canonical direction.

Singular Vector CCA (SVCCA). Another extension to CCA, proposed by Raghu et al. (2017), performs CCA on the truncated singular value decomposition (SVD) of the activation matrices. SVCCA keeps enough principal components to explain a fixed percentage of the variance.

Code available at: `https://github.com/google/svcca`.

Centered Kernel Alignment (CKA). CKA, Proposed by Kornblith et al. (2019), suggests comput-

	Penn TreeBank	WikiText
SVCCA	46.66	56.66
Dot product	8.33	6.66
Norm	10.00	11.66
ContraSim_norm		
Penn	100.00	90.00
Wiki	100.00	100.00

Table 7: Layer prediction benchmark with additional similarity measures.

ing a kernel matrix for each matrix representation input, and defining the scalar similarity index as the two kernel matrices' alignment. For linear kernel, CKA is defined as:

$$S_{\text{CKA}} = \frac{\|Y^T X\|_F^2}{\|X^T X\|_F \|Y^T Y\|_F} \tag{8}$$

Code available at: https://github.com/google-research/google-research/tree/master/representation_similarity.

Norm. For two representations, x and y, we defined the dis-similarity measure as the norm of the difference between the normalized representations:

$$Dis_{\text{Norm}}(x, y) = \|(x/\|x\| - y/\|y\|)\| \tag{9}$$

Since this is a dis-similarity measure, we defined the norm similarity measure as:

$$S_{\text{Norm}} = 1 - Dis_{\text{Norm}}(x, y) \tag{10}$$

For a batch of representations, we define batch similarity as the mean of pairwise norm similarity.

Dot-product. Measuring the similarity score between two feature vectors as their dot-product.

A.5 Additional Evaluations

In this section, we report evaluation results with additional baselines. Furthermore, we evaluated ContraSim with a different similarity measure than the dot-product and replaced it with the norm similarity measure.

Similar to PWCCA, SVCCA requires that the number of examples is larger than the vector dimension, thus we could only evaluate it in the layer prediction benchmark. All other similarity measures were evaluated with all evaluations - the layer prediction benchmark, the multilingual benchmark and the image-caption benchmark.

Table 7 shows layer prediction benchmark results. We can observe that SVCCA achieves slightly better results than PWCCA, and lower than CKA and ContraSim. Both dot product and norm achieve low accuracies. ContraSim_norm achieves the same or better results than ContraSim.

Multilingual benchmark results, summarized in Table 8 show that both dot product and norm achieve better results than CKA, although achieve low results under layer prediction and image-caption benchmarks. This emphasizes the importance of multiple evaluations for similarity measures. Compared to ContraSim, both methods achieve lower results. ContraSim_norm achieves lower results compared to ContraSim, under both random and FAISS sampling.

Image-caption benchmark results, summarized in Table 9, show that under both random sampling and FAISS sampling dot product and norm achieve low accuracy. Under random sampling, ContraSim_norm achieves perfect accuracy, while using FAISS sampling shows slight degradation compared to ContraSim.

Layer	Random			FAISS		
	Dot product	Norm	ContraSim_norm	Dot Product	Norm	ContraSim_norm
1	48.93 ± 7.07	68.67 ± 10.86	89.39 ± 14.46	15.00 ± 6.41	20.42 ± 9.32	15.76 ± 6.56
2	31.71 ± 3.31	74.19 ± 9.73	85.40 ± 18.00	20.14 ± 2.75	21.16 ± 11.43	18.83 ± 11.19
3	49.32 ± 6.53	83.92 ± 13.12	85.68 ± 18.88	13.00 ± 4.50	29.42 ± 22.99	26.36 ± 16.01
4	29.60 ± 2.44	99.61 ± 0.41	96.81 ± 5.04	16.20 ± 4.01	57.98 ± 15.68	28.79 ± 9.80
5	99.75 ± 0.29	99.86 ± 0.33	99.86 ± 0.23	82.17 ± 7.36	82.39 ± 7.73	74.04 ± 7.11
6	99.75 ± 0.35	99.84 ± 0.27	99.92 ± 0.13	83.38 ± 7.70	88.24 ± 6.15	77.00 ± 6.68
7	99.52 ± 0.77	99.85 ± 0.29	99.92 ± 0.13	89.72 ± 5.57	89.23 ± 6.98	78.21 ± 6.77
8	99.93 ± 0.13	99.89 ± 0.15	99.94 ± 0.11	93.49 ± 4.07	89.70 ± 6.42	81.97 ± 6.63
9	99.61 ± 0.38	99.76 ± 0.40	99.91 ± 0.17	82.48 ± 9.32	84.85 ± 8.57	81.37 ± 6.53
10	96.64 ± 2.46	99.38 ± 0.58	99.89 ± 0.15	55.02 ± 15.42	81.43 ± 9.77	80.59 ± 6.83
11	87.04 ± 8.13	98.40 ± 1.25	99.83 ± 0.31	29.62 ± 13.82	82.20 ± 9.46	80.74 ± 7.08
12	76.86 ± 15.23	87.25 ± 12.39	99.73 ± 0.36	25.75 ± 26.55	50.11 ± 32.55	80.24 ± 6.63

Table 8: Multilingual benchmark with additional similarity measures. The left block is with random sampling, and the right block is FAISS sampling.

Vision Model		ViT		ConvNext	
Language Model		BERT	GPT2	BERT	GPT2
Random	Dot Product	6.32	6.32	11.39	8.86
	Norm	6.32	7.59	15.19	7.59
	ContraSim_norm	100	100	100	100
FAISS	Dot Product	5.06	2.53	7.59	6.32
	Norm	5.06	5.06	6.32	10.12
	ContraSim_norm	93.67	98.73	81.03	93.67

Table 9: Image–caption benchmark results for additional similarity measures, on 4 different model pairs.