A Appendix

A.1 Details of numerical experiments

All our experiments were performed in PyTorch (35) (version 1.0) on a Linux workstation with 64GB of RAM and a GeForce GTX 1080 Ti NVIDIA graphic card. The code to compute the ID estimates with the TwoNN method and to reproduce our experiments is available at this repository. The data is downloadable at this link.

A.1.1 Datasets

Custom dataset A dataset of 1400 images developed for a neurophysiological study (20). The dataset consisted of 40 three-dimensional (3D), computer graphics models of both natural and man-made objects, each rendered in 36 different views, obtained by combining in-plane and in-depth rotations of the 3D models with horizontal translations and size variations. As a result, the image set encompassed a spectrum of object identities, poses and low-level features (e.g., luminance, contrast, position, size, aspect ratio, etc.), but without reaching the size, complexity and variety of shapes and identity-preserving transformations that are typical of naturalistic image sets, such as ImageNet.

A.1.2 Architectures

We describe the architectures used in order of appearance in the main text.

VGG-16-R We removed the last hidden layers (the last convolutional and all the dense layers) of a VGG-16 network (19) pre-trained on ImageNet (11) and substituted it with randomly initialized layers of the same size except for the last hidden layer, in order to match the correct number of categories (40) of the custom dataset described in A.1.1. We then fine-tuned it on the $\simeq 85\%$ of the data. More specifically we used 30 images for each category as training set and we tested on the remaining 6 images for each category. We called this network VGG-16-R (where R stands for restricted, with reference to this small dataset). For the fine-tuning, we used a SGD with momentum 0.9, and a learning rate of 10^{-4} in the last convolutional layer and of 10^{-3} in the dense layers. The other layers were kept frozen. The generalization performance after 15 epochs was $\approx 88\%$ accuracy on the test set.

Standard architectures pre-trained on ImageNet We instantiated fourteen pre-trained networks that are representative of the state-of-the-art models used in visual object recognition and image understanding: AlexNet (11), eight models belonging to the VGG class (11,13,16 and 19 with and without batch normalization) (19)), and five models belonging to the ResNet class (18,34,52,101,152) (18). All these models are available for download in Pytorch (35) at torchvision/models.html.

Small convolutional network for the experiments on the MNIST dataset We trained a small convolutional network on the MNIST dataset (36). The sequence of layers is: a convolutional layer with 1 input channel, 32 output channels and a kernel size of 3; a max pooling layer of kernel size 2; a convolutional layer with 32 input channels, 64 output channels and a kernel size of 3; a max pooling layer of kernel size 2; a fully connected layer with 1600 inputs and 128 outputs; a fully connected layer with 128 input and 10 output units; a softmax. We used ReLU non-linearity after each convolutional, pooling and fully connected layer. The stride is always set to zero. The network has been trained for 200 epochs with a small learning rate ($lr = 0.0004$) and zero momentum on the original dataset, for 5000 epochs $lr = 0.0001$ and momentum 0.9 on MNIST* and for 500 epochs $lr = 0.0005$ and momentum 0.9 on MNIST†.

VGG-16 adapted for CIFAR-10 We used the VGG-16 model adapted for CIFAR-10 available at github.com/kuangliu/pytorch-cifar in two series of experiments. In our experiment on the dependency of the ID (in the last hidden layer) on random initialization (results are in Sec. A.1.4) we trained the network for 300 epochs starting with a learning rate of 0.1 and reducing it by a factor 0.1 after every 100 epochs. L_2 regularization was applied with a weight decay set at 5×10^{-4}. In two experiments on dynamics we used the same configuration (see Fig. 9A) and one with a smaller learning rate $lr = 0.005$ (Fig. 9B,C), in order to elucidate better the early phases of the dynamics.

Checkpoints In each experiment we defined architecture-specific checkpoints from where to extract and analyze the representations. The only exception was in the case of the network used for MNIST, where we extracted the representations and performed the analysis in all the layers, in the experiments described in sections 3.4, 3.5. As a general rule, we always extracted representations at pooling layers after a convolution or a block of consecutive convolutions, and at fully connected layers. In the experiments with ResNets, we extracted the representations after each ResNet block (18) and the average pooling before the output. Depending on the computational demands of our experiments, we extracted and analyzed data samples of different sizes, we describe this in the following section A.1.3.

A.1.3 Estimating intrinsic dimension

Experiments with the custom dataset In this experiment (Fig. 2A,B), we fine-tuned the last layers of a VGG-16 network pre-trained on ImageNet using the $\approx 80\%$ of the 1440 images of the dataset in (20) (30 images for each category in the training set, the remaining 6 images for each category as test set). The whole dataset was used to estimate the ID of the representations across the layers of the network. The values of the ID reported in our analysis are the averages resulting from randomly sampling 20 times the 90% of the activations at each checkpoint layer. The error bars are the standard deviations across these estimates. In the decimation analysis (Fig. 2B) we proceeded as described in (16). After a random shuffling, we splitted the dataset X in a k-fold way, with k ranging from 20 to 1. The k-fold splits yielded k ID estimates at each layer on roughly N/k of the data. The k ID estimates were then averaged and the standard deviation were computed.

Experiments with ImageNet In the experiments with the pre-trained state-of-the-art networks, we performed two kinds of analysis. In the first one (Fig. 3A,B), we sampled randomly 500 images from each of the 7 most populated ImageNet categories: "koalas", "shih-tzu", "rhodesian", "yorkshire", "vizsla", "setter", "butterfly". These 7 sets were kept fixed in all the subsequent analysis. Let us call X_i the i-th set. We then estimated the ID of the resulting object manifolds across the layers of the networks, independently for each category. For each i, we randomly subsampled from the representations of X_i at each checkpoint layer the 90% of the data (450 data points) for 5 times and we computed their IDs. We then averaged these 7 values obtaining a category-specific estimate of the ID at each layer. We finally averaged the IDs obtained for the 7 categories and computed their standard deviations. In the second analysis (Fig. 4A,B), we randomly sampled 2000 images for 5 times from the ImageNet training set. Let us call X_i the i-th of these samples. We computed their representations $R_i^{\text{last hidden}}$ in the last hidden layer, then we randomly subsampled, in each $R_i^{\text{last hidden}}$, the 90% of the data (consisting of 1800 data points) for 20 times and we computed their IDs. We then pooled together all these 100 ID estimates, computed their average and their standard deviation: these are respectively our final ID estimate and its error. Notice that, in this case, the ID estimates refer to random mixtures of all possible object categories of ImageNet.

Experiments with MNIST In these experiments (Fig. 6B, black line), we randomly sampled a set of 2000 images from the test set; this set - called X in the following - was kept fixed. We extracted the activations R^l at each layer l of the trained network described in A.1.2. For $l = 0$ the representations are the original images. For each layer l we randomly subsampled from R^l the 90% of the data (1800 data points) for 50 times and we computed their IDs. We then averaged these ID values and computed their standard deviation: these are respectively our final ID estimate and its error.

Experiments with VGG-16 adapted for CIFAR-10 In these experiments (Fig. 9) we randomly sampled 500 images from each of the 10 CIFAR-10 classes; this set was kept fixed. Then we proceeded similarly to the MNIST case described above.

A.1.4 Further results

Variability of the ID across object categories In section 3.1, we showed that, across layers, the ID displays a typical 'hunchback' profile. In particular, the average of seven class-specific ID profiles is shown in Fig. 3A,B and their standard deviation is reported in Fig. 3A. To give a better intuition of how consistent this trend was across image categories, the ID profile of the seven individual object classes across the layers of AlexNet is reported in Fig. 8.

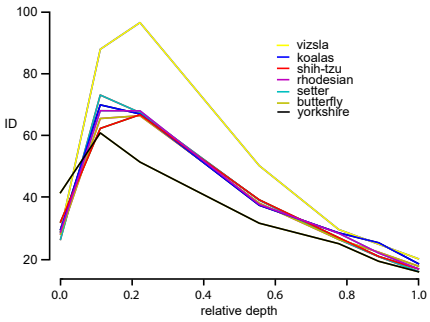

Figure 8: The ID variation across layers is generally consistent across object classes. This figure refers to AlexNet in Fig. 3, where the average across classes is reported.

Variability of the ID across random initializations In section 3.2, we showed a correlation between ID in the last hidden layer and test accuracy across a wide variety of architectures. Based on this result, one could wonder if this correlation is also present within a specific model retrained with different random initializations of the weights. To address this question, we estimated the variability of the ID in the last hidden layer of a VGG-16 adapted for CIFAR-10 across 50 different trainings, finding no correlation with accuracy (r=-0.003), likely because of the little variation in accuracy produced by different random weight initializations. This suggests that differences in accuracy across well-trained networks (see Fig. 4) are mostly due to differences in the architecture.

Dynamics In section 3.3, we observed that, during training, the ID of intermediate layers and the last hidden layer undergo opposite trends. To explore more carefully this finding, we monitored the evolution of the ID profile during training of a VGG-16 network with CIFAR-10. We found qualitative confirmation of the observations reported in Fig. 5C, including a flat ID profile in the untrained network (see Fig. 9A, black thick curve). Moreover, when we inspected more closely the early phase of the dynamics (by slowing down the learning rate and augmenting the 'temporal' resolution of our observations), we found that in the last hidden layer the evolution of the ID is non-monotonic, even in presence of negligeable overfitting (see Fig. 9B,C). Differently from what reported by (15), this suggests that the ID in the last hidden layer of a deep network is not always a reliable predictor of the onset of overfit, and whether this is the case may depend on the specific architectures and data used.

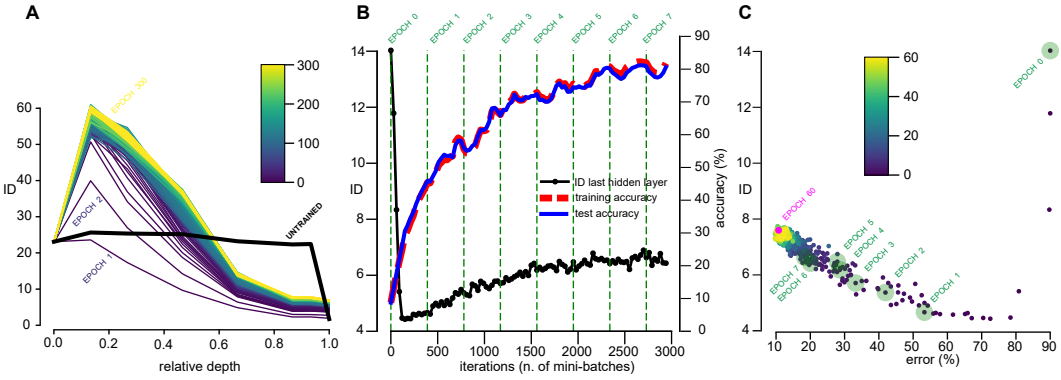

Figure 9: **Dynamics of the ID on a VGG-16 trained on CIFAR-10. A** Dynamics of the 'hunchback' shape for a typical training history. The black thick line refers to the untrained network. The color codes for the training epochs. **B** Training and test accuracy (dashed red and blue curves respectively) and ID in the last hidden layer in the early phases. The dynamics of the ID is non-monotonic; at the same time there is no substantial overfitting. **C** ID vs. test error (same data as in **B**).