nature communications

Article

https://doi.org/10.1038/s41467-025-60809-y

On convex decision regions in deep network representations

Received: 4 September 2024

Accepted: 5 June 2025

Published online: 02 July 2025

Check for updates

Lenka Tětková, Thea Brüsch, Teresa Dorszewski, Fabian Martin Mager, Rasmus Ørtoft Aagaard, Jonathan Foldager, Tommy Sonne Alstrøm & Lars Kai Hansen

Current work on human-machine alignment aims at understanding machine-learned latent spaces and their relations to human representations. We study the convexity of concept regions in machine-learned latent spaces, inspired by Gärdenfors' conceptual spaces. In cognitive science, convexity is found to support generalization, few-shot learning, and interpersonal alignment. We develop tools to measure convexity in sampled data and evaluate it across layers of state-of-the-art deep networks. We show that convexity is robust to relevant latent space transformations and, hence, meaningful as a quality of machine-learned latent spaces. We find pervasive approximate convexity across domains, including image, text, audio, human activity, and medical data. Fine-tuning generally increases convexity, and the level of convexity of class label regions in pretrained models predicts subsequent fine-tuning performance. Our framework allows investigation of layered latent representations and offers new insights into learning mechanisms, human-machine alignment, and potential improvements in model generalization.

With the proliferation of machine learning-based AI and applications in critical domains such as biomedicine, public service, and education, understanding the barriers to human-machine alignment is as important as ever (see, e.g.[1,2]). Alignment modeling is hampered by human representations being only indirectly accessible[3], and AI representations, while directly observable, are very complex to decode[4]. Yet, faced with these challenges, we seek to understand representational alignment as a first step towards a greater goal of value alignment[5]. To further such understanding, it is fundamental to establish a common language for the regularities observed in human and machine representations. Here, we motivate and introduce the concept of convexity of object regions in machine-learned latent spaces.

Human representational spaces are described in several ways, for example, geometric psychological spaces informed by similarity of judgments or, based in the neurosciences, where representations can be derived from the measurement of neural activity[3,6,7]. Conceptual spaces as proposed by Gärdenfors are a mature approach to the former, i.e., human-learned geometrical representations of semantic similarity[8]. The geometrical approach is rooted in the work by Shepard[9], which opens with the important observation: "Because any object or situation experienced by an individual is unlikely to recur in exactly the same form and context, psychology's first general law should, I suggest, be a law of generalization". This leads Shepard to favor geometrical representations in which concepts are represented by extended regions rather than single points, to allow for robust generalization. This view has been comprehensively expanded and quantified in[8]. The cognitive science insights are complemented by extant work investigating alignment between learned representations in machine and human conceptual spaces[10–12], and numerous specific properties of the latent geometrical structure have been studied, such as the emergence of semantic separability in machine latent representations[13]. New insights in representational geometry are found using the intrinsic dimension measure[12]. The relevant geometries are not necessarily flat Euclidean spaces but are often better described with general manifolds[14,15]. In fact[16,17], suggest that semantic separability emerges by flattening or straightening trajectories in latent spaces. Similar reasoning was crucial for early methodological developments like ISOMAP[18] and machine learning with 'kernel' methods[19].

Section for Cognitive Systems, DTU Compute, Technical University of Denmark, 2800 Kongens Lyngby, Denmark. e-mail: lenhy@dtu.dk

The geometry of latent spaces is a topic of significant interest in machine learning[20]. Extant work focuses on learning convex manifolds that allows for latent space interpolation in generative models with the aim of avoiding out-of-distribution samples[20–24] or to create augmented data sets[24], e.g., to reduce class imbalance in decision problems[25]. Additionally, enforcing Euclidean convexity can help Euclidean distance-based clustering in the latent space[26].

Another line of related work studies the geometry of specific deep structures based on so-called ReLU-units. Such networks are piecewise linear functions and the linear regions are convex polytopes by conjunction. It has been observed that ReLU-networks both at initialization and after training are surprisingly smooth, viz., the convex linear regions are larger and have much fewer facets than potentially could be realized with a particular architecture[27]. In the interesting work on the polytope lens, this discussion is related to the representation of semantic structure, speculating that polytope boundaries could be related to small semantic shifts[28]. As a polytope boundary is implemented by a single neuron changing its state, a boundary would naturally connect different, but related semantics (assuming that changing the state of a single neuron has limited semantic impact). Indeed, if all boundaries of a given polytope induce a semantic shift, the center polytope would represent a single semantic component, a concept. On the other hand, if a concept extends over multiple nearby polytopes, we cannot conclude convexity - as the union of convex sets may be non-convex. A detailed comparison of graph convexity and the polytope lens can be found in Supp. Mat. 2.4.

We also note the recent flurry of interest in decoding dense representations using sparse autoencoders[29] to disentangle semantics in generative models. This work suggests more complex relations, i.e., polysemantic neurons, and less impact on the individual neurons. Hence, in the following, we will build empirical tests for the convexity of such regions. Networks with more general activations, such as softmax units, may create smoother boundaries leading to more general non-linear manifolds[20], therefore we consider not only Euclidean convexity but also geodesic convexity, estimated using graph analytic techniques.

Based on Shephard's idea of objects as extended regions, Gärdenfors formulated the hypothesis that natural concepts form convex regions in human geometrical representations[8,30–32]. Strößner[33] elaborated on the notion of natural concepts as a social construct: "[Natural concepts] are often found in the core lexicon of natural languages-meaning that many languages have words that (roughly) correspond to such concepts-and are acquired without much instruction during language acquisition." One way to interpret the naturalness notion is to link it to independent physical mechanisms with macroscopic effects, i.e., effects that will be visible to all, hence, likely to appear in joint vocabularies. Such independent mechanisms are essential in causal modeling[34]. A more low-level interplay between human and machine conceptual representations was discussed in[35] with a specific focus on the grounding of shape spaces. The work reports good correspondences between human shape representations as obtained by pairwise similarity judgments and machine representations of the shape obtained from supervised and unsupervised learning, however, without touching the question of the convexity of object regions in machines.

Convexity is closely related to generalization in cognitive systems[36,37]. The defining property of convexity (see Def. 1) implies that categorization can be extended by interpolation. We also note that simple generalization based on closeness to prototypes leads to convex decision regions (Voronoi tesselation induces convex regions, see the proof in Supp. Mat. 1.5)[38]. Interestingly, convexity is also claimed to support few-shot learning[36]. When basic concepts are learned as convex regions, new labels can be formed by geometrically guided composition, leading to new convex regions (e.g. by conjunction) or by other inductions leading to sets of convex regions. Finally, it is observed that convexity supports communication and interaction and thus the negotiation of meaning between subjects and the emergence of socially universal concepts, i.e., natural concepts[31].

The geometry-driven cognitive science insights motivate our investigation here: Are generalizable, grounded decision regions implemented as convex regions in machine-learned representations?

The convexity of decision regions in machine-learned representations is new to our line of work and therefore, we first need to develop and describe the required analytical tools. In the remainder of our paper, we carefully develop the key dimensions of our approach: (1) Definitions of Euclidean and graph convexity; (2) Mathematical analysis of the stability of convexity to relevant latent space transformations; (3) Properties of workflows to measure Euclidean and graph convexity of decision regions in latent spaces; (4) Empirical evidence of pervasive convexity of decision regions in self-supervised models for images, audio, human activity recognition, text, and medical images. We finally present evidence that the convexity of a class' decision region in a 'pretrained' model (i.e., trained by label-free self-supervision) predicts the labeling accuracy of that class following fine-tuning. Our approach is illustrated in Fig. 1.

Results

Properties of euclidean convex sets

A set in Euclidean spaces is convex if for every two points from this set, the segment (i.e. shortest path, denoted $\mathbf{y}(t)$ with $t \in [0, 1]$), between these points is also within the set. Formally:

Definition 1. (Euclidean convexity). A subset $S \subset \mathbb{R}^D$ is convex iff $\forall\ \mathbf{x}, \mathbf{y} \in S\ \ \forall\ t \in [0, 1],\ \mathbf{y}(t) = t\mathbf{x} + (1 - t)\mathbf{y}$ is also in S[39].

From the definition, it follows that the intersection of two convex sets is also a convex set. Hence, conceptual conjunction (AND operation) preserves convexity. Disjunctions (OR operations), however, do not, since the union of convex sets is not necessarily convex (it is trivial to construct counter-examples)[39]. Euclidean convexity is conserved under affine transformations, and hence convexity is robust to relevant latent space transformations in deep networks (see a formal proof in Supp. Mat. 1.2). Euclidean convexity is closely related to conjunctions of linear classifiers. In fact, a convex set can alternatively be defined as the intersection of linear half-spaces (possibly infinite), e.g. implemented by a set of linear decision functions resulting in a polyhedron[39].

The relevant geometric structure of deep networks is not necessarily Euclidean, hence, we will also investigate the convexity of decision regions in data manifolds with more general geometry – Riemannian manifolds. The generalization of a segment to Riemannian manifolds is a geodesic, hence, we need to use geodesics as shortest paths between two points on a manifold. Formally, in a Riemannian manifold M with metric tensor g, the length L of a continuously differentiable curve $\mathbf{y}: [0, 1] \to M$ is defined by $L(\mathbf{y}) = \int_0^1 \sqrt{g_{\mathbf{y}(t)}(\dot{\mathbf{y}}(t), \dot{\mathbf{y}}(t))}dt$, where $\dot{\mathbf{y}}(t) := \frac{\partial}{\partial t}\mathbf{y}(t)$. A geodesic from $\mathbf{x}$ to $\mathbf{y}$ is a curve connecting $\mathbf{y}(0) = \mathbf{x}$ and $\mathbf{y}(1) = \mathbf{y}$, minimizing this length, i.e. $\text{geodesic}(\mathbf{x}, \mathbf{y}) = \arg\min_{\mathbf{y}} L(\mathbf{y})$. While geodesics are unique for Euclidean spaces, they may not be unique in manifolds. We can now generalize to geodesic convexity in manifolds:

Definition 2. (Geodesic convexity). A region $S \in M$ is geodesic convex, iff $\forall\ \mathbf{x}, \mathbf{y} \in S$, there exists at least one geodesic $\mathbf{y}(t)$ connecting $\mathbf{x}$ and $\mathbf{y}$, that is entirely contained in S.

When modeling latent spaces with sampled data, we must further transform the above definitions to data-driven estimators, such efforts are reported, e.g. in[14,15]. In this work, we choose a simple approach inspired by ISOMAP[18], hence based on graph convexity in data manifolds.

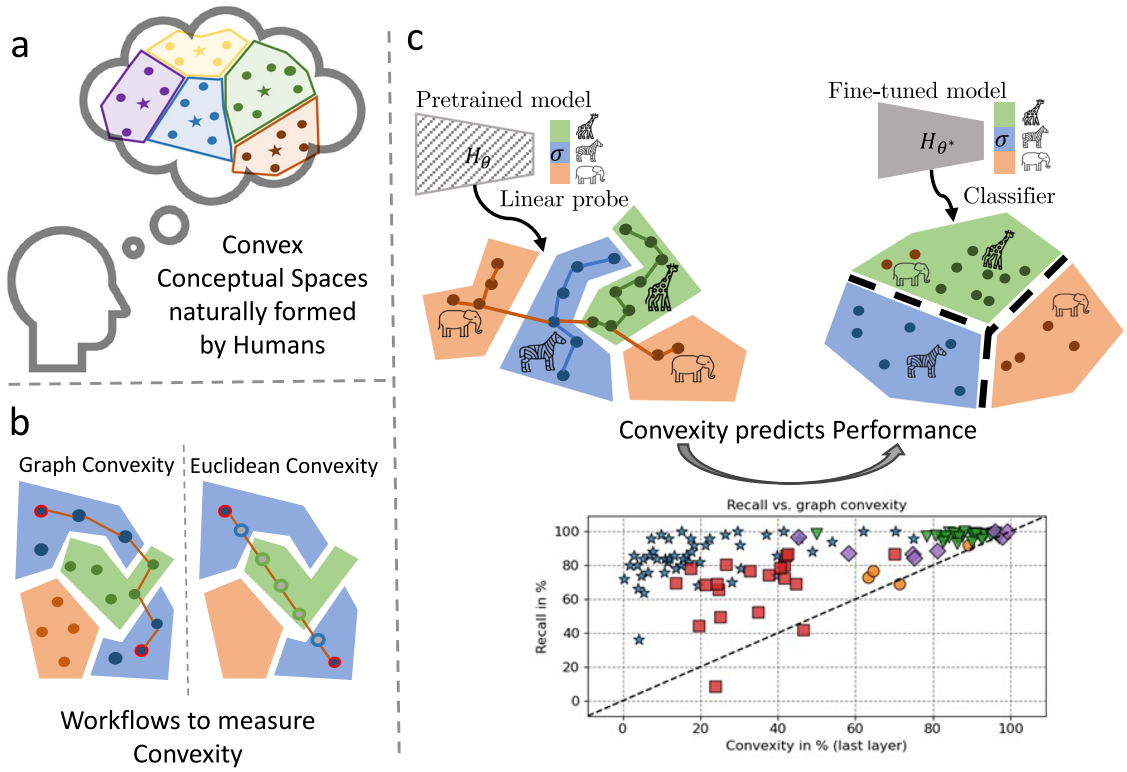


Fig. 1 | Convex decision regions in networks. a Our work is inspired by Gärdenfors' conceptual spaces[36] where similar objects (•) are grouped together and natural concepts (often defined by prototypes (⋆)) form convex regions[38]. **b** We develop two workflows to measure the convexity of data representations in latent spaces of neural networks. Graph convexity is based on the shortest path between two points of the same class through the nearest neighbors, with the convexity score being the proportion of points in the path belonging to the same class. For Euclidean convexity, we interpolate linearly between two points of the same class in the latent space and classify each interpolated point. The convexity score is the proportion of interpolated points being classified in the same class. **c** We measure the convexity of decision regions defined by the linear probe (i.e., linear classifier) in two types of models: models pretrained by self-supervised training, and the same models fine-tuned for classification. We present evidence that higher convexity of the decision regions of the pre-trained model is associated with higher recall in the fine-tuned model.

Definition 3. (Graph convexity, see e.g.[40]). Let (V, E) be a graph and $A \subseteq V$. We say that A is convex if for all pairs $x, y \in A$, there exists a shortest path $P = (x = v_0, v_1, v_2, \ldots, v_{n-1}, y = v_n)$ and $\forall\, i \in \{0, \ldots, n\}$: $v_i \in A$.

For reasonably sampled data, we can form the graph based on Euclidean nearest neighbors (as in ISOMAP), effectively creating an undirected, weighted graph. The shortest path between two points within such a graph is defined as the path, which minimizes the sum of weights of the edges[41].

We note two important properties of this estimator, first, the graph-based approximate convexity measure is invariant to isometric transformation and uniform scaling (see a formal proof in Supp. Mat. 1.3), and second, the sample-based estimator of convexity is consistent.

As we will measure convexity in labeled subgraphs within larger graphs, Dijkstra's algorithm is preferred over the Floyd-Warshall algorithm used in ISOMAP. Dijkstra's algorithm finds the shortest path from a given node to each of the other N nodes in the graph with E edges in $\mathcal{O}(N \log N + E)$[41,42], while Floyd-Warshall efficiently finds the shortest distance between all vertices in the graph in $\mathcal{O}(N^3)$[43,44]. As we have a sparse graph with $E \ll N^2$, Dijkstra's algorithm will be more efficient. With these approximations, we are in a position to create a graph-based workflow for quantifying convexity in Euclidean and manifold-based structures. Note that for sampled data, we expect a certain level of noise, and hence convexity will be graded.

The consistency of sample/graph-based geodesic estimates has been discussed in connection with the introduction of ISOMAP[45] and more generally in[46]. Graph connectivity-based estimates of geodesics from sample data are implemented using two procedures: The neighborhood graph can be determined by a distance cutoff ϵ, so that any points within the Euclidean distance ϵ are considered neighbors, or by K-nearest neighbors (KNN) based on the Euclidean distance. Consistency of these estimates, i.e., that the sample-based shortest paths converge to geodesics, is most straightforward to prove for the former approach[45,46]. The consistency proof for the ϵ-based procedure is based on the smoothness of the metric, a uniformly bounded data distribution ($1/c < p(x) < c$) and scaling of the distance cutoff or K so the connectivity (number of edges per node) increases for large samples (cutoff decays slowly $\epsilon \to 0$ as sample size $N \to \infty$)[46].

In finite samples, a (too) large connectivity will bias the geodesics, while a (too) small connectivity can lead to disconnected graphs and noisy estimates. We explore the role of the way we construct the graph. The consistency proof holds for a graph constructed with a distance cutoff ϵ. We compare it to graphs constructed by keeping K-nearest neighbors and symmetrizing the graph. We chose the number of nearest neighbors to be 3, 5, 10 and 20. The respective ϵ values were chosen to keep approximately the same number of edges in the graph.

Figure 2 d shows that the graph constructed with a distance cutoff ϵ is very disconnected, and the scores computed with this type of neighborhood are biased towards zero (see Fig. 2e). If we skip the disconnected pairs and compute the score only from existing paths, the results are very close to the scores based on K-nearest neighbors (see Fig. 2f). All experiments also demonstrate that the role of the size of K is negligible. Therefore, we choose a KNN-based approach with $K = 10$ in the complete set of experiments.

https://doi.org/10.1038/s41467-025-60809-y

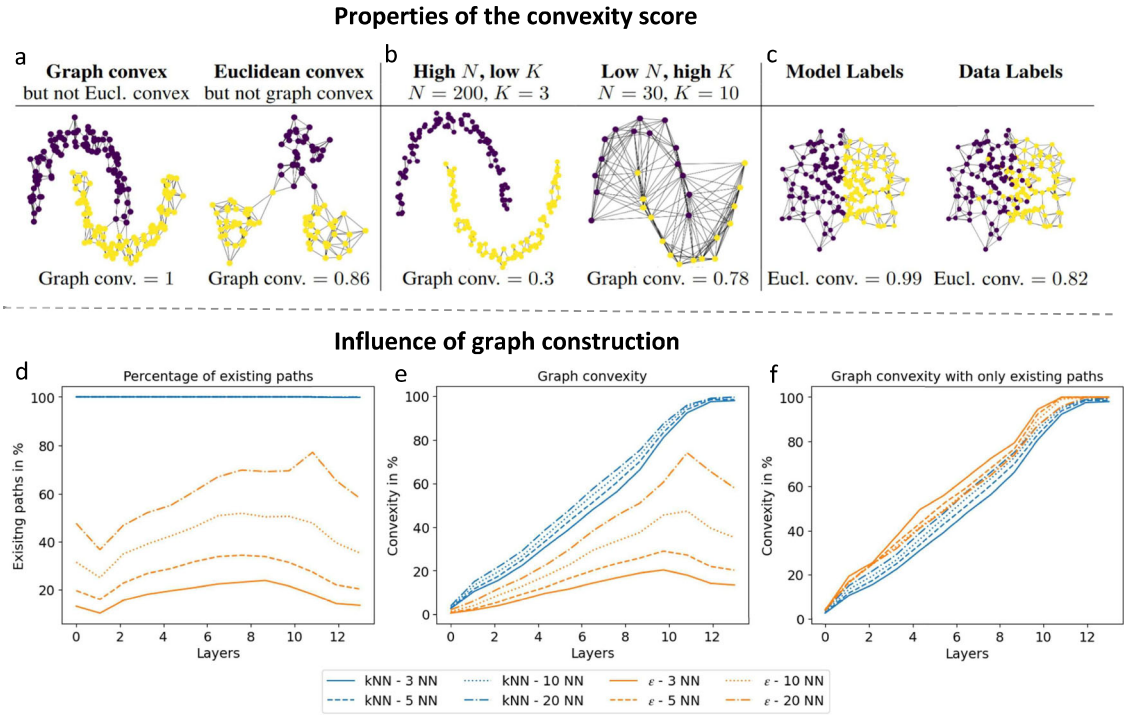


Fig. 2 | Properties of the graph convexity score. a Graph convexity can be more general than Euclidean convexity for a sufficiently large number of data points, N, and an appropriate number of nearest neighbors, K. An Euclidean convex set can have a graph convexity score lower than 1 in the case of isolated subgraphs of one class, connected only through the other class. **b** Smaller values of K can lead to a disconnected graph with a low graph convexity score due to missing paths between the nodes from the same class. Small N may lead to edges to distant points that are then used in the shortest paths. **c** Difference between computing the graph convexity scores of the data labels vs. the model labels in the last layer. Model labels create Euclidean-convex regions (left), which is not the case for data labels (right) if the model's accuracy is lower than 100%. **d** The graph constructed with a distance cutoff ϵ is very disconnected, resulting in many paths that don't exist, i.e. data points that cannot be connected via neighbors. The graph constructed using kNN is more complete, leading to all data points being able to connect via a path through their neighbors, independent of K. **e** The graph convexity scores computed with disconnected neighborhoods (based on ϵ) are biased towards zero, while the kNN-based graph leads to stable convexity scores, again independent of K. **f** If we skip the disconnected pairs and compute the score only from existing paths, the results of all graph construction methods are very similar. The results in (**d**)–(**f**) are computed on 100 ImageNet classes with data2vec.

Neural networks and convexity

Should we expect convexity of decision regions of neural network representations? In fact, several mechanisms could contribute to the promotion of convexity. First, the ubiquitous softmax is essentially a convexity-inducing device, hence, typical classification heads will induce convexity in their pre-synaptic activation layer. This is most easily seen by noting that softmax decision regions (maximum posterior decisions) are identical to the decision regions of a linear model, and linear models implement convex decision regions (see Supp. Mat. 1.1). Secondly, many recent models are based on transformer architectures with attention heads[47]. These heads contain softmax functions and thus induce convexity in their weighing of attention. Thirdly, typical individual artificial neurons, e.g. ReLUs, are latent half-space detectors, and half-spaces are convex as noted above. Note that deep multi-layer perceptrons can approximate any non-convex decision region, including disconnected decision regions[48]. These findings suggest that convexity is possible in neural networks, although it is not forced by design.

An extensive body of work concerns the geometry of input space representations in ReLU networks and has led to a detailed understanding of the role of so-called 'linear regions': In networks based on ReLU nonlinearity, the network's output is a piecewise linear function over convex input space polyhedra, formed by the intersection of neuron half-spaces[49–52]. Interestingly, in typical networks (e.g. at initialization or after training), the linear regions are much less in number and simpler in structure compared to theoretical upper bounds[50–52]. During training, the collection of convex linear regions is transformed and combined into decision regions through the later network layers. The resulting decision regions are, therefore, unions of convex sets and may in general be non-convex or non-connected, as noted in ref. 48. Our two measures of convexity, Euclidean and graph-based, probe different mechanisms of generalization, the former is associated with generalization by linear interpolation, while the latter is associated with generalization by interpolation along the data manifolds. As both mechanisms may be relevant for explaining generalization in cognitive systems, we are interested in the abundance and potential role of both measures of convexity. Note that a region may be convex in terms of the Euclidean measure but not the graph-based (e.g. a decision region formed by disconnected but linearly separated subsets) and a region may be graph-convex, but not Euclidean-convex (e.g., a general shaped connected region in the data manifold). For visual examples, see Fig. 2a.

Convexity measurement workflows

We developed and presented two workflows for measuring graph convexity and Euclidean convexity respectively in latent spaces of neural networks.

We are interested in measuring the approximate convexity of a decision region, here, a subset of nodes in a graph. A decision region for a specific class is a set of points that the model classifies as belonging to this class. We measure convexity per layer and per class separately.

We first create a graph containing the representations of the data points of all classes. The points are nodes in the graph and the Euclidean distances between the nodes are the weights of the edges. To handle manifold-based representation, for each node, we create an

undirected edge only to the nearest neighbors ($K = 10$). This procedure creates a sparse undirected weighted graph with positive weights (i.e. distances) only.

We now sample N_s pairs of points within the given predicted class label and compute the shortest path in the graph between the pairs using Dijkstra's algorithm[41]. For each path, we compute a score between 0 and 1. The path score is defined as the proportion of the number of nodes on the shortest path, without the endpoints, inside the decision region (i.e. with the same predicted label). If an edge directly connects the pair of nodes, the score is 1. If the points are not connected, the score is 0. We average the scores for all paths within one class to get a convexity score per class per layer. To get one convexity score per layer, we average over all paths regardless of class. Uncertainties are calculated as the standard error of the mean, where we set n as the number of points in the given class. This is likely a conservative estimate since the mean is based on many more pairs than there are points.

The runtime complexity of this procedure is $\mathcal{O}(N^2(D+1) + N_c N_s N(\log N + K))$, where N denotes the number of data points, D is the dimensionality of the representations and N_c is the number of classes. See Supp. Mat. 1.4 for details.

Euclidean convexity is also measured for each layer and class separately. To measure the convexity of class C in layer l, we first extract hidden representations at l of all points predicted to belong to class C. We sample N_s pairs of points. For each pair, we compute the $N_p = 10$ equidistant points on the linear segment connecting the representations of the two endpoints. We feed the interpolated points to the rest of the network (after layer l). The score for each pair is then the proportion of the interpolated points that are also predicted to belong to class C. Finally, to get the score of Euclidean convexity for the whole class, we average the scores over all the pairs of points. To get the Euclidean convexity for each layer, we average over all classes.

The runtime complexity of this procedure is $\mathcal{O}(N_c N_s N_p D)$, where N_c denotes the number of classes, and D is the dimensionality of the representations. See Supp. Mat. 1.4 for details.

To gain more intuition about the two types of convexity and their differences, we present examples of various properties of the graph and Euclidean convexity score on synthetic data. Figure 2a showcases two examples of data distributions that are either graph or Euclidean convex but not the other, this shows that graph convexity can be more general than Euclidean convexity and capture more complex structures. Figure 2b shows how different values of the sampled data (N) and nearest neighbors (K) influence the created graph and, therefore, also the graph convexity score. A high enough N and K is needed to guarantee a stable result. For too small N or K, the graph convexity score deteriorates and fails to provide a good estimate of the convexity in the data structures. In experiments (see Fig. 2d–f) we found that the influence of K is negligible given an appropriate N. Another consideration is the type of class labels, where we have two options: data labels (true classes) and model labels determined by the predictions of a model (decision regions). Figure 2c illustrates the difference between these two in the last layer of a model. From Theorem 3 in Supp. Mat. 1.3, we have that the last layer is always Euclidean convex for model labels. We focus on decision regions defined by model labels as they directly probe model representations and decision processes. Note that Euclidean convexity is always based on model labels since it is based on the prediction of interpolated points, while graph convexity can be computed for data as well as model labels.

For the pretrained models, we obtain the predicted labels by training a linear layer on top of the last hidden layer (with the rest of the model frozen). This procedure is similar to linear-classifier-based probes that are widely used in natural language processing to understand the presence of concepts in latent spaces[53,54]. A prominent example of probe-based explanation in image classification is the CAV (concept activation vector) scheme proposed by Kim et al.[55], in which auxiliary labeled data sets are used to identify concept directions with linear classifiers (concept class versus random data). In our approach, we use a multi-label probe of the last layer to identify the model labels and use these labels to compute and compare convexity throughout the network.

The score depends on the number of classes, and also on the number of predicted data points per class. It follows that the exact numbers are not directly comparable across modalities. A scale for convexity can be set using a null hypothesis that there is no relation. Under this null, we can estimate convexity with randomized class assignments and get a baseline score. For balanced datasets we find a baseline score of $1/C$ with C being the number of classes.

Measurements based on neighbors in high-dimensional data can be sensitive to the so-called hubness problem[56]. We evaluated the hubness of the latent representations in terms of k-skewness and the Robinhood score. Results are deferred to Supp. Mat., Section 2.1, since only mild hubness issues were detected for most domains. We decided to analyze convexity without adjustment for hubness, to avoid possible biases introduced by normalization schemes[56].

Convexity of latent representations before and after fine-tuning

To gain intuition on how and where convex regions form and the effect of fine-tuning on this process, we first inspect the t-SNE plots, which are a projection of high-dimensional data into 2D[57]. The t-SNE plots for a subset of layers of all modalities can be seen in Fig. 3a. From Theorem 3 in Supp. Mat. 1.3, we know that the last layer of both the pretrained and fine-tuned models is Euclidean convex. The t-SNE plot illustrates this quite clearly for the fine-tuned model, where the classes are clearly clustered and separated in the last layer. The picture is not quite as evident for the pretrained model, however, even here we see a clustering of the classes. We also see that points with the same predictions get clustered already within earlier layers. The observed structure in these low-dimensional representations adds motivation to our investigation of the convexity of class labels using Euclidean and graph methods in the high-dimensional latent spaces.

We investigate how convex regions develop across layers in the networks and the effect of fine-tuning on these decision regions. We find that convexity is pervasive both before and after fine-tuning, while convexity generally increases across layers within a model, and fine-tuning increases convexity. Figure 3b shows the results for all modalities. It is important to note that the magnitudes of the convexity scores differ among modalities because of different amounts of data and numbers of classes. However, the trend is the same for all modalities. Random labeling yields graph convexity scores of approximately $\frac{1}{C}$, where C is the number of classes. All the convexity scores are significantly higher than these baselines. Generally, class convexity is increased by fine-tuning. Note that the graph convexity in the last layer is (in some cases considerably) lower than 1 - 100%, even though this layer is always Euclidean convex. For detailed results of the analysis of individual data modalities including uncertainties, see Supp. Mat. 2.1, Supp. Figure 2.

Convexity and performance

To test the hypothesis motivated by cognitive science that convexity facilitates few-shot learning, we plot the post-finetuning accuracy (recall) per class versus the pretraining convexity scores in the last layer as shown in Fig. 3c for graph-based convexity. There is a strong association between both types of convexity measured in the pretrained model and the accuracy of the given class in the fine-tuned model. Indeed, there is a significant correlation in both cases. The Pearson correlation coefficient between Euclidean convexity and recall is 0.53 ± 0.06 and 0.52 ± 0.06 for graph convexity. In Supp. Mat. 2.5, we explore further the relation between graph convexity and recall, and find that the relation is best modeled by a linear relation.

https://doi.org/10.1038/s41467-025-60809-y

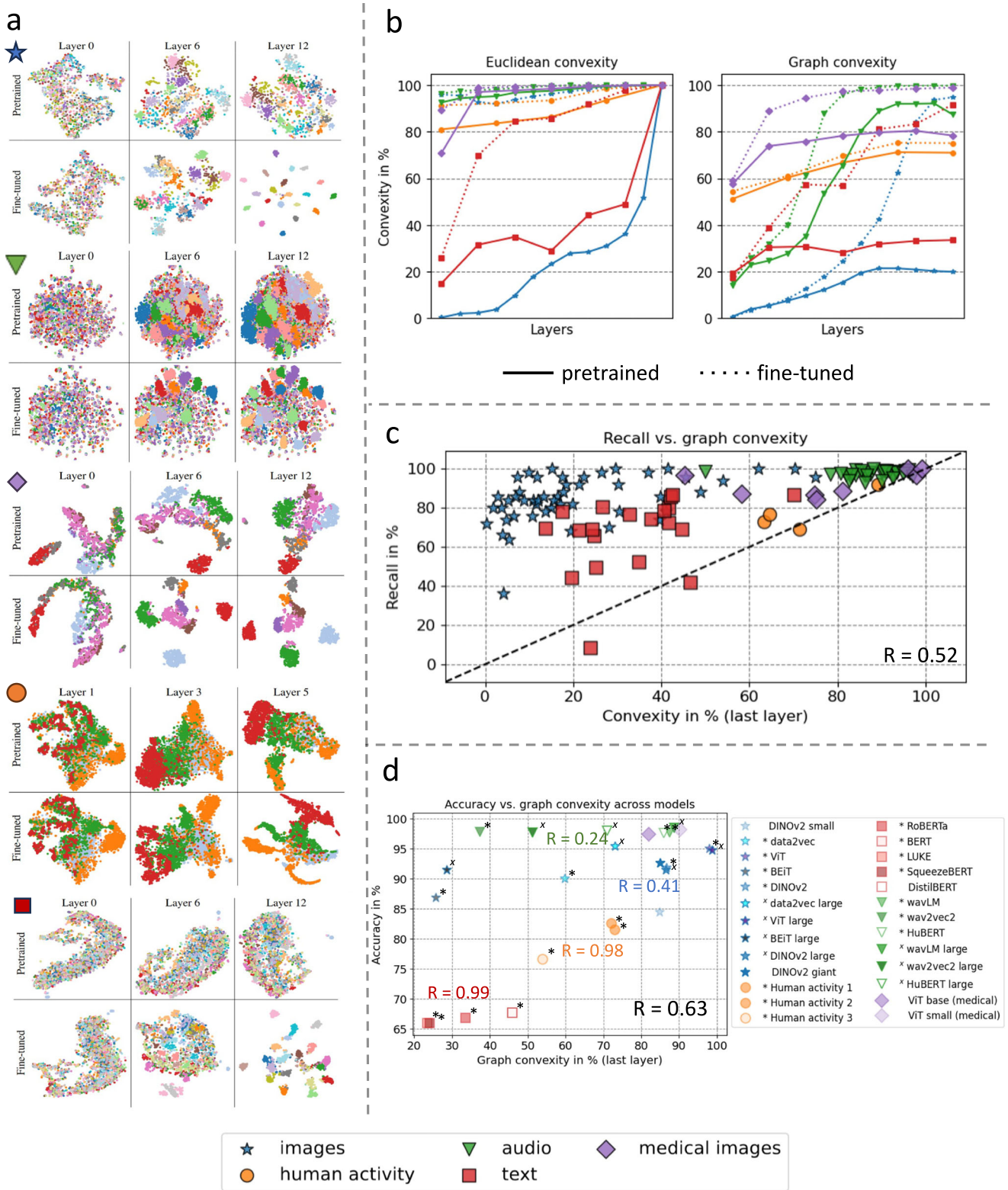


Fig. 3 | Results. a t-SNE plots show the forming of clustered regions already in pretrained models, which develop into more defined and clustered regions during fine-tuning. This gives an intuition of how convex regions form. **b** Euclidean and graph convexity scores for all modalities for decision regions of pretrained and fine-tuned networks. Decision regions are determined by model-predicted labels. Fine-tuning generally increases convexity scores. Note that the results are not directly comparable across modalities, as the convexity score depends on the number of classes. **c** Graph convexity of a subset of classes in the pretrained models (last layer) vs. recall rate of these individual classes in the fine-tuned models for all data domains. We find a positive correlation between the convexity in pretrained models and downstream performance in fine-tuned models. **d** Graph convexity (last layer) of pretrained models vs. accuracy of fine-tuned models for several models in each data domain. A positive linear relation can be observed for all domains. Models denoted with */× have the same architecture and size within each modality, showing the relation between convexity and performance is not driven by architectural differences.

We also perform experiments comparing multiple models within the same domain. We focus on different models in the image, text, and audio domains, and one model architecture pretrained on various datasets in the human activity domain. Note here that in order to include more models for the image domain, we use a challenging subset of ImageNet with 10 classes to evaluate and fine-tune the models. Figure 3d demonstrates a strong positive correlation ($r = 0.63$, $p = 5e-4$) between graph convexity in the last layer and downstream accuracy. It is important to note that the convexity scores are not directly comparable across domains but only within each domain due to different amounts of classes. Therefore, we report correlations separately for each domain (see Fig. 3d). The positive correlation between convexity and performance is particularly notable in the text, human activity and medical image domain. In the audio domain, the relationship is ambiguous since all models achieve very high accuracy. Generally, we observe that high convexity in pretrained models can be related to good performance in fine-tuned models and suggest that convexity can be used as an indicator for model choice. Ansuini et al. showed that the so-called intrinsic dimension of the last layer in a pretrained model, also predicts downstream performance down[58]. In Supp. Mat. 2.2 we compare the predictions by convexity and different versions of the intrinsic dimension and we find stronger association for convexity.

Discussion

Understanding machine and human representational spaces is important for human-machine alignment and trust. Such investigations can be furthered by aligning the mechanisms and vocabularies used to describe human and machine representations. Inspired by the conceptual space research of Gärdenfors and co-workers, we introduce the idea of convexity as a relevant measure for machine-learned representations. Convexity is closely related to generalization in cognitive systems, based on mechanisms such as generalization by interpolation or by proximity to prototype.

Machine representations are often found to be better described as curved spaces, hence, we recapitulated salient properties of convex sets in flat and curved spaces. In particular, we considered both conventional Euclidean and graph-based convexity. Importantly, we found that convexity is stable to relevant latent space transformations for deep networks, such transformations appear due to randomness in the learning procedures and measures must therefore be invariant to be relevant.

Convexity is a new concept for machine representations and we have carefully developed (and share) workflows for the estimation of approximate convexity based on Euclidean and graph methods for measuring convexity in latent spaces. We carried out extensive experiments in multiple domains including visual object recognition, human activity data, audio, text, and medical imaging. Our experiments included networks trained by self-supervised learning and next fine-tuned on domain-specific labels. First, we found evidence that both types of convexity are pervasive in both self-supervised pretrained and fine-tuned models.

To test the hypothesis based on cognitive systems, that convexity supports few-shot learning, we designed experiments in the five domains to find relations between pretrained convexity and subsequent fine-tuning. On fine-tuning, we indeed found that decision region convexity increases. More importantly, we find evidence that the higher convexity of a class decision region after pretraining is associated with a higher level of recognition of the given class after fine-tuning, i.e., network representations with higher convexity seem to further fine-tuning. This is clearly of scientific and engineering importance. From a scientific point of view, this may lead to a better understanding of the mechanisms furthering learning. From an engineering point of view, we may develop new tools to induce convexity in the self-supervision process and improve subsequent generalization when fine-tuning to a specific application. A recent work[59] successfully applied the convexity score to make informed pruning decisions before fine-tuning leading to improved performance and reduced training and inference times, highlighting the great potential of the proposed convexity scores. Moreover, Dorszewski et al.[60] found a link between the convexity score and human-machine alignment measured as similarity to human similarity judgments in odd-one-out image triplet task[61,62]. This leads back to the original motivation of using convexity as a common property[63] to bridge human and machine representations, potentially leading to better generalization[64,65].

In addition, while the focus of the work is primarily a supervised setting, in Supp. Mat. 2.3, we discuss an unsupervised setting without any known labels. We show that, if we define labels by any clustering method, the convexity score gives additional insights into the evaluation of the clustering.

The interplay between knowledge stored in pretrained representations and fine-tuning has recently been discussed in the context of chatbot factuality. First, Zhou et al.[66] introduced the 'superficial alignment hypothesis', namely that most factual knowledge is obtained during pretraining, while subsequent fine-tuning merely adds superficial 'stylistic' competencies. In[67], this was further detailed and it was shown that conflicting information presented during fine-tuning is harder to learn. Our findings provide a possible mechanism for these observations, as they suggest that important concepts are learned during pretraining in the form of convex decision regions that allow swift generalization by interpolation. Conflicting concepts, on the other hand, will suffer from the absence of the corresponding convex decision regions in the pretrained representations.

While our contribution is mainly empirical and a detailed theoretical model explaining how pretraining induced convexity predicts fine-tuned performance is pending, the mechanisms proposed to explain generalization in human representations[36,37] may also be relevant for machine representations: The very defining property of convexity (see Definition 1 and Definition 2) implies that categorization can be generalized to new test data by interpolation (linear or geodesic). We hope our introduction of these concepts in the human-machine alignment discourse will spark new theoretical work to further clarify these matters.

The proposed method of evaluating convexity works well with tens of concepts. It would be extremely computationally demanding to compute the convexity metric for thousands or millions of concepts at once. We consider optimization of the code as future work.

Methods

We apply the developed convexity workflows to models trained by self-supervision in five data modalities, namely image, audio, text, human activity, and medical image. We extract the latent representations of the respective datasets after each layer (or layer block) and perform the convexity analysis on a layer- and class-wise basis. To perform the convexity analysis on the pretrained models, we train a softmax linear layer to obtain the predicted model labels. For the analysis of the fine-tuned models, each pretrained model is fine-tuned to their specific classification task by adding a linear layer and fine-tuning the whole model. We also compare several different models in each data modality w.r.t. convexity and performance. The details of used models and datasets as well as the training process of each model are described in the following sections. An overview of all models, their pretraining and finetuning datasets, and accuracies can be found in Table 1.

For all models, we determine a random baseline convexity score by testing the graph convexity with randomly assigned labels. For the estimation of the convexity scores, we sample a maximum of 5000 paths per class.

Table 1 | Overview of trained and analyzed models

Model	Architecture	Pretraining (Pt) Dataset	Finetuning (Ft) Dataset	Pt Acc.	Ft Acc.
data2vec-base[68]	12 T-blocks	ImageNet-1k	ImageNet-1k	46.98%	83.64%
	768 Emb.		69,70		
data2vec-base[68]	12 T-blocks	ImageNet-1k	Imagewoof[72]	47.5%	90.0%
	768 Emb.				
data2vec-large[68]	24 T-blocks	ImageNet-1k	Imagewoof[72]	77.3%	95.4%
	1024 Emb.				
ViT-base[73]	12 T-blocks	ImageNet-21k	Imagewoof[72]	96.2%	95.1%
	768 Emb.				
ViT-large[73]	24 T-blocks	ImageNet-21k	Imagewoof[72]	96.9%	94.8%
	1024 Emb.				
BEiT-base[74]	12 T-blocks	ImageNet-21k	Imagewoof[72]	53.1%	86.9%
	768 Emb.				
BEiT-large[74]	24 T-blocks	ImageNet-21k	Imagewoof[72]	54.4%	91.4%
	1024 Emb.				
DINOv2-small[75]	12 T-blocks	ImageNet-21k	Imagewoof[72]	95.2%	84.5%
	384 Emb.				
DINOv2-base[75]	12 T-blocks	ImageNet-21k	Imagewoof[72]	96.2%	91.9%
	768 Emb.				
DINOv2-large[75]	24 T-blocks	ImageNet-21k	Imagewoof[72]	96.5%	91.4%
	1024 Emb.				
DINOv2-giant[75]	40 T-blocks	ImageNet-21k	Imagewoof[72]	96.4%	92.6%
	1536 Emb.				
Human activity 1[76]	21 1D-Conv.	UK Biobank	Capture-24[77]	72.2%	77.0%
	5 blocks	100.000 part.			
Human activity 2[76]	21 1D-Conv.	UK Biobank		69.3%	76.7%
	5 blocks	1.000 part.			
Human activity 3[76]	21 1D-Conv.	Rowlands		58.5%	73.7%
	5 blocks	55 part.			
wavLM-base[80]	12 T-blocks	LibriLight,	speech commands	94.37%	97.7%
	768 Emb.	VoxPopuli &	v0.02[81]		
		GigaSpeech			
wavLM-large[80]	24 T-blocks	LibriLight,		93.00%	98.45%
	1024 Emb.	VoxPopuli &			
		GigaSpeech			
HUBERT-base[83]	12 T-blocks	LibriSpeech		92.3%	97.5%
	768 Emb.				
HuBERT-large[83]	24 T-blocks	LibriLight		55.2%	98.0%
	1024 Emb.				
wav2vec2-base[82]	12 T-blocks	LibriSpeech		27.7%	97.8%
	768 Emb.				
wav2vec2-large[82]	24 T-blocks	LibriSpeech		8.2%	97.7%
	1024 Emb.				
RoBERTa[84]	12 T-blocks	160GB	20 newsgroup[86]	60.0%	68.9%
	768 Emb.	English text			
BERT[85]	12 T-blocks	Wikipedia &		40.9%	67.7%
	768 Emb.	BookCorpus			
distilBERT[87]	6 T-blocks	distilled		60.8%	69.2%
	768 Emb.	from BERT			
squeezeBERT[88]	12 T-blocks	Wikipedia &		33.0%	67.2%
	768 Emb.	BookCorpus			
LUKE[89]	12 T-blocks	Wikipedia		52.7%	67.6%
	768 Emb.				
ViT-base[73]	12 T-blocks	Bloodcell images	Bloodcell images	93.4%	98.5%
	768 Emb.	as in I-JEPA[90]	91		
ViT-small[73]	12 T-blocks	Bloodcell images		96.3%	98.9%
	384 Emb.	as in I-JEPA[90]			

Models and data

Image domain. For the main experiments, we used the data2vec-base[68] architecture. It consists of 12 transformer layers[47] and was trained to produce the same representations for an input image and its masked version. It was pretrained on an unlabeled version of ImageNet-1k. For fine-tuning, a linear layer was added on top of the mean-pooled output of the last layer. Both the pretrained model and the model fine-tuned on ImageNet-1k[69,70] were obtained from Huggingface[71]. Details on pretraining and fine-tuning can be found in the original paper[68]. We extracted the input embedding and the embeddings after each of the 12 transformer blocks of the validation set of ImageNet-1k. For each layer, we averaged the hidden states across patches to get 768-dimensional feature vectors.

To obtain predictions for the pretrained model, we trained a linear layer with a learning rate of 0.1, a polynomial scheduler, and a weight decay of 0.0001 for 1000 epochs with the whole training set of ImageNet-1k. In our experiments comparing multiple models, we instead use Imagewoof, which is a subset of ImageNet[72]. Imagewoof is smaller than ImageNet, allowing us to train more models while reducing the number of GPU hours. On this dataset, we use the following models: data2vec base, data2vec large (24 layers, 1024 embedding size); the Vision Transformer (ViT)[73] and BEiT[74] in their base versions (12 layers, 768 embedding size) and large versions (24 layers, 1024 embedding size); DINOv2[75] in the small (12 layers, 368 embedding size), base (12 layers, 768 embedding size) and large (24 layers, 1024 embedding size) version. All models are pre-trained on variants of ImageNet, but they differ in the exact pretraining strategies, which can be found in the corresponding publications. All models were obtained from Huggingface. To get the embeddings for ViT, we take the vector corresponding to the classifying token.

Human activity domain. We used a pretrained human activity model from[76] to extract the latent representations. The model follows the architecture of ResNet-V2 with 1D convolutions and a total of 21 convolutional layers. It was pretrained on a large unlabeled dataset from the UK Biobank, which contains 700,000 person-days of free-living tri-axial accelerometer data. The pretraining procedure is a multi-task self-supervised learning schedule, where the model is trained to predict whether several augmentations have been applied or not. The exact details can be found in[76].

We use the Capture-24 dataset[77] for analysis. The Capture-24 dataset contains free-living wrist-worn activity tracking data from 152 participants. We use the four labels introduced in[78], namely; sleeping, sedentary behavior, light physical activity behaviors, and moderate-to-vigorous physical activity behaviors. These 4 labels are used as classes when fine-tuning the model. We randomly selected 30 subjects to hold out for testing and convexity analysis.

We added a single linear layer with softmax activation to obtain the decision regions for both the pretrained and the fine-tuned model. We fine-tune the entire network (encoder and classifier) jointly with a learning rate of 10^{-4}. Following the authors in[76], we selected a small validation set and used early stopping with a patience of 5 epochs. The same procedure was used to obtain the decision regions for the pretrained model, however, the weights of the encoder were frozen. The baseline convexity scores are 25.05 and 25.07 for the pretrained and fine-tuned models, respectively.

We compare the original model to two models of the same architecture but trained differently. The original model is pretrained using three different augmentations, namely arrow of time, permutation, and time warp[76] on data from 100,000 different participants from the UK Biobank dataset. The first variant is pretrained using the same augmentations as the full model, but on a smaller dataset, Rowlands[79], consisting of 55 participants. The second version is pretrained without the arrow of time augmentation and on data from only 1,000 participants from the UK Biobank dataset.

Audio domain. In the audio domain, we used the wavLM[80], pretrained on 94k hours of speech (consisting of LibriLight, VoxPopuli, and GigaSpeech dataset). The exact pretraining objectives can be found in[80]. The model consists of a CNN-based feature encoder and a transformer-based context network. We were especially interested in the latent space representation in the initial embedding layer and the 12 transformer layers. After each layer, we extracted the feature vector of dimension 768.

We use the speech commands v0.02 dataset for analysis[81]. All audio files are sampled to 16000Hz and zero-padded to a length of 16000 if necessary. The dataset consists of 35 classes (excluding the _silence_ class). We use the validation set (9982 files) for the convexity analysis and the train set (84848 files) for training the fine-tuned model. For fine-tuning, an average pooling layer and a linear layer were added to the network to perform a classification task.

To obtain the predictions of the pretrained model, only a final linear layer was added and trained with a frozen model. This was done with a learning rate of 10^{-3} and early stopping. The final model reached an accuracy of 93.00%. For fine-tuning of the full model, only the initial CNN layers were frozen, while the transformer layers and the added linear layer were fine-tuned. The model was fine-tuned with early stopping for 1000 steps (batch size 16), and the learning rate was set to 10^{-4}. The baseline convexity score is 3.58 and 2.87 for the pretrained and fine-tuned models, respectively.

We compare wavLM to its predecessors, the wav2vec2 model[82] and the HuBERT model[83]. For all models, we compare the base version (12 layers, 768 embedding size) and the large version (24 layers, 1024 embedding size). All models are trained in the same way as described above. The exact pretraining strategies can be found in respective publications.

Text domain. For text, we used the base version of RoBERTa[84] which is pretrained to perform Masked Language Modelling[85] to reconstruct masked pieces of text. The model consists of an embedding layer followed by 12 transformer encoder layers and a classification head. The pretraining of RoBERTa is performed on 160GB of uncompressed English-language text in order to learn latent representations of text which are expressed as 768-dimensional vectors.

Analysis was done on the 20 newsgroups dataset[86], which consists of around 18,000 newsgroup posts, covering 20 different topics. Recommended preprocessing steps were performed to remove headers, signature blocks, and quotations from each news article, as the model would otherwise be likely to overfit to features generated from those fragments. The data was split into a training, validation, and test set with 10,000, 1,000, and 7,000 posts, respectively. The data and exact splits that were used are available on Hugging Face (https://huggingface.co/datasets/rasgaard/20_newsgroups). The validation set was used to perform early stopping during training.

We use a single linear layer to perform classification. For predictions of the pretrained model, only the final linear layer was trained for 8 epochs (dictated by early stopping) with a learning rate of 10^{-2}. For complete fine-tuning, the entire model was trained for 3 epochs (dictated by early stopping) with a learning rate of 10^{-4} and a weight decay of 10^{-2}. The baseline convexity score is 5.13 and 5.12 for the pretrained and fine-tuned models, respectively.

We compared the results from RoBERTa to BERT[85], distilBERT[87], squeezeBERT[88], and LUKE[89] in order to have a variety of comparable models of varying sizes. Results from these models were obtained by following the same procedure as with RoBERTa.

Medical imaging domain. We fully pretrained a small and base ViT with the procedure proposed in I-JEPA[90]. We pretrained the models on digital images of normal peripheral blood cells[91]. The dataset contains 17,092 images of eight normal blood cell types: neutrophils, eosinophils, basophils, lymphocytes, monocytes, immature granulocytes,

erythroblasts, and platelets. We used 80% of the available data for pretraining and fine-tuning, 10% for validation and 10% for the convexity analysis. For both model versions, we used a patch size of 16 and 12 transformer blocks. Small and base versions have a feature size of 384 and 768 respectively. I-JEPA is a masked image model, where the pretext task is to predict embeddings of hold-out image context from the embeddings of the encoder. The model was trained using a smooth L1 loss with default settings in PyTorch. Hyperparameters were kept consistent with the original work of[90]. We used the AdamW optimizer, a learning rate of 0.001 with cosine decay and a linear warm-up of 40 epochs and total of 300 epochs.

After pretraining, we froze the encoder model weights and trained a linear layer for 50 epochs with the same optimizer and a learning rate of 0.005 on the same dataset. The baseline convexity score is 12.74 and 12.09 for the pretrained and fine-tuned base models, respectively.

We used the 1709 hold-out samples for the convexity analysis. To retrieve a single embedding vector, we averaged over tokens, resulting in a 384 and 768-dimensional vector per sample and layer for the small and base version, respectively. We extracted these vectors after the convolutional patch embedding layer, after transformer blocks 2, 4, 6, 8, and 10, and after normalization of the 12th transformer block, resulting in seven feature vectors per sample.

Data availability

All datasets used are openly available under the corresponding links detailed in the readme file at https://github.com/LenkaTetkova/Convexity-of-representations.git.

Code availability

The code is available at https://github.com/LenkaTetkova/Convexity-of-representations.git.

References

1. Bender, E. M., Gebru, T., McMillan-Major, A. & Shmitchell, S. On the dangers of stochastic parrots: Can language models be too big? In *Proceedings of the 2021 ACM conference on fairness, accountability, and transparency*, 610–623 (2021).
2. Mahowald, K. et al. Dissociating language and thought in large language models. *Trends Cogn. Sci.* **28**, 517–540 (2024).
3. Kriegeskorte, N., Mur, M. & Bandettini, P. A. Representational similarity analysis-connecting the branches of systems neuroscience. *Front. Syst. Neurosci.* **2**, 249 (2008).
4. Samek, W., Montavon, G., Vedaldi, A., Hansen, L. K. & Müller, K.-R.*Explainable AI: Interpreting, Explaining and Visualizing Deep Learning*, vol. 11700 (Springer Nature, 2019).
5. Christian, B.*The alignment problem: Machine learning and human values* (WW Norton & Company, 2020).
6. Balkenius, C. & Gärdenfors, P. Spaces in the brain: From neurons to meanings. *Front. Psychol.* **7**, 1820 (2016).
7. Tang, J., LeBel, A., Jain, S. & Huth, A. G. Semantic reconstruction of continuous language from non-invasive brain recordings. *Nat. Neurosci.* **26**, 858–866 (2023).
8. Gärdenfors, P.*The Geometry of Meaning: Semantics Based on Conceptual Spaces* (MIT press, 2014).
9. Shepard, R. N. Toward a universal law of generalization for psychological science. *Science* **237**, 1317–1323 (1987).
10. Chung, S. & Abbott, L. Neural population geometry: An approach for understanding biological and artificial neural networks. *Curr. Opin. Neurobiol.* **70**, 137–144 (2021).
11. Goldstein, A. et al. Shared computational principles for language processing in humans and deep language models. *Nat. Neurosci.* **25**, 369–380 (2022).
12. Valeriani, L. et al. The geometry of hidden representations of large transformer models. *Adv. Neural Inf. Process. Syst.*, **36**, 51234–51252 (2023).
13. Mamou, J. et al. Emergence of separable manifolds in deep language representations. In *Proceedings of the 37th International Conference on Machine Learning*, 6713–6723 (2020).
14. Hénaff, O. J. & Simoncelli, E. P. Geodesics of learned representations. In *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*. (eds. Bengio, Y. & LeCun, Y.) (2016).
15. Arvanitidis, G., Hansen, L. K. & Hauberg, S. Latent space oddity: on the curvature of deep generative models. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings* https://openreview.net/forum?id=SJzRZ-WCZ (OpenReview.net, 2018).
16. Brahma, P. P., Wu, D. & She, Y. Why deep learning works: A manifold disentanglement perspective. *IEEE Trans. neural Netw. Learn. Syst.* **27**, 1997–2008 (2015).
17. Hénaff, O. J., Goris, R. L. & Simoncelli, E. P. Perceptual straightening of natural videos. *Nat. Neurosci.* **22**, 984–991 (2019).
18. Tenenbaum, J. B., Silva, V. D. & Langford, J. C. A global geometric framework for nonlinear dimensionality reduction. *science* **290**, 2319–2323 (2000).
19. Mika, S. et al. Kernel pca and de-noising in feature spaces. *Adv. Neural Inform. Process. Syst.* **11** (1998).
20. Arvanitidis, G., Hansen, L. K. & Hauberg, S. Latent space oddity: On the curvature of deep generative models. In *6th International Conference on Learning Representations, ICLR 2018* (2018).
21. Oring, A., Yakhini, Z. & Hel-Or, Y. Autoencoder image interpolation by shaping the latent space. In Meila, M. & Zhang, T. (eds.) *Proceedings of the 38th International Conference on Machine Learning*, vol. 139 of *Proceedings of Machine Learning Research*, 8281–8290 https://proceedings.mlr.press/v139/oring21a.html (PMLR, 2021).
22. Struski, L., Sadowski, M., Danel, T., Tabor, J. & Podolak, I. T. Feature-based interpolation and geodesics in the latent spaces of generative models. *IEEE Trans. Neural Netw. Learn. Syst.* **35**, 12068–12082 (2024).
23. Sainburg, T., Thielk, M., Theilman, B., Migliori, B. & Gentner, T. Generative adversarial interpolative autoencoding: adversarial training on latent space interpolations encourage convex latent distributions https://arxiv.org/abs/1807.06650 (2019).
24. Schultz, K. et al. Convgen: A convex space learning approach for deep-generative oversampling and imbalanced classification of small tabular datasets. *Pattern Recognit.* **147**, 110138 (2024).
25. Chawla, N. V., Bowyer, K. W., Hall, L. O. & Kegelmeyer, W. P. Smote: synthetic minority over-sampling technique. *J. Artif. Intell. Res.* **16**, 321–357 (2002).
26. Mrabah, N., Amar, M. M., Bouguessa, M. & Diallo, A. B. Toward convex manifolds: A geometric perspective for deep graph clustering of single-cell rna-seq data. In Elkind, E. (ed.) *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence, IJCAI-23*, 4855–4863 (International Joint Conferences on Artificial Intelligence Organization, 2023). https://doi.org/10.24963/ijcai.2023/540.
27. Hanin, B. & Rolnick, D. Deep relu networks have surprisingly few activation patterns. *Advances in neural information processing systems* **32** (2019).
28. Black, S. et al. Interpreting neural networks through the polytope lens. *arXiv preprint arXiv:2211.12312* (2022).
29. Elhage, N. et al. Toy models of superposition. *arXiv preprint arXiv:2209.10652* (2022).
30. Gärdenfors, P. Induction, conceptual spaces and ai. *Philos. Sci.* **57**, 78–95 (1990).
31. Warglien, M. & Gärdenfors, P. Semantics, conceptual spaces, and the meeting of minds. *Synthese* **190**, 2165–2193 (2013).

32. Douven, I., Elqayam, S., Gärdenfors, P. & Mirabile, P. Conceptual spaces and the strength of similarity-based arguments. *Cognition* **218**, 104951 (2022).
33. Strößner, C. Criteria for naturalness in conceptual spaces. *Synthese* **200**, 78 (2022).
34. Parascandolo, G., Kilbertus, N., Rojas-Carulla, M. & Schölkopf, B. Learning independent causal mechanisms. In *International Conference on Machine Learning*, 4036–4044 (PMLR, 2018).
35. Bechberger, L. & Kühnberger, K.-U. Grounding psychological shape space in convolutional neural networks. In *Software Engineering and Formal Methods. SEFM 2021 Collocated Workshops: CIFMA, CoSim-CPS, OpenCERT, ASYDE, Virtual Event, December 6–10, 2021, Revised Selected Papers*, 86–106 (Springer, 2022).
36. Gärdenfors, P. Concept learning: a geometrical model. In *Proceedings of the Aristotelian Society (Hardback)*, vol. 101, 163–183 (Wiley Online Library, 2001).
37. Gärdenfors, P., Jost, J. & Warglien, M. From actions to effects: Three constraints on event mappings. *Front. Psychol.* **9**, 1391 (2018).
38. Gärdenfors, P. & Williams, M.-A. Reasoning about categories in conceptual spaces. In *IJCAI*, 385–392 (2001).
39. Boyd, S. P. & Vandenberghe, L.*Convex Optimization* (Cambridge University Press, 2004). https://web.stanford.edu/%7Eboyd/cvxbook/.
40. Marc, T. & Šubelj, L. Convexity in complex networks. *Netw. Sci.* **6**, 176–203 (2018).
41. Dijkstra, E. W. A note on two problems in connexion with graphs. *Numerische Mathematik* **1**, 269–271 (1959).
42. Fredman, M. L. & Tarjan, R. E. Fibonacci heaps and their uses in improved network optimization algorithms. *J. ACM* **34**, 596–615 (1987).
43. Cormen, T. H., Leiserson, C. E., Rivest, R. L. & Stein, C.*Introduction to Algorithms, 4th edition* (MIT press, 2022).
44. Floyd, R. W. Algorithm 97: Shortest path. *Commun. ACM* **5**, 345 https://doi.org/10.1145/367766.368168 (1962).
45. Bernstein, M., De Silva, V., Langford, J. C. & Tenenbaum, J. B. Graph approximations to geodesics on embedded manifolds. Tech. Rep., Citeseer (2000).
46. Davis, E. & Sethuraman, S. Approximating geodesics via random points. *Ann. Appl. Probab.* **29**, 1446–1486 (2019).
47. Vaswani, A. et al. Attention is all you need. *Advances in neural information processing systems* **30** (2017).
48. Bishop, C. M. et al. *Neural networks for pattern recognition* (Oxford university press, 1995).
49. Montufar, G. F., Pascanu, R., Cho, K. & Bengio, Y. On the number of linear regions of deep neural networks. In *Advances in Neural Information Processing Systems*, vol. 27 https://proceedings.neurips.cc/paper_files/paper/2014/hash/109d2dd3608f669ca17920c511c2a41e-Abstract.html (Curran Associates, Inc., 2014).
50. Hanin, B. & Rolnick, D. Complexity of linear regions in deep networks. In *Proceedings of the 36th International Conference on Machine Learning*, 2596-2604 (PMLR, 2019). https://proceedings.mlr.press/v97/hanin19a.html.
51. Goujon, A., Etemadi, A. & Unser, M. The role of depth, width, and activation complexity in the number of linear regions of neural networks http://arxiv.org/abs/2206.08615. [cs, math, stat] (2022).
52. Fan, F.-L. et al. Deep relu networks have surprisingly simple polytopes http://arxiv.org/abs/2305.09145. [cs] (2023).
53. Belinkov, Y., Durrani, N., Dalvi, F., Sajjad, H. & Glass, J. What do neural machine translation models learn about morphology? In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 861–872 (2017).
54. Hewitt, J. & Liang, P. Designing and interpreting probes with control tasks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, 2733–2743 (2019).
55. Kim, B. et al. Interpretability beyond feature attribution: Quantitative testing with concept activation vectors (tcav). In *International conference on machine learning*, 2668–2677 (PMLR, 2018).
56. Radovanović, M., Nanopoulos, A. & Ivanović, M. Hubs in space: Popular nearest neighbors in high-dimensional data. *J. Mach. Learn. Res.* **11**, 2487–2531 (2010).
57. Hinton, G. E. & Roweis, S. Stochastic neighbor embedding. *Advances in neural information processing systems* **15** (2002).
58. Ansuini, A., Laio, A., Macke, J. H. & Zoccolan, D. Intrinsic dimension of data representations in deep neural networks. In *Advances in Neural Information Processing Systems*, vol. 32 (Curran Associates, Inc., 2019). https://proceedings.neurips.cc/paper/2019/hash/cfcce0621b49c983991ead4c3d4d3b6b-Abstract.html.
59. Dorszewski, T., Tetkova, L. & Hansen, L. K. Convexity based pruning of speech representation models. In *2024 IEEE 34th International Workshop on Machine Learning for Signal Processing (MLSP)*, 1–6 (IEEE, 2024).
60. Dorszewski, T., Tětková, L., Linhardt, L. & Hansen, L. K. Connecting concept convexity and human-machine alignment in deep neural networks. In *Northern Lights Deep Learning Conference* (PMLR, 2025).
61. Muttenthaler, L., Dippel, J., Linhardt, L., Vandermeulen, R. A. & Kornblith, S. Human alignment of neural network representations. In *The Eleventh International Conference on Learning Representations* (2023).
62. Muttenthaler, L. et al. Improving neural network representations using human similarity judgments. *Adv. Neural Inf. Process. Syst.* **36**, 50978–51007 (2024).
63. Gärdenfors, P. Natural concepts and the economics of cognition and communication. *Philosophia* 1–18 (2024).
64. Sucholutsky, I. & Griffiths, T. Alignment with human representations supports robust few-shot learning. *Adv. Neural Inf. Process. Syst.* **36**, 73464–73479 (2023).
65. Sucholutsky, I. et al. Representational alignment supports effective machine teaching. *arXiv preprint arXiv:2406.04302* (2024).
66. Zhou, C. et al. Lima: Less is more for alignment. *Adv. Neural Inf. Process. Syst.* **36**, 55006–55021 (2024).
67. Gekhman, Z. et al. Does Fine-Tuning LLMs on New Knowledge Encourage Hallucinations?. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, 7765–7784 (Association for Computational Linguistics, 2024).
68. Baevski, A. et al. Data2vec: A general framework for self-supervised learning in speech, vision and language. In *International Conference on Machine Learning*, 1298–1312 (PMLR, 2022).
69. Russakovsky, O. et al. ImageNet Large Scale Visual Recognition Challenge. *Int. J. Computer Vis. (IJCV)* **115**, 211–252 (2015).
70. Deng, J. et al. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09* (2009).
71. Wolf, T. et al. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, 38–45 (Association for Computational Linguistics, Online, 2020). https://www.aclweb.org/anthology/2020.emnlp-demos.6.
72. Howard, J. Imagewoof: a subset of 10 classes from imagenet that aren't so easy to classify https://github.com/fastai/imagenette#imagewoof (2019).
73. Dosovitskiy, A. et al. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations* (2021).

74. Bao, H., Dong, L., Piao, S. & Wei, F. Beit: Bert pre-training of image transformers. In *International Conference on Learning Representations* (2022).
75. Oquab, M. et al. Dinov2: Learning robust visual features without supervision. In *Transactions on Machine Learning Research*, 2835–8856 (2024).
76. Yuan, H. et al. Self-supervised learning for human activity recognition using 700,000 person-days of wearable data 2206.02909 (2022).
77. Willetts, M., Hollowell, S., Aslett, L., Holmes, C. & Doherty, A. Statistical machine learning of sleep and physical activity phenotypes from sensor data in 96,220 UK biobank participants. *Sci. Rep.* **8**, 7961 (2018).
78. Walmsley, R. et al. Reallocation of time between device-measured movement behaviours and risk of incident cardiovascular disease. *Br. J. Sports Med.* **56**, 1008–1017 (2022).
79. Esliger, D. W. et al. Validation of the genea accelerometer. *Med. Sci. Sports Exerc.* **43**, 1085–1093 (2011).
80. Chen, S. et al. Wavlm: Large-scale self-supervised pre-training for full stack speech processing. *IEEE J. Sel. Top. Signal Process.* **16**, 1505–1518 (2022).
81. Warden, P. Speech Commands: A Dataset for Limited-Vocabulary Speech Recognition. *ArXiv e-prints* https://arxiv.org/abs/1804.03209. (2018).
82. Baevski, A., Zhou, Y., Mohamed, A. & Auli, M. wav2vec 2.0: A framework for self-supervised learning of speech representations. *Adv. neural Inf. Process. Syst.* **33**, 12449–12460 (2020).
83. Hsu, W.-N. et al. Hubert: Self-supervised speech representation learning by masked prediction of hidden units. *IEEE/ACM Trans. audio, speech, Lang. Process.* **29**, 3451–3460 (2021).
84. Liu, Y. et al. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692* (2019).
85. Devlin, J., Chang, M.-W., Lee, K. & Toutanova, K. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* (2018).
86. Lang, K. Newsweeder: Learning to filter netnews. In *Machine learning proceedings 1995*, 331–339 (Elsevier, 1995).
87. Sanh, V., Debut, L., Chaumond, J. & Wolf, T. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108* (2019).
88. Iandola, F., Shaw, A., Krishna, R., & Keutzer, K. SqueezeBERT: What can computer vision teach NLP about efficient neural networks?. In *Proceedings of SustaiNLP: Workshop on Simple and Efficient Natural Language Processing*, 124–135 (2020).
89. Yamada, I., Asai, A., Shindo, H., Takeda, H., & Matsumoto, Y. LUKE: Deep Contextualized Entity Representations with Entity-aware Self-attention. *In Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 6442–6454 (2020).
90. Assran, M. et al. Self-supervised learning from images with a joint-embedding predictive architecture. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 15619–15629 (2023).
91. Acevedo, A. et al. A dataset of microscopic peripheral blood cell images for development of automatic recognition systems (2020).

Acknowledgements

This work was supported by the the Novo Nordisk Foundation grant NNF22OC0076907 "Cognitive spaces - Next generation explainability" and the Pioneer Centre for AI, DNRF grant number P1. This work was supported by the DIREC Bridge project Deep Learning and Automation of Imaging- Based Quality of Seeds and Grains, Innovation Fund Denmark grant number 9142-00001B. This work was supported by the Danish Data Science Academy, which is funded by the Novo Nordisk Foundation (NNF21SA0069429) and VILLUM FONDEN (40516). This work was partially supported by DeiC National HPC (g.a. DeiC-DTU-N5-20230028) and by the "Alignment of human and machine representations" project (g.a. DeiC-DTU-N5-20230033) and by the "self-supervised brain model" project (g.a. DeiC-DTU-S5-202300105). We acknowledge Danish e-infrastructure Cooperation (DeiC), Denmark, for awarding this project access to the LUMI supercomputer, owned by the EuroHPC Joint Undertaking, hosted by CSC (Finland) and the LUMI consortium through Danish e-infrastructure Cooperation (DeiC), Denmark, "Alignment of human and machine representations", DeiC-DTU-N5-20230028.

Author contributions

L. K. H. conceived the presented idea. L. T., T. B., T. D., F. M., R. A., T. S. A. and L. K. H. designed the experiments. L. T., T. B., T. D., F. M., and R. A. carried out the experiments. L. T., J. F., T. S. A., and L. K. H. developed the theoretical formalism. L. T., T. B., T. D., F. M., R. A., and T. S. A. prepared the figures. T. S. A. and L. K. H. supervised the project. All authors participated in discussions on the content and design of the experiments, contributed to the interpretation of the results, and wrote and reviewed the manuscript.

Competing interests

The authors declare no competing interests.

Additional information

Supplementary information The online version contains supplementary material available at https://doi.org/10.1038/s41467-025-60809-y.

Correspondence and requests for materials should be addressed to Lenka Tětková.

Peer review information *Nature Communications* thanks Corina Stroessner, and the other, anonymous, reviewer(s) for their contribution to the peer review of this work. A peer review file is available.

Supplementary Materials – On convex decision regions in deep network representations

1 Theory

In this appendix, we present some mathematical background and intuition for the approximate convexity workflow.

1.1 Euclidean convexity is invariant to affine transformations

Theorem 1. Let $f : \mathbb{R}^n \to \mathbb{R}^n$ be an affine transformation (i.e., $f(\mathbf{x}) = \boldsymbol{A}\mathbf{x} + \mathbf{b}$, where $\mathbf{b} \in \mathbb{R}^n$ and $\boldsymbol{A} : \mathbb{R}^n \to \mathbb{R}^n$ is an invertible linear transformation). Let $X \subset \mathbb{R}^n$ be a convex set. Then $f(X)$ is convex [1].

Proof. Let $\mathbf{x}, \mathbf{y} \in f(X)$ and $\lambda \in [0, 1]$. There exist $\overline{\mathbf{x}}, \overline{\mathbf{y}} \in X$ such that $f(\overline{\mathbf{x}}) = \mathbf{x}$ and $f(\overline{\mathbf{y}}) = \mathbf{y}$. Since X is convex, we have $\lambda\overline{\mathbf{x}} + (1 - \lambda)\overline{\mathbf{y}} \in X$. Therefore,

$$f\left(\lambda\overline{\mathbf{x}} + (1 - \lambda)\overline{\mathbf{y}}\right) \in f(X). \tag{1}$$

Moreover, because of the linearity of A,

$$f\left(\lambda\overline{\mathbf{x}} + (1 - \lambda)\overline{\mathbf{y}}\right) = \boldsymbol{A}\left(\lambda\overline{\mathbf{x}} + (1 - \lambda)\overline{\mathbf{y}}\right) + \mathbf{b} = \lambda\boldsymbol{A}\overline{\mathbf{x}} + (1 - \lambda)\boldsymbol{A}\overline{\mathbf{y}} + \mathbf{b}. \tag{2}$$

Finally, we have

$$\begin{aligned} \lambda\boldsymbol{A}\overline{\mathbf{x}} + (1 - \lambda)\boldsymbol{A}\overline{\mathbf{y}} + \mathbf{b} &= \lambda\left(\boldsymbol{A}\overline{\mathbf{x}} + \mathbf{b}\right) + (1 - \lambda)\left(\boldsymbol{A}\overline{\mathbf{y}} + \mathbf{b}\right) && (3)\\ &= \lambda f(\overline{\mathbf{x}}) + (1 - \lambda)f(\overline{\mathbf{y}}) && (4)\\ &= \lambda\mathbf{x} + (1 - \lambda)\mathbf{y}. && (5) \end{aligned}$$

Hence,

$$f\left(\lambda\overline{\mathbf{x}} + (1 - \lambda)\overline{\mathbf{y}}\right) = \lambda\mathbf{x} + (1 - \lambda)\mathbf{y} \tag{6}$$

and $\lambda\mathbf{x} + (1 - \lambda)\mathbf{y} \in f(X)$. Therefore, $f(X)$ is convex. □

1.2 Graph convexity is invariant to isometry and uniform scaling

Theorem 2. Graph convexity is invariant to isometry and uniform scaling.

Proof. Isometry (with respect to Euclidean distance) is a map $I : \mathbb{R}^n \to \mathbb{R}^n$ such that

$$\forall \mathbf{x}, \mathbf{y} \in \mathbb{R}^n : \|\mathbf{x} - \mathbf{y}\|_2 = \|I(\mathbf{x}) - I(\mathbf{y})\|_2. \tag{7}$$

Since isometry preserves distances, it induces the same distance graph. Therefore, graph convexity is invariant to isometries.

Universal scaling is a map $S : \mathbb{R}^n \to \mathbb{R}^n$ such that

$$\exists C \in \mathbb{R} \;\; \forall \mathbf{x} \in \mathbb{R}^n : \;\; S(\mathbf{x}) = C\mathbf{x}. \tag{8}$$

It holds that

$$\|S(\mathbf{x}) - S(\mathbf{y})\|_2 = \|C\mathbf{x} - C\mathbf{y}\|_2 = |C| \cdot \|\mathbf{x} - \mathbf{y}\|_2. \tag{9}$$

All the distances are scaled by $|C|$. It follows that the chosen nearest neighbours are the same. We show by contradiction that all the shortest paths are also the same. We denote G the graph with the original edges and G_S the graph with the scaled edges.

Given $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$, suppose that there exists a shortest path:

$$P = (\mathbf{p_0} = \mathbf{x}, \mathbf{p_1}, \ldots, \mathbf{p_j} = \mathbf{y}) \tag{10}$$

from $\mathbf{x}$ to $\mathbf{y}$, and a different path:

$$Q = (\mathbf{q_0} = S(\mathbf{x}), \mathbf{q_1}, \ldots, \mathbf{q_k} = S(\mathbf{y})) \tag{11}$$

from $S(\mathbf{x})$ to $S(\mathbf{y})$ such that

$$L(Q) < L(S(P)), \tag{12}$$

where

$$S(P) = (S(\mathbf{p_0}), S(\mathbf{p_1}), \ldots, S(\mathbf{p_j})) \tag{13}$$

and $L(\cdot)$ is an operator measuring the length of a path. From equation 12, it follows that $C \neq 0$. Because all the edges are scaled by a factor $|C|$, it holds that $L(S(P)) = |C|L(P)$. Moreover, for every pair of nodes $\mathbf{v_1}$, $\mathbf{v_2}$, there exists an edge in G from $\mathbf{v_1}$ to $\mathbf{v_2}$ if and only if there exists an edge in G_S from $S(\mathbf{v_1})$ to $S(\mathbf{v_2})$. Hence, there exists a path R from $\mathbf{x}$ to $\mathbf{y}$ such that $Q = S(R)$. It follows that

$$|C|L(R) = L(S(R)) < L(S(P)) = |C|L(P). \tag{14}$$

Therefore, $L(R) < L(P)$ and that contradicts the assumption that P is the shortest path from $\mathbf{x}$ to $\mathbf{y}$. □

1.3 Softmax induces convexity

Theorem 3. The preimage of each decision region under the last dense layer and softmax function is a convex set. More precisely: Let us denote the output of the last dense layer by

$$a_k(\mathbf{z}) = \sum_{j=1}^{J} w_{k,j} z_j \tag{15}$$

for $\mathbf{z} \in \mathbb{R}^n$ and the probabilities

$$p_k(\mathbf{z}) = \frac{\exp a_k(\mathbf{z})}{\sum_{k'=1}^{K} \exp a_{k'}(\mathbf{z})}. \tag{16}$$

Denote

$$C_k \subseteq \mathbb{R}^n = \{\mathbf{x} \in \mathbb{R}^n : p_k(\mathbf{x}) > p_j(\mathbf{x}) \; \forall j \in \{1, \ldots, K\}, j \neq k\}. \tag{17}$$

Then C_k is a convex set for any $k \in \{1, \ldots, K\}$ [2].

Proof. Let us define regions

$$\mathbf{x} \in \mathcal{R}_k \iff (a_k(\mathbf{x}) > a_j(\mathbf{x}) \;\; \forall j \neq k). \tag{18}$$

Because $\sum_{k'=1}^{K} \exp a_{k'} > 0$ and by the monotonicity of the exponential function, it holds that

$$k_{opt}(\mathbf{x}) := \arg\max_k p_k(\mathbf{x}) = \arg\max_k \frac{\exp a_k(\mathbf{x})}{\sum_{k'=1}^{K} \exp a_{k'}(\mathbf{x})} = \arg\max_k a_k(\mathbf{x}), \tag{19}$$

Therefore,

$$\mathcal{R}_k = \mathcal{C}_k \;\; \forall k \in \{1, \ldots K\} \tag{20}$$

and we can from now on work with $\mathcal{R}_k$.

Following the definitions from [2], pages 182-184:
Let $\mathbf{x}_A, \mathbf{x}_B \in \mathcal{R}_k$ and define any point $\hat{\mathbf{x}}$ that lies on the line connecting the two:

$$\hat{\mathbf{x}} = \lambda \mathbf{x}_A + (1 - \lambda)\mathbf{x}_B, \tag{21}$$

where $0 \leq \lambda \leq 1$. Cf. equation 15, the discriminant functions, a_k, are linear and it follows that:

$$a_k(\hat{\mathbf{x}}) = \lambda a_k(\mathbf{x}_A) + (1 - \lambda) a_k(\mathbf{x}_B) \tag{22}$$

for all k. From the definition of $\mathcal{R}_k$, we know that $a_k(\mathbf{x}_A) > a_j(\mathbf{x}_A)$ and $a_k(\mathbf{x}_B) > a_j(\mathbf{x}_B)$ for all $j \neq k$. Therefore, $a_k(\hat{\mathbf{x}}) > a_j(\hat{\mathbf{x}})$ for all $j \neq k$ and $\hat{\mathbf{x}}$ belongs to $\mathcal{R}_k$. Hence, $\mathcal{R}_k = \mathcal{C}_k$ is convex. □

1.4 Runtime complexity

We have a dataset $\mathcal{D}$ with C classes each with N_c data points, N total data points, and each data point has a representation in $\mathbf{z} \in \mathbb{R}^D$.

Graph Convexity: Computing the graph convexity score requires three steps:

1. Compute the nearest neighbor matrix. This cost $1/2N^2D$.

2. Create the adjacency matrix. This cost $1/2N^2$. The adjacency matrix will at most have $E = NK$ edges, where K is the K chosen for K-nearest neighbors
3. Compute the convexity score for each pair of data points for each class or concept using Dijkstra's algorithm. This cost $1/2CN_c(N_c - 1)(N \log N + E)$.

The total computational cost is then

$$\text{cost} = 1/2N^2D + 1/2N^2 + 1/2CN_c(N_c - 1)\left(N \log N + NK\right) \tag{23}$$

This simplifies to

$$\text{cost} = \mathcal{O}\left(N^2(D+1) + CN_c(N_c - 1)N\left(\log N + K\right)\right) \tag{24}$$

Since $CN_c = N$, then if $(N_c - 1)(\log N + K) \gg (D + 1)$, the dominating term will be the latter. This can be seen by rewriting:

$$\text{cost} = \mathcal{O}\left(N^2(D+1) + N^2(N_c - 1)\left(\log N + K\right)\right) \tag{25}$$

Instead of computing the convexity score for a graph exact, we can estimate it using N_s samples per class/concept, we get

$$\text{cost} = \mathcal{O}\left(N^2(D+1) + CN_s(N_s - 1)N\left(\log N + K\right)\right) \tag{26}$$

Euclidean Convexity: Computing the Euclidean convexity score for each class/concepts entails the following operations

1. Sample N_p point pairs, denoting their representations $\mathbf{z}_1$ and $\mathbf{z}_2$. This operation scales with N_p.
2. Per point pair, generate N_s representations equidistant on a line between $\mathbf{z}_1$ and $\mathbf{z}_2$. This scales with N_s .
3. For each generated representation, compute the classification. This scales with D.

Since all the costs are scaling linearly, the total cost of computing the Euclidean convexity score is

$$\text{cost} = \mathcal{O}\left(CN_sN_pD\right) \tag{27}$$

1.5 Voronoi tessellation

If we classify based on closeness to a set of prototypes, we get convex decision regions. Formally:

Theorem 4. Let $\{\mathbf{p}_i\}_{i=1}^{C} \subset \mathbb{R}^d$ be a finite set of points, *prototypes*, in $\mathbb{R}^d$ and for all $\{1, \ldots, C\}$, we define

$$R_i = \left\{\mathbf{x} \in \mathbb{R}^d : \forall j \in \{1, \ldots, C\} \setminus \{i\} : \|\mathbf{x} - \mathbf{p}_i\|_2 < \|\mathbf{x} - \mathbf{p}_i\|_2\right\}. \tag{28}$$

Then $\forall i \in \{1, \ldots, C\} : R_i$ is Euclidean-convex.

Proof. Let $\mathbf{x}, \mathbf{y} \in R_i$, $j \in \{1, \ldots, C\}, j \neq i$, and $t \in (0, 1)$ and consider a point on the segment between $\mathbf{x}$ and y, $t\mathbf{x} + (1 - t)\mathbf{y}$. Since $\mathbf{x} \in R_i$, the following holds for any $j \in \{1, \ldots, C\} \setminus \{i\}$:

$$\|\mathbf{x}\|_2^2 + \|\mathbf{p}_i\|_2^2 - 2\mathbf{x} \cdot \mathbf{p}_i = \|\mathbf{x} - \mathbf{p}_i\|_2^2 < \|\mathbf{x} - \mathbf{p}_i\|_2^2 = \|\mathbf{x}\|_2^2 + \left\|\mathbf{p}_j\right\|_2^2 - 2\mathbf{x} \cdot \mathbf{p}_j. \tag{29}$$

Hence,

$$\|\mathbf{p}_i\|_2^2 - 2\mathbf{x} \cdot \mathbf{p}_i < \left\|\mathbf{p}_j\right\|_2^2 - 2\mathbf{x} \cdot \mathbf{p}_j. \tag{30}$$

Similarly, the same is true for y. We use these inequalities in the following:

$$\|t\mathbf{x} + (1-t)\mathbf{y} - \mathbf{p}_i\|_2^2 = \|t\mathbf{x} + (1-t)\mathbf{y}\|_2^2 + \|\mathbf{p}_i\|_2^2 - 2(t\mathbf{x} + (1-t)\mathbf{y}) \cdot \mathbf{p}_i \tag{31}$$

$$= \|t\mathbf{x} + (1-t)\mathbf{y}\|_2^2 + +t(\|\mathbf{p}_i\|_2^2 - 2\mathbf{x} \cdot \mathbf{p}_i) + (1-t)(\|\mathbf{p}_i\|_2^2 - 2\mathbf{x} \cdot \mathbf{p}_i) \tag{32}$$

$$< \|t\mathbf{x} + (1-t)\mathbf{y}\|_2^2 + +t(\left\|\mathbf{p}_j\right\|_2^2 - 2\mathbf{x} \cdot \mathbf{p}_j) + (1-t)(\left\|\mathbf{p}_j\right\|_2^2 - 2\mathbf{x} \cdot \mathbf{p}_j) \tag{33}$$

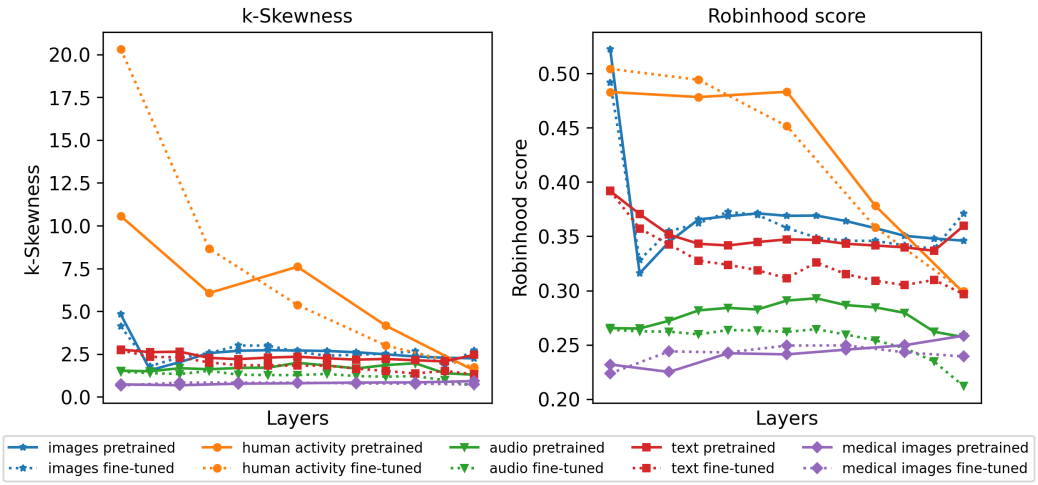
$$= \left\|t\mathbf{x} + (1-t)\mathbf{y} - \mathbf{p}_j\right\|_2^2. \tag{34}$$

Hence, $t\mathbf{x} + (1 - t)\mathbf{y} \in R_i$ and, therefore, R_i is convex. □

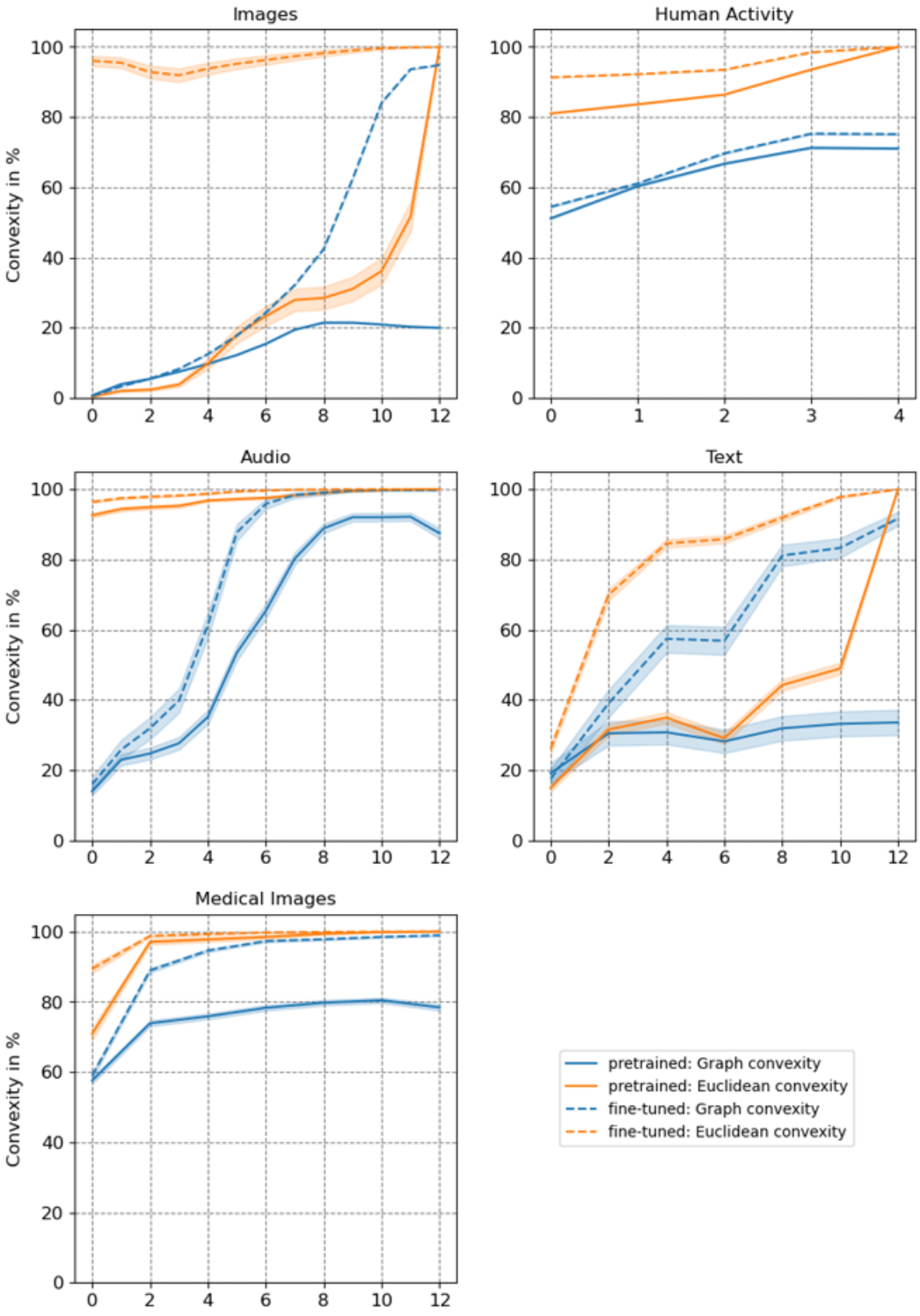
2 Additional Results

2.1 Hubness analysis

Supplementary Figure 1 shows hubness metrics (k-skewness and Robinhood score) for all models and domains. It follows that hubness is not a problem for any of the domains. Supplementary Figure 2 shows the individual convexity results for all domains including the standard error of the mean. We observe an increase in convexity across layers for all models and an increased convexity in fine-tuned models.



Supplementary Figure 1: Hubness evaluation for all modalities. No measures to correct for hubness were taken, as the numbers for k-skewness and Robinhood score are generally low.



Supplementary Figure 2: Individual convexity results for all domains. Error bars show the error of the mean. All results are aggregated in Fig. 3b in the main paper. We observe an increase in convexity across layers for all models and an increased convexity in fine-tuned models.

	graph convexity	accuracy	ID: DANCO	ID: PCA 10	ID: kNN 10
graph convexity					
accuracy	0.265				
ID: DANCO	0.087	0.083			
ID: PCA 10	-0.072	0.015	0.012		
ID: kNN 10	0.042	0.019	-0.007	-0.033	

Supplementary Table 1: Pearson correlations for data2vec: graph convexity in the pretrained model (mean over all layers), accuracy of the fine-tuned model, intrinsic dimension (in the last layer Nature Communications of the pretrained model) estimated using the Dimensionality from Angle and Norm Concentration algorithm (DANCO) [4], the PCA algorithm (10 nearest neighbors) [5], and the kNN algorithm (10 nearest neighbors) [6]. Correlation is computed for 1000 ImageNet classes.

2.2 Comparison to other latent space metrics

To show how similar is the proposed convexity score to existing metrics evaluating latent spaces, we measure Pearson correlations for graph convexity in the pretrained model, accuracy in the fine-tuned model, and three estimators or intrinsic dimension. The intrinsic dimension is the minimal number of parameters needed to accurately describe the class representations learned by the network. Ansuini et al. show that the intrinsic dimension of the last hidden layer predicts classification performance [3].

We report the correlation coefficients in Supplementary Table 1. It shows that the graph convexity scores are distinct from all estimators of intrinsic dimension. Moreover, the stronger relationship is between graph convexity and accuracy, i.e. the relation this paper focuses on.

2.3 Application of convexity in an unsupervised setting

While the main use of the presented work is in a supervised setting, here we suggest how it could be extended beyond known labels. In an unsupervised setting, the class/concept labels could be cluster assignments determined by any clustering method. With these labels, the rest of the analysis can be done in the same way as in the supervised setting. Here we show that the graph convexity of clusters could be used as an additional criterion for evaluating the quality of the clusterings.

For this experiment, we use a subset of 100 randomly chosen ImageNet classes and the corresponding images from the validation set, i.e. in total 5000 images. We extract representations of the last layer in data2vec [7]. We choose four clustering methods (k-means clustering [8], spectral clustering, Gaussian mixture, Ward's hierarchical agglomerative clustering [9]) and various combinations of hyperparameters for them: number of clusters for all the methods (10, 50, 100), initialization strategy for k-means (random, k-means++ [10]), and two hyperparameters for spectral clustering: label assignment (k-means, discretize) and method for construction of the affinity matrix (nearest neighbors, radial basis function kernel). These combinations give us in total 24 clustering methods.

We evaluate each clustering with a number of methods. First, we evaluate the graph convexity (with labels assigned by the clusterings). Since we want to compare results for different number of clusters/classes, we need to normalize the convexity score, so that it becomes independent of this number. We do it in the following way:

$$\text{convexity}_{\text{normalized}} = \frac{\text{convexity} - b}{1 - b}, \tag{35}$$

where b is the baseline convexity score depending on the number of classes/clusters ($b = \frac{1}{n_{\text{classes}}}$).

Since, in this case, we know the ImageNet labels, we can use them to evaluate the clustering. Specifically, we compute the adjusted Rand index (number of agreeing pairs over the number of all pairs, corrected for chance) [11] and adjusted mutual information (mutual information adjusted to account for chance) [12].

We also evaluate the clustering without any labels with Silhouette coefficient [13] (using the mean intra-cluster distance and the mean nearest-cluster distance), Calinski-Harabasz [14] (ratio of the sum of between-cluster dispersion and of within-cluster dispersion), and Davies-Bouldin [15] (average similarity measure of each cluster with its most similar cluster, where similarity is the ratio

	Convexity	Rand	MI	Silhouette	C-H	D-B
Convexity normalized						
Rand adjusted	0.632					
Adjusted mutual information	0.913	0.868				
Silhouette	0.862	0.68	0.855			
Calinski-Harabasz	0.751	0.068	0.473	0.594		
Davies-Bouldin	-0.713	-0.636	-0.764	-0.41	-0.449	

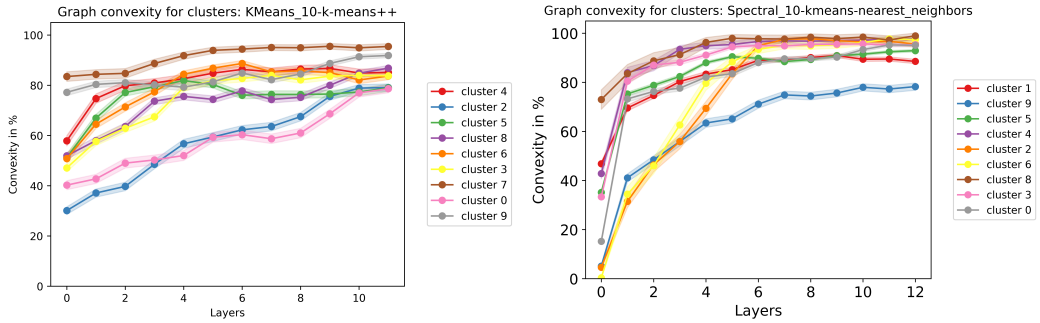
Supplementary Table 2: Pearson correlations for convexity and clustering evaluation metrics for 24 clustering methods on representations of the data2vec pretrained model. More details in Section 2.3. Note that the negative correlations with Davies-Bouldin are caused by the fact that lower Davies-Bouldin score indicates better clustering, whereas it is the opposite for the rest of the scores.

of within-cluster distances to between-cluster distances). A better clustering gets lower Davies-Bouldin score, whereas in the other score higher values mean better separation between the clusters. A potential drawback of these three scores is that they in general assign higher values to convex clusters, hence, are likely to be highly correlated with our convexity score.

We compute Pearson correlation for these scores (over all the clustering methods). The results can be found in Supplementary Table 2. We see that the convexity score is strongly correlated to other metrics estimating the quality of clustering. However, the correlations are not perfect, indicating that convexity can be used as an additional metric picking up different qualities.

Supplementary Figure 3 shows how the graph convexity of clusters evolves throughout the network. Analyses like this can be used for a thorough investigation of the quality of clustering and identification of different types of clusters.

In the case of lower number of clusters than the original 100 ImageNet classes, these clusters can be seen as natural "superclasses" of the selected ImageNet classes. Convexity is a good criterion also in this case because it can give additional information on the coherence of the "superclasses".

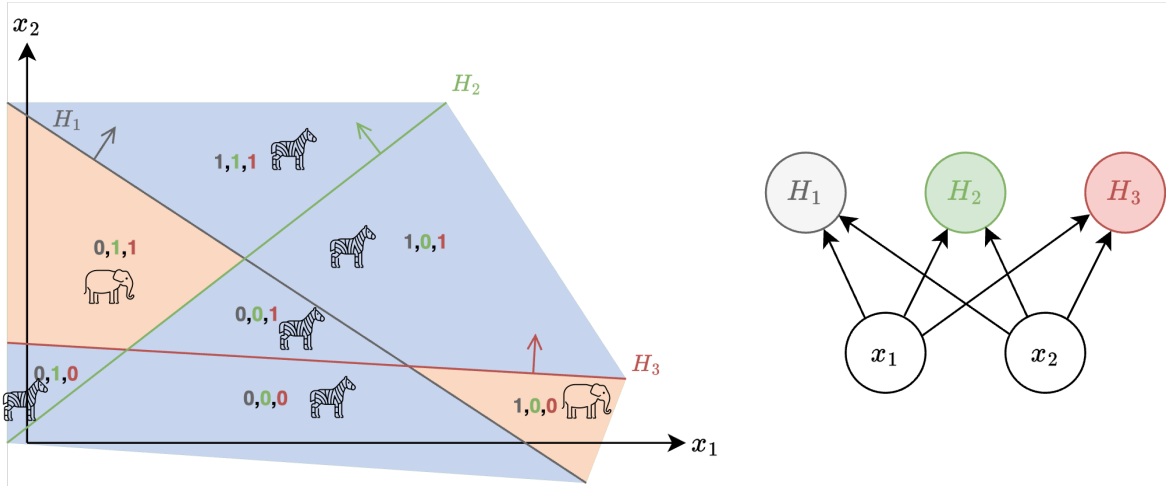


Supplementary Figure 3: Evolution of graph convexity scores of each cluster for two clustering methods: k-means (left) and spectral clustering (right).

2.4 Comparison of graph convexity to the Polytope lens

We will discuss how measuring the graph convexity captures different properties compared to the existing analysis tool, the Polytope lens [16]. Both our work and the polytope lens compute distances between points of the same class in representation space. The polytype lens computes the Hamming Distance (HD) between activations of two points of the same class, whereas the graph convexity score (GCS) computes the proportion of the shortest path (on the graph) that is contained within the same class. The two methods capture different properties and are complementary methods.

To illustrate, consider the two-class problem depicted on Supplementary Figure 4, where an MLP with two activations in the input layer, denoted x_1 and x_2, and three activations in the subsequent layer, denoted H_1, H_2, and H_3, is used. For simplicity, we assume that each of the seven regions has the same number of data points and that the data points are uniformly distributed within the region. Analyzing this network with the polytope lens will yield a HD between 0 (class points are



Supplementary Figure 4: The hyperplanes and the decision regions created by an MLP with two input activations and three output activations. The code in each region denotes whether the given activation H_i is activated.

compact within one of the seven regions) and 3 (class points are distributed so that all 3 neurons flip activation). Analyzing the network with the convexity score will yield a GCS between 0 (no paths will be in the class region) and 1 (the entire path will be within the class region).

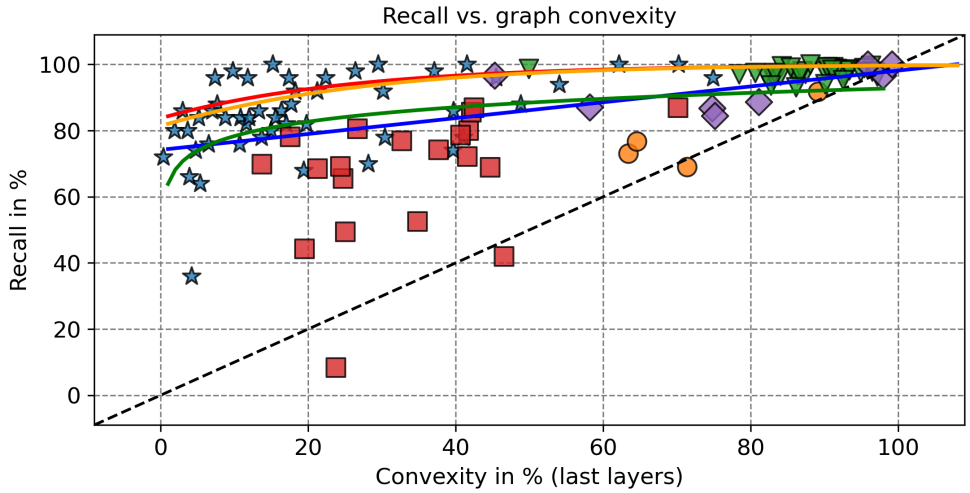
For the Elephant class, we get HD(Elephant) = 3 and GCS(Elephant) = 0.5, whereas for the Zebra class, we get HD(Zebra) = 1.8 and GCS(Zebra) = 1. Thus both methods find that the Elephant class is widely distributed compared to the Zebra class. However, GCS(Zebra) = 1 finds that the Zebra class is fully connected, whereas the HD(Zebra) = 1.8 indicates that the Zebra class is widely distributed. Considering the analysis of both the polytope lens and graph convexity, we can thereby conclude that the Zebra class is fully connected but still distributed. This conclusion can only be made by using both methods, thereby demonstrating HD and GCS are complementary analysis tools.

2.5 Relation between graph convexity and accuracy after fine-tuning

We used the data from Figure 3c from the manuscript (per-class convexity scores and accuracies) to fit three models to capture the relationship between graph convexity and accuracy after fine-tuning:

1. linear (M1): recall $= a \cdot \text{conv} + b$;
2. logarithmic (M2): recall $= a \cdot \log(\text{conv}) + b$;
3. log-linear (M3): $\log(\text{1-recall}) = a \cdot \text{conv} + b$;
4. log-linear transformed (M4): $\log\left(\frac{1-\text{recall}}{\text{recall}}\right) = a \cdot \text{conv} + b$;

Supplementary Figure 5 shows the fitted models for all modalities. Supplementary Table 3 shows a statistical analysis for these models. The only significant conclusion (with $\alpha = 0.05$) is that the fitted linear model explains the data better than a constant (mean of recalls). Even though the data visually suggest a log-linear relationship, the statistical analysis does not support this hypothesis.



Supplementary Figure 5: Graph convexity of a subset of classes in the pretrained models (last layer) vs. recall rate of these individual classes in the fine-tuned models for all data domains. The blue, green, red, and orange lines are the best M1, M2, M3, and M4 models for all modalities combined, respectively.

	all modalities	images	human activity	text	audio	medical
R^2 M1	0.268	0.072	0.690	0.121	0.008	0.183
R^2 M2	0.188	0.041	0.657	0.099	0.002	0.104
R^2 M3	-0.211	-0.257	0.533	0.079	-0.397	-109.684
R^2 M4	-0.136	-0.198	0.582	0.113	-0.386	-14.963
p M1 vs. constant	0.952	0.882	0.763	0.606	0.509	0.594
p M1 vs. M2	0.711	0.700	0.526	0.520	0.506	0.543
p M1 vs. M3	0.996	1.000	0.601	0.539	0.835	1.000
p M1 vs. M4	0.990	1.000	0.574	0.507	0.829	0.999

Supplementary Table 3: Coefficients of determination (R^2) for each model and p-values in F-test between the linear model and the others (*constant* denotes the mean of the recalls). In columns, we have the individual modalites and all combined. Note that *all modalities* includes only 50 image classes (as shown in Figure 3c in the manuscript and in Supplementary Figure 5, whereas *images* takes into account all 1000 classes.

Supplementary References

[1] Boyd, S. P. & Vandenberghe, L. Convex optimization (Cambridge Univ. Press, 2010).

[2] Bishop, C. M. & Nasrabadi, N. M. Pattern recognition and machine learning (Springer, 2006).

[3] Ansuini, A., Laio, A., Macke, J. H. & Zoccolan, D. Intrinsic dimension of data representations in deep neural networks. Advances in Neural Information Processing Systems **32** (2019).

[4] Ceruti, C. et al. Danco: dimensionality from angle and norm concentration. arXiv preprint arXiv:1206.3881 (2012).

[5] Fan, M., Gu, N., Qiao, H. & Zhang, B. Intrinsic dimension estimation of data by principal component analysis. arXiv preprint arXiv:1002.2050 (2010).

[6] Carter, K. M., Raich, R. & Hero III, A. O. On local intrinsic dimension estimation and its applications. IEEE Transactions on Signal Processing **58**, 650–663 (2009).

[7] Baevski, A. et al. Data2vec: A general framework for self-supervised learning in speech, vision and language. In International Conference on Machine Learning, 1298–1312 (PMLR, 2022).

[8] Lloyd, S. Least squares quantization in pcm. IEEE transactions on information theory **28**, 129–137 (1982).

[9] Ward Jr, J. H. Hierarchical grouping to optimize an objective function. Journal of the American statistical association **58**, 236–244 (1963).

[10] Arthur, D. & Vassilvitskii, S. k-means++: The advantages of careful seeding. Tech. Rep., Stanford (2006).

[11] Hubert, L. & Arabie, P. Comparing partitions. Journal of classification **2**, 193–218 (1985).

[12] Vinh, N. X., Epps, J. & Bailey, J. Information theoretic measures for clusterings comparison: is a correction for chance necessary? In Proceedings of the 26th annual international conference on machine learning, 1073–1080 (2009).

[13] Rousseeuw, P. J. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. Journal of computational and applied mathematics **20**, 53–65 (1987).

[14] Caliński, T. & Harabasz, J. A dendrite method for cluster analysis. Communications in Statistics-theory and Methods **3**, 1–27 (1974).

[15] Davies, D. L. & Bouldin, D. W. A cluster separation measure. IEEE transactions on pattern analysis and machine intelligence 224–227 (1979).

[16] Black, S. et al. Interpreting neural networks through the polytope lens. arXiv preprint arXiv:2211.12312 (2022).