Interpolation, extrapolation, and local generalization in common neural networks

Laurent Bonnasse-Gahot

Interpolation, extrapolation, and local generalization in common neural networks

Laurent Bonnasse-Gahot

Centre d'Analyse et de Mathématique Sociales
CAMS, UMR 8557 CNRS-EHESS
École des Hautes Études en Sciences Sociales
54 bd. Raspail, 75006 Paris, France
lbg@ehess.fr

July 18, 2022

Abstract

There has been a long history of works showing that neural networks have hard time extrapolating beyond the training set. A recent study by Balestriero et al. (2021) challenges this view: defining interpolation as the state of belonging to the convex hull of the training set, they show that the test set, either in input or neural space, cannot lie for the most part in this convex hull, due to the high dimensionality of the data, invoking the well known curse of dimensionality. Neural networks are then assumed to necessarily work in extrapolative mode. We here study the neural activities of the last hidden layer of typical neural networks. Using an autoencoder to uncover the intrinsic space underlying the neural activities, we show that this space is actually low-dimensional, and that the better the model, the lower the dimensionality of this intrinsic space. In this space, most samples of the test set actually lie in the convex hull of the training set: under the convex hull definition, the models thus happen to work in interpolation regime. Moreover, we show that belonging to the convex hull does not seem to be the relevant criteria. Different measures of proximity to the training set are actually better related to performance accuracy. Thus, typical neural networks do seem to operate in interpolation regime. Good generalization performances are linked to the ability of a neural network to operate well in such a regime.

1 Introduction

Deep learning is the modern rebranding of artificial neural networks, which were revived ten years ago and have been very successful since then in many domains including computer vision and natural language processing (see LeCun et al., 2015; Schmidhuber, 2015, for reviews). But in spite of this tremendous success, some authors also point out the limitations of these current techniques, described as essentially curve-fitting models (Chollet, 2021; Marcus, 2018). Contrary to human beings, these techniques are very "data-hungry", as they need to have access to a very large training set, and have limited generalization capabilities, well below human abilities (Marcus, 2018). In particular, neural networks do not extrapolate well beyond their training data: there is a long history of results on this point (Barnard and Wessels,

1992; Haley and Soloway, 1992; Marcus, 1998), and more recent results come to a similar conclusion (see for example Barrett et al., 2018; Lake and Baroni, 2018; Saxton et al., 2019). Recently, Balestriero et al. (2021) have challenged the idea that neural networks essentially perform interpolation of the training data. The authors provide an operational definition of interpolation/extrapolation: there is interpolation whenever a new sample lie within the convex hull of the training data. The argument is then essentially based on the well-known curse of dimensionality: in high dimension, every point is far from another. In particular, in high dimension, any new point almost surely lie outside of the training set convex hull. Given the high-dimensionality nature of the data that we are typically dealing with, the authors conclude that neural networks cannot do interpolation and thus contrary to previous claims always actually operate in an extrapolation regime. Beyond looking at the input (pixel) space, Balestriero et al. (2021) extend their analysis to the study of the embedding space constructed by a neural network, anticipating the criticism that interpolation does not happen in input space, but rather in such embedding space. In this case too, their argument remains the same: due to the high dimensionality of such space, the model still operates in extrapolation regime. A good point of this work is the will to provide a quantitative investigation of this debate. Although we agree on the facts about the curse of dimensionality, we do not agree that it ruled out the possibility that current neural networks essentially perform interpolation. Input data such as images are indeed high-dimensional at the surface level, but they actually often live in a lower dimensional space called the intrinsic manifold. This refers to the manifold hypothesis, and is at the basis of dimensionality reduction techniques such as Isomap (Tenenbaum et al., 2000), Locally Linear Embedding (Roweis and Saul, 2000)—see Cayton (2005) for a review. Similarly, the embedding space defined by the neural space (the last layer before the categorical decision) is not the latent space itself, nor is any of its affine subspace, but there exists a low-dimensional (latent) space that implicitly characterizes the neural space.

There has been a recent surge of works in neuroscience that show how (real-world biological) neural activities have to be understood as acting on a lower-dimensional intrinsic manifold (Archer et al., 2014; Sadtler et al., 2014; Cunningham and Byron, 2014; Gallego et al., 2017; Chung and Abbott, 2021; Jazayeri and Ostojic, 2021). In the machine learning literature too, different works have also shown that both input data and neural activities in the hidden layers of artificial neural networks actually lie in a low-dimensional space (Ma et al., 2018; Ansuini et al., 2019; Pope et al., 2021). In such a low-dimensional manifold, the models might operate in interpolation mode according to the convex hull definition, although this might not be obvious in the higher dimensional space defined by the neural activities. To clarify this point, let us take a look at Fig. 1 that present two very simple illustrative examples. Fig. 1A describes two neurons with bell-shaped tuning curves along a one-dimensional space (the tuning curve represents the response of a neuron to a range of stimulus; see for example Hubel and Wiesel, 1959; Henry et al., 1974; Taube et al., 1990). Let us imagine that the training set is made of the two red dots pictured in Fig. 1A. The convex hull of this training set is the red line, and we consider as the test set all the points, in blue, between these two training samples. By definition here they all belong to the convex hull of the training set. Now, if we consider how this situation translates in the state space defined by the activities of the two neurons, the picture changes drastically (Fig. 1B): all the test samples lie outside the new convex hull of the training set. A similar situation is exemplified in Fig. 1C and 1D. It might thus be misleading to directly look at the neural activities, as the high-dimensionality of this neuronal space is only apparent, and can actually be seen as lying in a lower dimensional manifold.

In an artificial neural network, the activity of a given layer can be seen as a population coding of a low-dimensional space paved with neurons (without necessarily a simple contiguous tuning curve). In this work we use an autoencoder (a multilayer perceptron here) in order to recover the implicit intrinsic

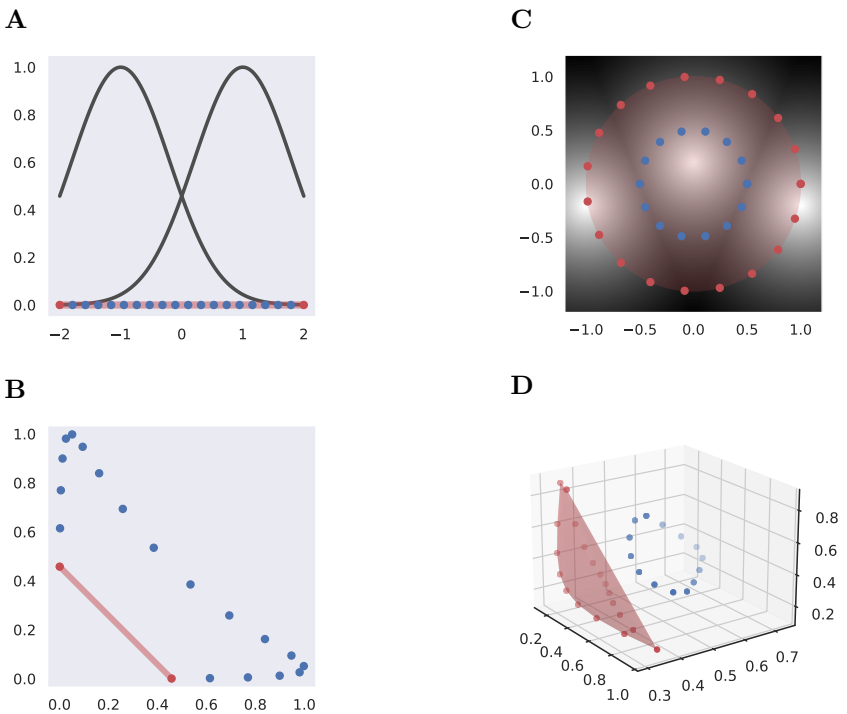

Figure 1: **Convex hull in intrinsic space (top) does not correspond to convex hull in embedded neural space (bottom).** (A) One dimensional intrinsic space. Response of two neurons with bell-shaped tuning curves along a one-dimensional space. If we consider the red points as the training set, the red line corresponds to the convex hull; then all the blue points, considered here as test samples, between the two red ones lie within the convex hull. But if we now move to the neural space defined by the activities of these two neurons, (B), none of these blue points actually lie within the convex hull (the red line) defined by the red samples. Similarly, (C) Two dimensional intrinsic space, with three neurons (whose responses are represented in shades of gray). In this space, all the blue points (the inner circle) lie within the convex hull (the red disk) defined by the red points (the outer circle), but turning to the neural space, (D), none of them actually lie within the new red convex hull.

space underlying the neural space. An autoencoder can be used to reduce the dimensionality of any high-dimensional data pertaining to a lower dimensional manifold (Hinton and Salakhutdinov, 2006). Contrary to other techniques such as Principal Component Analysis, Isomap (Tenenbaum et al., 2000) or Locally Linear Embedding (Roweis and Saul, 2000), an autoencoder makes it possible not only to map the activities to positions in a low dimensional latent space, but also to estimate the neural activities from this intrinsic space. Here, we use an autoencoder to probe the representation of a given layer in a neural network after learning, and study a posteriori, without modifying the base network under study, how the classification performance changes with the number of estimated intrinsic dimension. In this latent space, we also look at the fraction of samples from the test set that lie within the convex hull defined by the training set. We find that the intrinsic dimensionality underlying the neural activity is actually quite low, and for such low values all or almost all the test set is included in the convex hull of the training set. We also experiment changing the characteristics of the networks so as to study how the intrinsic dimensionality of the neural space vary with the classification performances, while keeping the same number of ambient dimension. We find that the better the performances the lower the number of intrinsic dimensions. Following Balestriero et al. (2021) definition of interpolation as being in the convex hull, we thus find that the better a model is, the more it is in an interpolative regime. In a second part, we will see that the notion of interpolation as belonging to the convex hull is not the most relevant, and we study properties of the network with respect to the local proximity to elements of the training set. This study reveals that as expected the closer a new test sample of the training set, the higher its probability of being correctly classified: this behavior is typical of interpolation. This phenomenon is actually even more marked the better a model is. It is not the case that the best models manage to extrapolate better: they actually better represent the data so as to take advantage of the interpolative regime in which they operate. Thus, contrary to the claims of Balestriero et al. (2021), classic deep learning techniques does seem to perform in interpolative regime, as described for instance by Marcus (2018) or Chollet (2021). Generalization performances are actually tightly related to the notion of interpolation, at least with the simple models that we consider here.

2 Materials and Methods

2.1 Probing the neural representation

Our method to probe the neural representation is illustrated in Figure 2. A neural network (we consider a multilayer perceptron or a convolutional neural network in the Results section) is trained to learn some classification task. In the Results section we will consider the recognition of handwritten digits with the MNIST database (LeCun et al., 1998) and the classification of natural images with the CIFAR-10 database (Krizhevsky, 2009). The first layer corresponds to the input space. For instance, in the case of MNIST, this input space (pixel space) is of dimension $28 \times 28 = 786$, and in the case of CIFAR-10 $32 \times 32 \times 3 = 3072$, which correspond in all cases to high-dimensional data. One or several hidden layer(s) follow(s) the input layer, depending on the architecture. Although we could probe any layer in the network, we will focus in this work on probing the last layer before the categorical decision, where the categories are organized into linearly separable clusters of neural activities. We call this space the neural space. In all the numerical experiments of the section Results, this neural space has the same ambient dimension, 128. In order to probe it, we use an autoencoder to recover the latent space underlying the neural activities of the layer under scrutiny (Fig. 2B). This is done by training the autoencoder to reconstruct the neural activities, using the training set (the loss is the mean square error here). It is important to note that this analysis is done a posteriori, after the network under investigation has learned its task. Finally, we substitute the neural activities by the ones approximated by the autoencoder, and we look

at the classification accuracy of this hybrid network (Fig. 2C). We repeat this procedure for different numbers of dimension of the estimated intrinsic space (the number of neurons in the bottleneck of the autoencoder). We assume that for a given dimension, if the autoencoder is able to reconstruct the neural activities well enough, this hybrid network should have similar classification performance (on the test set). If the number of dimensions is too low, the classification performances should be lower than the original network. Starting from the true intrinsic dimension, the performances should be equal to the ones of the original network, and plateau for larger values of the estimated intrinsic dimension. Note again that during this analysis all the parts from the original network are left untouched, completely frozen, and that the autoencoder is not trained on the classification task, but only on the task of reconstructing the neural activities.

2.2 Toy example with ten Gaussian categories

In order to validate this general framework, we first go through a simple fully controlled example involving classification of Gaussian categories. In this example we consider 10 categories (as in the MNIST and CIFAR-10 examples that will be used in the Results section). In all the following cases, the stimuli drawn from these categories live in same input space, with $n_{\text{input}} = 32$ dimensions, but depending on the condition the intrinsic space has different dimensions. In order to control for the number of dimensions of the intrinsic space, for a given desired value notated n_{id}, we generate the centers of the categories randomly in a space of dimension n_{id}. These centers are then plunged into a n_{input}-dimensional space by applying a random rotation/reflection matrix. This transformation is generated thanks to a QR decomposition of a random matrix $A = QR$ drawn from a normal distribution with mean 0 and variance 1. This factorization yields Q, which is an orthogonal matrix, and corresponds to a rotation in input space (possibly combined with a reflection). Finally, we generate 5000 stimuli per class for the training set and 1000 stimuli per class for the test set by drawing random realizations of a normal distribution centered in each of these centers and with an isotropic variance scaled so as to yield an average classification performance around 70%.

A multilayer perceptron with one hidden layer of 32 neurons is then trained to classify the categories. It is important to note that in all the different cases, the neural space has then the exact same dimension $n_{\text{neural}} = 32$. After learning, we freeze and analyze this network by probing the hidden layer with the method described above, using different number of neurons in the autoencoder bottleneck as an estimate of the intrinsic dimension. Finally, in each case, using the test set, we compute the classification accuracy of the hybrid network that use the frozen base neural network and the neural activations predicted from the autoencoder. Figure 2D shows that this simple setting indeed enables to recover the true intrinsic dimension. As expected, when the probe has a lower dimension than the true intrinsic dimension, the accuracy of the model is lower than the base model; then, when it reaches the true intrinsic dimension (known here by construction), the model has the same classification accuracy, which then plateaus for greater inner dimension.

2.3 Technical details

We consider two different databases, namely the MNIST database (LeCun et al., 1998) and the CIFAR-10 database (Krizhevsky, 2009). Figure 3 presents the two neural architectures (and their variants) that we consider. We vary the width of the first layer in the case of the multilayer perceptron, and the depth of the convolutional neural network, by repeating a certain number of times the convolutional block described in Fig. 3B. The goal here is to explore different variants of a same architecture, with the same final neural space, but different performances due to the width or depth of the network. For

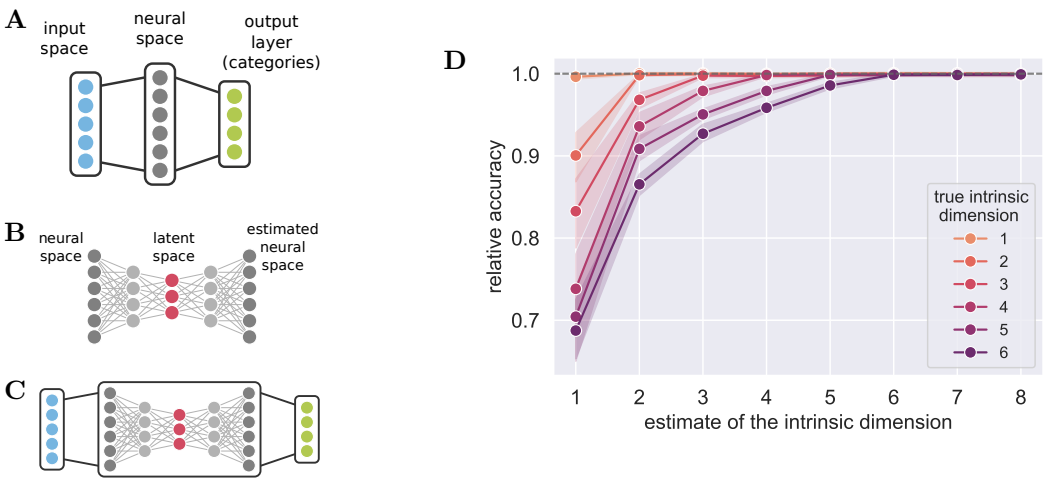

Figure 2: **Description of the method use to probe the representation of a neural space.** A neural network, (A), trained on a given classification task is then analyzed by using an autoencoder, (B), which is trained to reconstruct the neural activities under consideration. Using an hybrid network, (C), made of the original neural network and the activities predicted by the autoencoder, we then compute the classification accuracy on the test set (without any learning/finetuning). Varying the number of neurons in the bottleneck of the autoencoder serves as an estimate of the number of dimensions of the intrinsic space underlying the neural activities. (D) A simple example involving 10 Gaussian categories living in a controlled space of varying intrinsic dimension but same input dimension: when then number of dimensions in the reconstructed latent space reaches the true intrinsic dimension, the classification accuracy of the hybrid network (that forces the neural activities to be estimated through a small dimensional bottleneck) is the same as the one of the original network. The relative accuracy refers to accuracy of the hybrid network divided by the one of the original network (mean absolute accuracy is 71%). For each value of the true intrinsic dimension, the process is repeated for 10 trials, and error bars indicate 95% bootstrap confidence intervals.

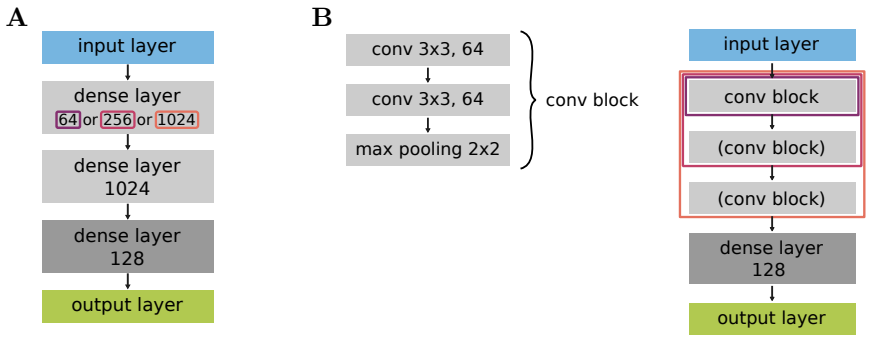

Figure 3: **Artificial neural networks used in the present study.** (A) Multilayer perceptrons with three hidden layers. The number of neurons in the first hidden layer vary so as to induce variability in classification performances, in order to study its relationship with intrinsic dimensionality. (B) Convolutional neural networks. A convolutional block is defined as two conv layers followed by a max pooling layer. We consider three different cases, that use either 1, 2, or 3 such conv blocks. In all cases, the neural space under consideration is the dark gray layer, the hidden dense layer right before the output layer that gives the class probabilities. In all the different cases, this neural space has the same ambient dimension, 128 here.

the multilayer perceptron, each dense layer is followed by a dropout layer (Srivastava et al., 2014), with probability of dropping a unit $p = 0.2$ for the first hidden layer, $p = 0.4$ for the second one, and $p = 0.5$ for the last one. For the convolutional neural network, all the conv layers and the dense layer are followed by a batch normalization layer, and each conv block is followed by a dropout layer, with rate $p = 0.2$ for the first one, $p = 0.3$ for the second one, and $p = 0.4$ for the last one. Before the softmax layer, we also apply dropout with rate $p = 0.5$. All networks are trained through gradient descent using the Adam optimizer (Kingma and Ba, 2015).

The autoencoder used to probe the representation of a given layer is a multilayer perceptron with 256 cells before and after the bottleneck. We consider 2, 4, 8 or 16 neurons in the bottleneck, used as an estimate of the intrinsic dimensionality of the neural space. All neurons use the ReLU activation function, except in the bottleneck layer, that makes use of the linear activation function. From the neural activities of a given layer (the layer before the output layer here), computed from the training set, the autoencoder is trained to reconstruct these neural activities, using the mean square error as the loss function.

For each case, all the results are averaged over 10 trials, with different random initializations, error bars indicating 95% bootstrap confidence intervals.

Computer code. The custom Python 3 code written for the present project makes use of the following libraries: `tensorflow v2.8.0` (Abadi et al., 2015) (using `tf.keras` API, Chollet et al., 2015), `numpy v1.21.4` (Harris et al., 2020), `scipy v1.8.0` (Virtanen et al., 2020), `pandas v1.3.4` (McKinney et al., 2010), `matplotlib v3.4.3` (Hunter, 2007), `seaborn v0.11.2` (Waskom, 2021) and `statsmodels v0.13.2` (Seabold and Perktold, 2010). The code will be made available on GitHub.

3 Results

3.1 Convex hull in latent space

Figure 4 presents the results of the investigation on the latent space using the autoencoder method described in the previous section. All the results converge to the same findings. First and foremost, the latent space underlying the neural activities is of low dimension, with 8 or even 4 dimensions being enough to capture these activities sufficiently well so as to yield the same classification performance.

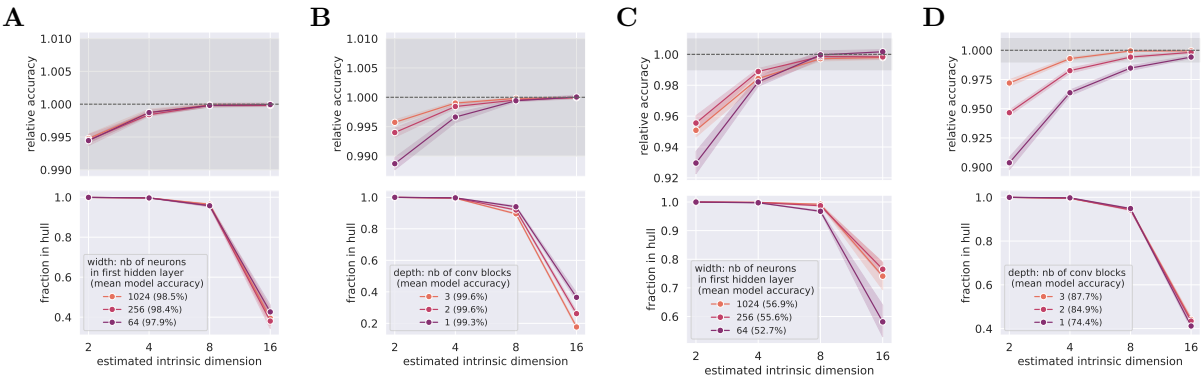

Figure 4: **Estimate of the intrinsic dimension of the neural space along with the percentage of test samples in the convex hull of training set.** Experiment with different data sets, MNIST (A, B) and CIFAR-10 (C, D), and different architectures, multilayer perceptrons (A, C) and convolutional neural network (B, D). The color code goes from violet to yellow with increasing performance. The gray filled area represents a 1% deviation from the model mean accuracy. The x-axis is in log2 scale.

These figures are actually in agreement with the estimations of intrinsic dimensions found for these two data sets by Ma et al. (2018) and Ansuini et al. (2019), using different techniques for this estimation. Moreover, the better the model the lower the intrinsic dimension of its neural space, which echoes the result found by Ansuini et al. (2019). If we compare the results yield by the two data sets, it seems that the intrinsic dimension of the neural space is larger for CIFAR-10 than for MNIST, in agreement with the intuition one can have with respect to the complexity of natural images vs. handwritten digits.
Second, the proportion of the test set within the convex hull of the training set decreases significantly with the number of dimensions, as expected by the famous curse of dimensionality (Balestriero et al., 2021). But interpolation is not "doomed": given the low intrinsic dimension of the neural space, the vast majority of the test samples actually lie within the convex hull of the training set. Hence, adopting the definition of interpolation proposed by Balestriero et al. (2021), the model actually operate in interpolation regime, although one has to look at the latent space that underlies the neural activities.

3.2 Local proximity to the training set

In Figure 4, for a given estimated intrinsic dimension, no obvious relationship emerges from looking at the association between generalization performance and percentage of test set in training convex hull. One can ask whether the definition of interpolation as belonging to the convex hull is the most appropriate. Chollet (2021) rather talks about *local generalization* (see also the related blog page at `https://blog.keras.io/the-limitations-of-deep-learning.html`): performances do not deteriorate abruptly as soon as you get outside the convex hull, and conversely, a new sample might lie within the convex hull but be relatively far from any training point and then more prone to error in classification, if the model is essentially in interpolative mode. Although with the 2, 4 and 8 dimensional cases almost all the test samples fall within the convex hull of the training set, in the case of a 16 dimensional latent space, about half of the test samples fall within the convex hull. This gives the opportunity to study the relationship between the fact of falling within the convex hull and the probability of being correctly classified. All the following examples consider the 16 dimensional latent space only. We also look at different measures of proximity to the training set. Figure 5 presents the results of this investigation for the case of multilayer perceptrons trained on the MNIST data set (corresponding to Fig. 4A), and with the use of the Euclidean distance. Supplementary figures present similar results for the other architectures and data sets presented in Fig. 4, as well as a variety of distances, namely (nearest neighbor) Euclidean

distance, cosine distance, and class conditional Euclidean distance (distance to nearest neighbor of the right class). All in all, the results can be summarized as the following (interestingly, for some distance measures the CIFAR-10/MLP case diverges from that, but it actually goes well with the main point as in this case these models perform quite poorly).

First, be it in neural space (Fig. 5A) or in latent space (Fig. 5B), the closer a new sample to the training set, the greater the chance of a correct classification. Points in the convex hull are closer to the training set than points outside, but in both cases, correctly classified new samples are closer to the training set than incorrectly classified ones (Fig. 5C). If we try to predict the probability of being correctly classified by taking into account both the fact of being in the convex hull and the distance to the training set (and the interaction between these two factors), fitting a logistic regression show that the main contribution comes from the distance factor (Fig. 5D), in the direction as expected: the closer a sample the more its chance of being correctly classified (note: in this case, Fig. 5D, being in the convex hull increases the chance of being correctly classified, as might be expected, but the full range of cases presented in supplementary material show that this is not very reliable, contrary to the distance to training set). Finally, if we look directly in the neural space, we observe the same finding that the distance to the training set is smaller when the test samples are correctly classified (Fig. 5E, top), and this effect is actually stronger the better the model (as quantified by the distance between the distributions of distances for the incorrectly vs. correctly classified new items). Thus, this behavior is not surprising and is very typical of pure interpolation regime. In the end, these results fit well with the "local generalization" picture given by Chollet (2021).

4 Conclusion

We saw that the neural activities of the feature space constructed by a neural network actually live in a much smaller dimensional space, called the intrinsic latent space. In this latent space, the vast majority of new samples from the test set lie within the convex hull defined by the training set, although this is not the case in the high dimensional neural state space, as discussed in the Introduction and Figure 1. If we first go with the convex hull definition of interpolation proposed by Balestriero et al. (2021), this means that these neural networks actually operate in interpolation regime, contrary to the claims made by these authors. Moreover, considering cases with higher dimensional latent space, where all test samples do not necessarily fall within the convex hull, we found that the probability of misclassifying a new sample is actually not well predicted by whether it lies within the convex hull or not. Even within the convex hull, if one test sample is far from the training set, its probability of being misclassified increases with this distance. Conversely, a point outside the hull but in close proximity to the training set is likely to be well classified. To be in the convex hull or not is not important: what seems to matter is the distance to the training set. As soon as one new sample gets far from the training data, be it within the convex hull or not, the performance declines—an example of the *local generalization* described by Chollet (2021). This is typical of an interpolative regime, and is in stark contrast with the generalization abilities of human subjects (Lake et al., 2017). Hence, in spite of the high-dimensionality of the data, the notion of distance to the training set remains an important aspect that governs the generalization performance of an artificial neural network (at least for common architectures like multilayer perceptrons or convolutional neural networks). These artificial neural networks face difficulties extrapolating beyond a local area close to the training set. The models that generalize best are the ones that manage to place themselves in interpolative mode, with a representation where new samples lie close to the training set.

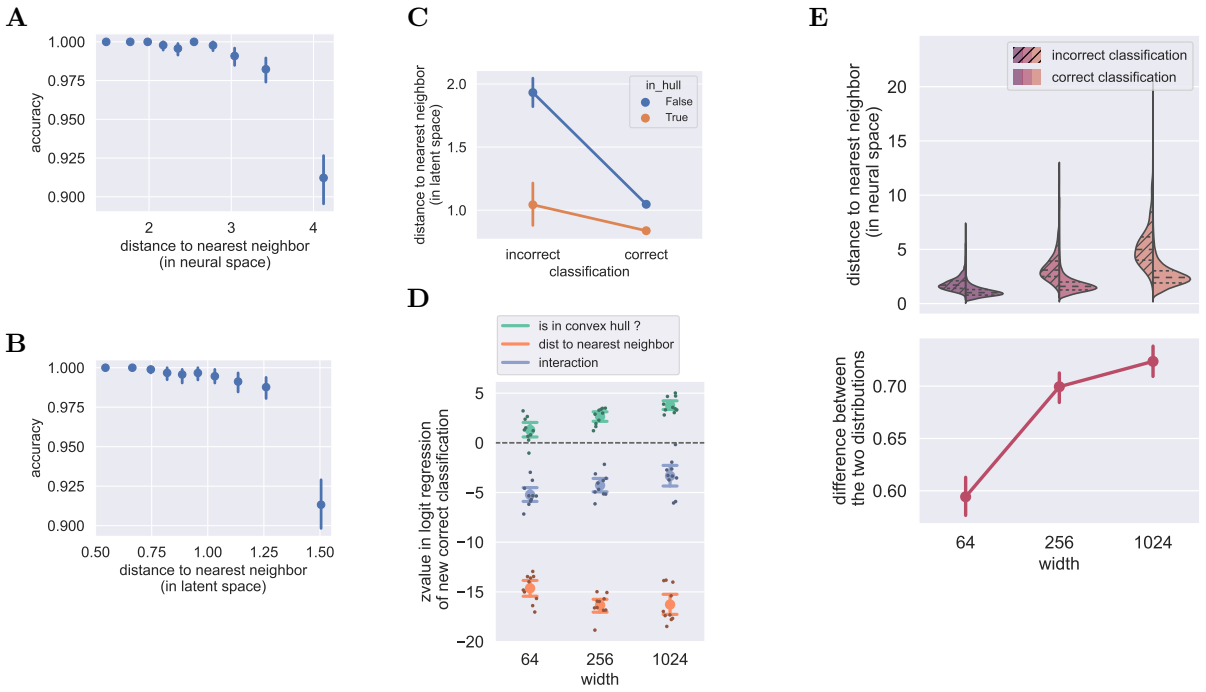

Figure 5: **Distance to training set is indicative of generalization performance.** Results for the MNIST data set and the three multilayer perceptrons considered (see Fig. 3A and Fig. 4A). See Supplementary Figures for other data sets, architectures, and definitions of distance. The distance used here corresponds to the Euclidean distance of a new test sample to the nearest neighbor in the training set. (A) Accuracy as a function of the distance to the training set, in neural space, for the best network (the one with 1024 units in the first layer), for the first trial. Each bin represents 10% of the test data, while error bars indicate 95% bootstrap confidence intervals. (B) Same, in latent space. The latent space is 16 dimensional here (for the other lower values, almost all the samples lie in the convex hull). (C) Average distance to training set as a function of whether a new test samples is correctly classified or not (x-axis), and depending on whether it lies inside ou outside the convex hull of the training set (orange vs. blue). (D) Results of the logistic regression `accuracy ~ distance + in_hull + distance:in_hull`. zvalues (which provides the significance of the corresponding coefficient, a zvalue of 1.96 for instance corresponding to a p-value of 0.05) of the coefficient for each of the two factors and their interaction. Each dot corresponds to a given trial, error bars indicate 95% bootstrap confidence intervals over ten trials. (E) In neural space, (top) distribution of the distance of a test sample to the training set, depending whether the sample is correctly or incorrectly (dashed) classified, for the three multilayer perceptrons considered, and (bottom) for each case the corresponding difference between these two distributions, computed thanks to the Kolmogorov-Smirnov statistic. Error bars indicate 95% bootstrap confidence intervals over ten trials.

References

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., and Zheng, X. (2015). TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org.

Ansuini, A., Laio, A., Macke, J. H., and Zoccolan, D. (2019). Intrinsic dimension of data representations in deep neural networks. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

Archer, E. W., Koster, U., Pillow, J. W., and Macke, J. H. (2014). Low-dimensional models of neural population activity in sensory cortical circuits. *Advances in neural information processing systems*, 27.

Balestriero, R., Pesenti, J., and LeCun, Y. (2021). Learning in high dimension always amounts to extrapolation. *arXiv preprint arXiv:2110.09485*.

Barnard, E. and Wessels, L. (1992). Extrapolation and interpolation in neural network classifiers. *IEEE Control Systems Magazine*, 12(5):50–53.

Barrett, D., Hill, F., Santoro, A., Morcos, A., and Lillicrap, T. (2018). Measuring abstract reasoning in neural networks. In *International conference on machine learning*, pages 511–520. PMLR.

Cayton, L. (2005). Algorithms for manifold learning. *Univ. of California at San Diego Tech. Rep*, 12(1-17):1.

Chollet, F. (2021). *Deep Learning with Python, Second Edition*. Manning.

Chollet, F. et al. (2015). Keras. `https://keras.io`.

Chung, S. and Abbott, L. (2021). Neural population geometry: An approach for understanding biological and artificial neural networks. *Current opinion in neurobiology*, 70:137–144.

Cunningham, J. P. and Byron, M. Y. (2014). Dimensionality reduction for large-scale neural recordings. *Nature neuroscience*, 17(11):1500–1509.

Gallego, J. A., Perich, M. G., Miller, L. E., and Solla, S. A. (2017). Neural manifolds for the control of movement. *Neuron*, 94(5):978–984.

Haley, P. J. and Soloway, D. (1992). Extrapolation limitations of multilayer feedforward neural networks. In *[Proceedings 1992] IJCNN International Joint Conference on Neural Networks*, volume 4, pages 25–30. IEEE.

Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., et al. (2020). Array programming with numpy. *Nature*, 585(7825):357–362.

Henry, G., Dreher, B., and Bishop, P. (1974). Orientation specificity of cells in cat striate cortex. *Journal of Neurophysiology*, 37:1394–1409.

Hinton, G. E. and Salakhutdinov, R. R. (2006). Reducing the dimensionality of data with neural networks. *science*, 313(5786):504–507.

Hubel, D. H. and Wiesel, T. N. (1959). Receptive fields of single neurones in the cat's striate cortex. *The Journal of physiology*, 148(3):574–591.

Hunter, J. D. (2007). Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, 9(3):90–95.

Jazayeri, M. and Ostojic, S. (2021). Interpreting neural computations by examining intrinsic and embedding dimensionality of neural activity. *Current opinion in neurobiology*, 70:113–120.

Kingma, D. P. and Ba, J. (2015). Adam: A method for stochastic optimization. In *Proceedings of the 3rd International Conference on Learning Representations (ICLR)*.

Krizhevsky, A. (2009). Learning multiple layers of features from tiny images.

Lake, B. and Baroni, M. (2018). Generalization without systematicity: On the compositional skills of sequence-to-sequence recurrent networks. In *International conference on machine learning*, pages 2873–2882. PMLR.

Lake, B. M., Ullman, T. D., Tenenbaum, J. B., and Gershman, S. J. (2017). Building machines that learn and think like people. *Behavioral and brain sciences*, 40.

LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *Nature*, 521(7553):436–444.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

Ma, X., Wang, Y., Houle, M. E., Zhou, S., Erfani, S., Xia, S., Wijewickrema, S., and Bailey, J. (2018). Dimensionality-driven learning with noisy labels. In *International Conference on Machine Learning*, pages 3355–3364. PMLR.

Marcus, G. (2018). Deep learning: A critical appraisal. *arXiv preprint arXiv:1801.00631*.

Marcus, G. F. (1998). Rethinking eliminative connectionism. *Cognitive psychology*, 37(3):243–282.

McKinney, W. et al. (2010). Data structures for statistical computing in python. In *Proceedings of the 9th Python in Science Conference*, volume 445, pages 51–56. Austin, TX.

Pope, P., Zhu, C., Abdelkader, A., Goldblum, M., and Goldstein, T. (2021). The intrinsic dimension of images and its impact on learning. *arXiv preprint arXiv:2104.08894*.

Roweis, S. T. and Saul, L. K. (2000). Nonlinear dimensionality reduction by locally linear embedding. *science*, 290(5500):2323–2326.

Sadtler, P. T., Quick, K. M., Golub, M. D., Chase, S. M., Ryu, S. I., Tyler-Kabara, E. C., Byron, M. Y., and Batista, A. P. (2014). Neural constraints on learning. *Nature*, 512(7515):423–426.

Saxton, D., Grefenstette, E., Hill, F., and Kohli, P. (2019). Analysing mathematical reasoning abilities of neural models. *arXiv preprint arXiv:1904.01557*.

Schmidhuber, J. (2015). Deep learning in neural networks: An overview. *Neural networks*, 61:85–117.

Seabold, S. and Perktold, J. (2010). statsmodels: Econometric and statistical modeling with python. In *9th Python in Science Conference*.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958.

Taube, J. S., Muller, R. U., and Ranck, J. B. (1990). Head-direction cells recorded from the postsubiculum in freely moving rats. i. description and quantitative analysis. *Journal of Neuroscience*, 10(2):420–435.

Tenenbaum, J. B., De Silva, V., and Langford, J. C. (2000). A global geometric framework for nonlinear dimensionality reduction. *science*, 290(5500):2319–2323.

Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., van der Walt, S. J., Brett, M., Wilson, J., Jarrod Millman, K., Mayorov, N., Nelson, A. R. J., Jones, E., Kern, R., Larson, E., Carey, C., Polat, İ., Feng, Y., Moore, E. W., Vand erPlas, J., Laxalde, D., Perktold, J., Cimrman, R., Henriksen, I., Quintero, E. A., Harris, C. R., Archibald, A. M., Ribeiro, A. H., Pedregosa, F., van Mulbregt, P., and Contributors, S. . . (2020). SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272.

Waskom, M. L. (2021). seaborn: statistical data visualization. *Journal of Open Source Software*, 6(60):3021.