How Deep Networks Learn Sparse and Hierarchical Data: the Sparse Random Hierarchy Model

Umberto Tomasini [1] **Matthieu Wyart** [1]

Abstract

Understanding what makes high-dimensional data learnable is a fundamental question in machine learning. On the one hand, it is believed that the success of deep learning lies in its ability to build a hierarchy of representations that become increasingly more abstract with depth, going from simple features like edges to more complex concepts. On the other hand, learning to be insensitive to invariances of the task, such as smooth transformations for image datasets, has been argued to be important for deep networks and it strongly correlates with their performance. In this work, we aim to explain this correlation and unify these two viewpoints. We show that by introducing sparsity to generative hierarchical models of data, the task acquires insensitivity to spatial transformations that are discrete versions of smooth transformations. In particular, we introduce the Sparse Random Hierarchy Model (SRHM), where we observe and rationalize that a hierarchical representation mirroring the hierarchical model is learnt precisely when such insensitivity is learnt, thereby explaining the strong correlation between the latter and performance. Moreover, we quantify how the sample complexity of CNNs learning the SRHM depends on both the sparsity and hierarchical structure of the task.

1. Introduction

Deep Learning has demonstrated remarkable efficacy across diverse tasks, from image classification (Voulodimos et al., 2018) to the development of chatbots (Brown et al., 2020). This success is not well understood: learning generic tasks requires a number of training points exponential in the data dimension (Luxburg & Bousquet, 2004; Bach, 2017), which is unachievable in practice for images or text that lie in high dimension. Learnable data must then be highly structured. Understanding the nature of this structure is key to various fundamental problems of machine learning, ranging from the existence of scaling laws (Cortes et al., 1993; Hestness et al., 2017; Spigler et al., 2020; Kaplan et al., 2020; Bommasani et al., 2022) that relate the size of the training set to the performance and the emergence of new abilities, the success of transfer learning, or that of unsupervised methods such as diffusion models (Ho et al., 2020).

One view is that learnable data often consists of local features that are assembled hierarchically (Grenander, 1996): a dog is made of a body, limbs, and a face, itself consisting of ears, eyes, etc... This view is consistent with the observation that after training, deep neural networks form a hierarchical representation of the input (Zeiler & Fergus, 2014; Doimo et al., 2020). These properties also mirror the architecture of Convolutional Neural Networks (CNNs) (Lecun et al., 1998; LeCun et al., 2015), which are deep and display small filter sizes. From a theoretical viewpoint, the locality of the features proves to be advantageous to approximate and learn high-dimensional tasks (Favero et al., 2021; Abbe et al., 2021; Bietti et al., 2022; Xiao, 2022; Xiao & Pennington, 2022; Bietti, 2022; Mei et al., 2022; Cagnetta et al., 2023a). Moreover, various hierarchical models of data, organizing local features within a hierarchical structure, have been proposed, (Mossel, 2018; Poggio et al., 2017; Malach & Shalev-Shwartz, 2018; Schmidt-Hieber, 2020; Malach & Shalev-Shwartz, 2020; Cagnetta et al., 2023b; Allen-Zhu & Li, 2023). As detailed in subsection 1.2, deep networks can represent and learn such models more efficiently than shallow networks, both in terms of number of parameters and number of training points.

Intrinsically, hierarchical models have a discrete nature, which seems well-suited for texts. Images however are often approximated as a continuous function of a two-dimensional space (Castleman, 1996), whose label is invariant to smooth transformations. This stability of image labels to small smooth transformations has been proposed in (Bruna & Mallat, 2013; Mallat, 2016) as a key simplification enabling image classification in high dimension. Enforcing such sta-

[*]Equal contribution [1]Institute of Physics, EPFL, Lausanne, Switzerland. Correspondence to: Umberto Tomasini <umberto.tomasini@epfl.ch>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024. Copyright 2024 by the author(s).

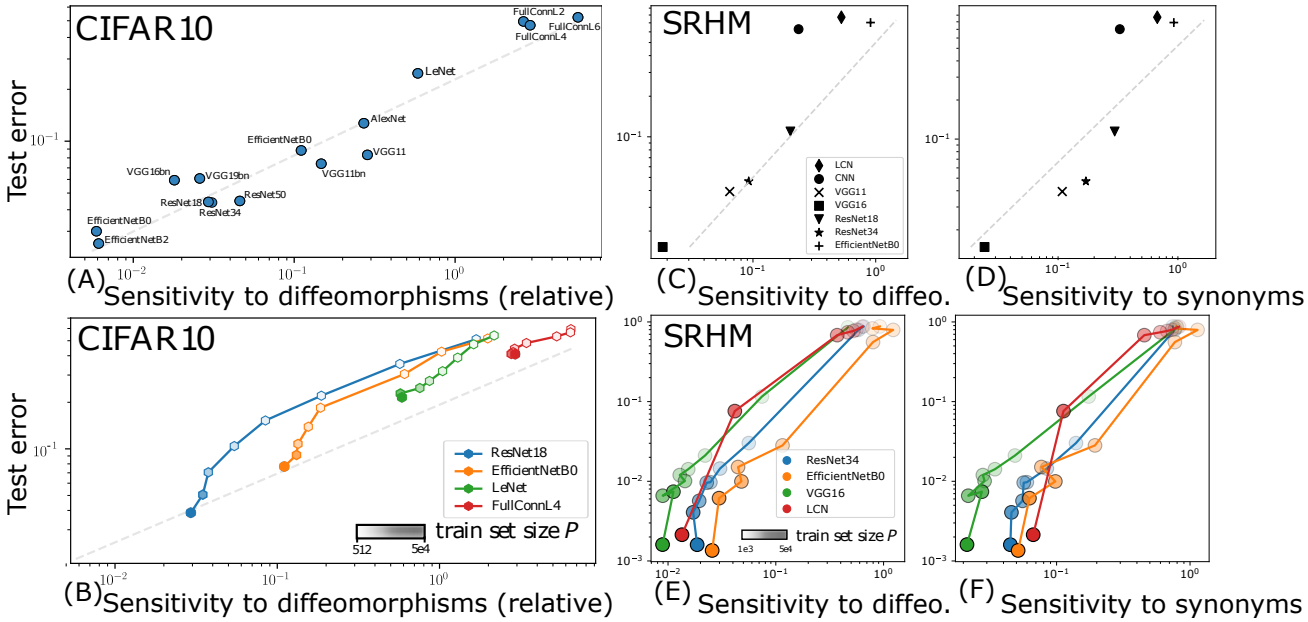

Figure 1. **CIFAR 10:** (A) Test error vs sensitivity to diffeomorphisms of common architectures trained on all CIFAR10, showing a remarkable correlation between the two quantities. A grey line, corresponding to a power-law, guides the eye. (B) Same as (A), for increasing size of the training set P, whose value is indicated by the degree of opacity. The sensitivity to smooth transformations is computed in relative terms to the sensitivity to white noise. (A, B) are adapted from (Petrini et al., 2021). **The SRHM captures these observations:** (C) Test error vs sensitivity to diffeomorphisms of a CNN trained with $P = 7400$ on the SHRM model, with parameters $L = s = s_0 = 2$ and $n_c = m = 10$. The sensitivity to diffeomorphisms is defined as the change of network output induced by a diffeomorphism applied on the input, see Eq. 7. For details about the architectures and their training process, see Appendix B. (D) Same as (C) for sensitivity to exchange of synonyms, defined as the change of the network output induced by an exchange of synonyms (defined in Section 2) applied on the input, see Eq. 6. (E) and (F): as top panels (C) and (D), for increasing P (increasing opacity).

bility in neural networks can improve their performance (Kayhan & Gemert, 2020). Moreover, the stability can also improve during training by learning pooling operations, (Dieleman et al., 2016; Azulay & Weiss, 2018; Ruderman et al., 2018; Zhang, 2019; Tomasini et al., 2023). One finds that (i) there exists a strong correlation between the network's test error and its sensitivity to diffeomorphisms, which characterizes the change of output when a diffeomorphism is applied to the input (Petrini et al., 2021), as recalled in Figure 1 (A) and (ii) the sensitivity to diffeomorphisms decreases with the size of the training set, as shown in Figure 1 (B). Currently, these observations remain unexplained, and it is not clear how they relate with the fact that neural networks build a hierarchical representations of data.

1.1. Our Contributions

- We argue that incorporating sparsity into hierarchical generative models naturally leads to classification tasks insensitive to the exact position of the local features, implying insensitivity to discrete versions of diffeomorphisms.
- To illustrate this process, we introduce the Sparse Random Hierarchy Model (SRHM), which captures the empirically observed correlation between sensitivity to diffeomorphisms and test error, as depicted in Fig.1 (C).
- This correlation arises because a hierarchical representation, crucial for achieving good performance, is learnt precisely at the same number of training points at which insensitivity to diffeomorphisms is achieved. We provide arguments justifying this observation.
- We quantify how the number of training points needed to learn the task, also called sample complexity, of deep networks is influenced by both the sparsity and hierarchical nature of the task, and how it depends on the presence of weight sharing in the architecture.

1.2. Prior Work

Hierarchical representations: It is recognized that deep networks can represent hierarchically compositional functions with significantly fewer parameters than shallow networks

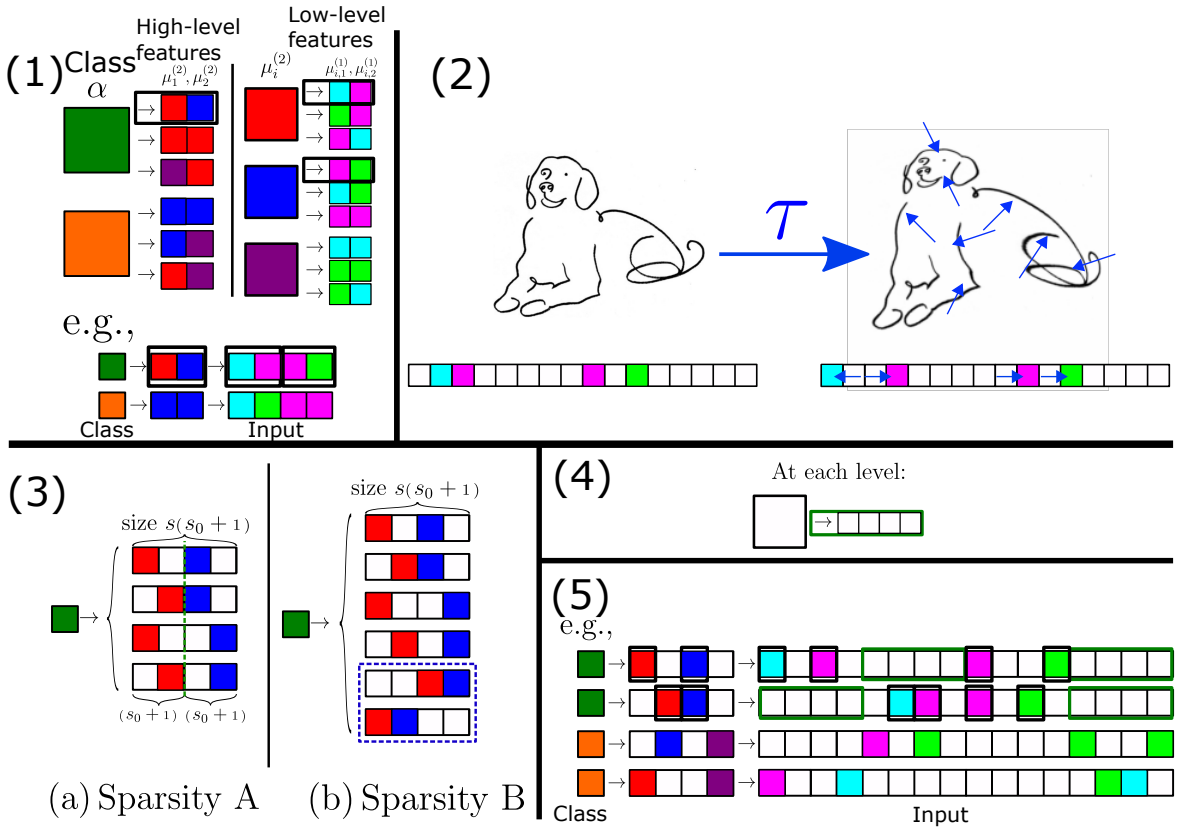

Figure 2. **(1)** On the top of that panel, an instance of the production rules of a Random Hierarchical Model (RHM) with $n_c = 2$ classes, $L = 2$, $m = 3$ synonyms per feature, vocabulary size $v = 3$, and $s = 2$. Here the classes set is $\mathcal{C} = \{\text{green, orange}\}$, the high-level features vocabulary is $\mathcal{V}_2 = \{\text{red, blue, purple}\}$ and the low-level features vocabulary is $\mathcal{V}_1 = \{\text{turquoise, pink, green}\}$. On the bottom, a couple of examples generated via the rules above are shown. The first example is generated by the production rules in the black boxes (i.e. the green label generates (red,blue), which themselves generate the couples (turquoise, pink) and (pink, green). **(2)** Top: effect of a diffeomorphism τ on a dog. The blue arrows represent the displacement field induced by τ. Bottom: effect of a diffeomorphism τ on an instance of a sparse generative hierarchical task. **(3)** Different definitions of sparsity. (A) Each one of the s informative features is embedded in a sub-patch of size $s_0 + 1$ with strictly s_0 uninformative elements, yielding a patch of $s(s_0 + 1)$ elements. (B) The s informative features can occupy any position within the patch of $s(s_0 + 1)$ elements. In both cases, all the possible rearrangements are shown for $s = 2$ and $s_0 = 1$. At the next production rule, each uninformative element generates an empty patch of $s(s_0 + 1)$ uninformative elements, as pictured in **(4)**. **(5)** Four data sampled from the generative hierarchical task shown in panel (1) with sparsity (A). The first two examples follow the rules in black boxes in panels (1) and (4), showcasing different feature rearrangements.

(Poggio et al., 2017). Sufficient information exists in a training set with size polynomial in d to reconstruct such tasks (Schmidt-Hieber, 2020)— although the training may take an immense time. Generative hierarchical models of data (Mossel, 2018; Malach & Shalev-Shwartz, 2018; 2020) can be learnt efficiently by iterative clustering methods, if correlations exist between input features and output. Deep architectures trained with gradient descent display a hierarchical representation of the data, corresponding to the latent variables in these models (Cagnetta et al., 2023b; Allen-Zhu & Li, 2023). Deep networks, as opposed to shallow ones, beat the curse of dimensionality, with a sample complexity that is polynomial in the dimension (Cagnetta et al., 2023b). However, none of these works consider sparsity in feature space, and the stability it confers to discretised smooth transformations of the input.

Task structure and invariance: In (Hupkes et al., 2021) a classification of different combinatorial properties of tasks is introduced. The SRHM displays the properties of systematicity, substitutivity and localism in that classification (Hupkes et al., 2021).

CNNs were crafted to have internal representations equivariant to certain transformations such as rotations (Cohen & Welling, 2014; 2016; Ensign et al., 2017; Kondor & Trivedi, 2018; Finzi et al., 2021; Batzner et al., 2022; Blum-Smith & Villar, 2023). How such a procedure can improve sample complexity has been addressed for linear models such as

random features and kernel methods (Favero et al., 2021; Bietti et al., 2021; Elesedy & Zaidi, 2021; Elesedy, 2021; Mei et al., 2021). Instead, our work focuses on how invariance emerges during training and how it affects sample complexity, in a regime where features are learnt.

Finally, in the context of adversarial robustness the response of trained networks to small-norm transformations of the input data that change the input label are investigated (Fawzi & Frossard, 2015; Kanbak et al., 2018; Alaifari et al., 2018; Athalye et al., 2018; Xiao et al., 2018; Alcorn et al., 2019; Engstrom et al., 2019). Our approach differs from this literature since we consider the response to generic perturbations belonging to specific ensembles, rather than worst-case perturbations.

2. Background: Hierarchical Generative Models

We consider hierarchical classification tasks where images are generated from class labels through a hierarchical composition of production rules (Mossel, 2018; Malach & Shalev-Shwartz, 2018; DeGiuli, 2019; Malach & Shalev-Shwartz, 2020; Cagnetta et al., 2023b; Allen-Zhu & Li, 2023). These tasks represent a specific case of context-free grammars, a generative model in formal language theory (Rozenberg & Salomaa, 1997). We focus on L-level context-free grammar, considering a set of class labels $\mathcal{C} \equiv \{1, \dots, n_c\}$ and L disjoint vocabularies $\mathcal{V}_\ell \equiv \left\{a_1^\ell, \dots, a_{v_\ell}^\ell\right\}$ of low- and high-level features, with $\ell = 1, ..., L$. Henceforth, we refer to $\ell > 1$ as high-level features or latent variables. Upon selecting a class label α uniformly at random from $\mathcal{C}$, the data is generated iteratively from a set of production rules:

$$\alpha \mapsto \mu_1^{(L)}, \dots, \mu_{s_L}^{(L)} \text{ for } \alpha \in \mathcal{C} \text{ and } \mu_i^{(L)} \in \mathcal{V}_L, \quad (1)$$

$$\mu^{(\ell)} \mapsto \mu_1^{(\ell-1)}, \dots, \mu_{s_\ell}^{(\ell-1)} \text{ for } \mu^{(\ell)} \in \mathcal{V}_\ell, \mu_i^{(\ell-1)} \in \mathcal{V}_{\ell-1}, \quad (2)$$

for $\ell = 2, ..., L$. At each level ℓ, we consider m_ℓ distinct rules stemming from each higher-level feature $a_i^{(\ell)}$. In other words, there are m_ℓ equivalent lower-level representations of $a_i^{(\ell)}$ for all $i = 1, ..., v$. These equivalent representations are termed 'synonyms'. In Figure 2, panel 1, we show on the top the generation rules of a generative hierarchical task with $n_c = 2$ classes with two levels of production rules, each counting 3 synonyms.

To simplify notation, we opt for the case of the Random Hierarchy Model (RHM) (Cagnetta et al., 2023b), for which: (i) $\forall \ell, v_\ell = v, m_\ell = m, s_\ell = s$, (ii) the m production rules associated with any latent variable or class, shown on top in the example of panel 1 in Figure 2, are randomly chosen among the v^s possibles ones and (iii) the m production rules of any latent variable or class are sampled uniformly during data generation, as exemplified on the bottom of panel 1 in Figure 2. Consequently, the total number of possible data produced per class is $m^{\frac{d-1}{s-1}}$, where the dimension d is defined as $d = s^L$. (iv) Two distinct classes or latent variables cannot yield the same low-level representation. This condition ensures, for example, that two distinct classes never generate the same data, implying that $m \leq v^{s-1}$. An example of a hierarchical generation process is shown in Figure 2, panel (1).

Note that (i) we represent each input level feature in $\mathcal{V}_1$ with a one-hot encoding of length v, yielding input data with dimension $d \times v$. The representation of higher-level features need not be specified, as they are latent variables. (ii) Although we focused on generating one-dimensional data patches, the same model can generate square 'images' without modifications if s is the square of a natural number.

The RHM can be used to generate P training points of input-label pairs (x, y) where $y = 1, .., n_c$ indicates the class and $x \in \mathbb{R}^{d \times v}$ is the input corresponding to d sub-features $\mu^{(1)}$, each one-hot encoded in a dimension v. Using such data to train deep networks to classify y from x, one finds (Cagnetta et al., 2023b) that:

- The sample complexity P^* of shallow network grows exponentially in d, whereas for deep networks it is polynomial in d and reads $P^* \propto n_c m^L$, with a prefactor of order unity for CNNs.
- The sample complexity corresponds also to the training set size at which a hierarchical neural representation emerges. The latent variables μ^ℓ are encoded closer to the output as ℓ increases. This representation is insensitive to change of synonyms in the input.
- Synonyms produced by the same high-level feature $\mu^{(2)}$ of the image share the same correlation with the output, a fact true for any given patch of size s in the input. When a sufficient number of data is present, i.e. $P \geq P^*$, these correlations are larger than the sampling noise given by the finitess of the training set. These correlations can be used to group synonyms together. Gradient descent can capture these correlations, constructing a representation invariant to synonyms exchange. These correlations between synonyms and output are necessary for the network to solve the task: if a hierarchical model is designed without them, learning is essentially impossible (it requires an exponential number of data in the dimension)(Cagnetta et al., 2023b).

3. Sparsity and Stability to Diffeomorphisms

Our key insight is that spatial sparsity of features implies stability to diffeomorphisms. This point is illustrated using

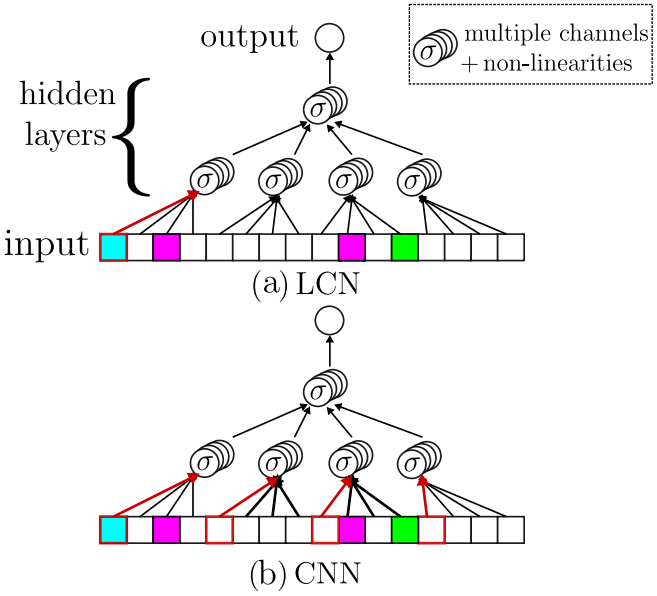

Figure 3. **Networks:** (a) Locally Connected Network (LCN). Each neuron's weight focuses on a single input element (in red). Networks have L hidden layers, with filters matching patches of size $s(s_0+1)$ from the generative process in Figure 2 (here $L=2$, $s=2$, $s_0=1$). A last fully connected layer connects the output of the last local layer with the output. (b) Convolutional Neural Network (CNN) with the structure of (a), featuring weight sharing such that each weight considers different pixels in all patches of size $s(s_0+1)$ (in red).

a sketched dog in Figure 2, panel (2): if a few lines can define what is on a drawing, then small changes in the relative distances between these lines should not alter the class label.

This idea can be readily implemented in generative models by adding an 'uninformative' feature 0 to each vocabulary $\mathcal{V}_\ell$ and imposing, for example, the constraint that each production rule in Eq. 1 and Eq. 2 contains exactly $s \times s_0$ uninformative features. We implement sparsity in two ways, as shown in Figure 2, panel (3). (A) Each of the s informative features is embedded in a sub-patch of size (s_0+1) with strictly s_0 empty elements. The position of each informative feature is independent of the other feature positions. (B) The s informative features can occupy any position within the patch of $s(s_0+1)$ elements, as long as their order remains the same. For both (A) and (B), at each level of the hierarchy, each uninformative feature will produce a patch of $s(s_0+1)$ uninformative features at the next level, as depicted in Figure 2, panel (4). We denote model A as the Sparse Random Hierarchical Model (SRHM).

Note that (i) these processes generate very sparse data, with just s^L informative features randomly placed in inputs of dimension $d=(s(s_0+1))^L$, as shown for a few examples in Figure 2, panel (5). The informative features are represented with one-hot encoding with dimension v, while uninformative pixels are represented by empty columns, yielding input with size $d \times v$. (ii) Sparsity implies that some transformation leaves the task invariant. For (A), the transformations that do not affect the task include the motion of the s informative features μ^1 within patches of size (s_0+1). For (B), any motion of the set of informative low-level features μ^1 that leaves their order unchanged (as diffeomorphisms would do) does not alter the label, as illustrated in Figure 2, panel (3).

4. Sample Complexity

We empirically analyze the number of training points required to learn the SRHM for both CNNs and for Locally Connected Networks (LCNs), a version of CNNs without weight sharing (Fukushima, 1975; le Cun, 1989; Favero et al., 2021). In (Cagnetta et al., 2023b), it is shown that in the absence of sparsity, the sample complexity mildly depends on the architecture, as long as it is deep enough–even for fully connected networks. Here, we start by restricting ourselves to architectures that match the generative process. Specifically, the LCN architecture has L hidden layers with filter size and stride both equal to $s(s_0+1)$, followed by a linear readout layer, as shown in Figure 3 (a). The CNNs we consider are structured as the LCNs, but they implement weight sharing, as depicted in Figure 3 (b). By comparing the sample complexities of LCNs and CNNs we quantify the improvement achieved through the implementation of weight sharing. For details about the training scheme, we refer to Appendix D. How common architectures such as VGG, ResNet, and EfficientNet learn the SRHM is investigated in Appendix B.

We utilize the SRHM to generate P training points, corresponding to input-label pairs (x,y) where $y=1,..,n_c$ and $x \in \mathbb{R}^{d \times v}$. An example of a learning curve representing the test error ϵ dependence on training set size P is shown in Figure 4 (A). The test error remains large with increasing P until it decays rapidly to near-zero values at a certain scale P^*, corresponding to the sample complexity. To estimate it, we fix a threshold (e.g. 0.1) and measure the minimal number of training points P^* at which the test error achieves such a threshold. Modifying the threshold value does not qualitatively alter our results below. We focus on sparsity A, depicted in Figure 2 panel 3, as it is easier to analyze than the sparsity B. We show in Appendix A that they lead to the same sample complexity.

We systematically apply this procedure to extract the sample complexity P^* as the parameters s, s_0, v and L are changed while keeping the number of informative synonyms $m=v^{s-1}$ to its maximal value.

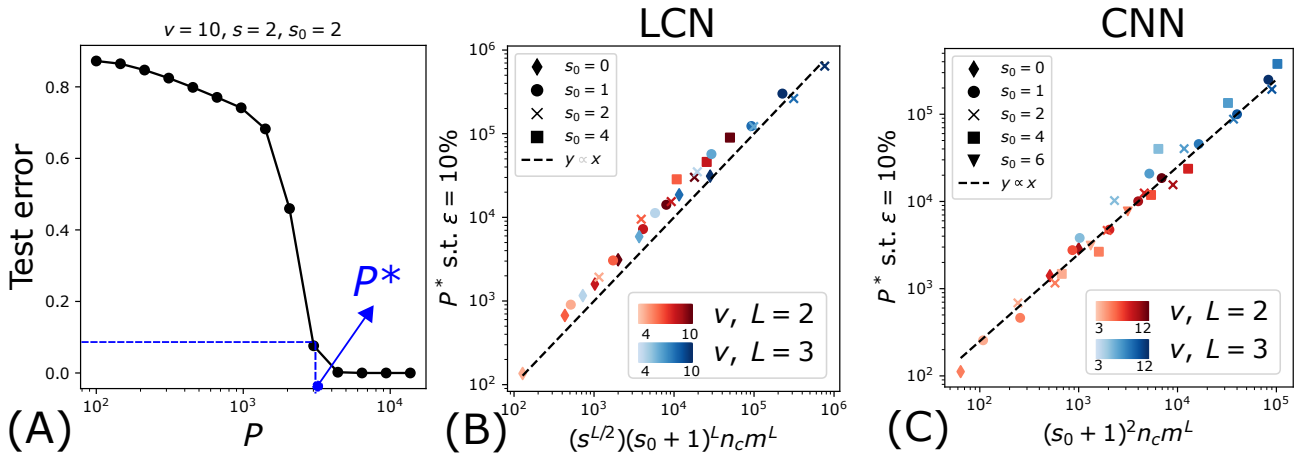

Figure 4. (A) Test error $\varepsilon(P)$ versus number of training points P. To extract the sample complexity P^*, we fix an arbitrary threshold $\varepsilon^* = \varepsilon(P^*)$. Here $\varepsilon^* = 10\%$. (B) Empirical sample complexity P^* for a LCN to reach a 10% test error ε versus estimation of Eq. 3 for $s = 2$, different depths L (red for $L = 2$, blue for $L = 3$), different vocabulary sizes v (different darkness), number of classes $n_c = v$, maximal $m = v^{s-1}$ and different s_0 (different markers). (C) Same as (B) for CNNs, supporting Eq. 4. Further support for Eq. 3 and Eq. 4 is obtained by varying s, as shown in Appendix D, Figure 10 and Appendix E, Figure 13.

For the LCNs, the results shown in Figure 4 (B) and Figure 10 in Appendix D indicate that:

$$P^*_{\text{LCN}} \sim C_0(s, L)(s_0 + 1)^L n_c m^L. \tag{3}$$

where our observations for $C_0(s, L)$ are consistent with $C_0(s, L) \sim s^{L/2}$. We will motivate the dependence of P^*_{LCN} with s_0 in section 6.

For CNNs, we observe in Figure 4 (C) and Figure 13 in Appendix E that the sample complexity follows:

$$P^*_{\text{CNN}} \sim C_1(s_0 + 1)^2 n_c m^L. \tag{4}$$

where C_1 is a constant.

The relations Eq. 3 and Eq. 4 are central to this work, as they specify how sparsity and combinatorial properties of the data affect the sample complexity. Remarkably, both sample complexities are exponential in L and thus polynomial in the input dimension $d = (s(s_0 + 1))^L$, effectively beating the curse of dimensionality. Weight sharing proves to be beneficial for networks trained on the SRHM, since the sample complexity is only quadratic in L in $(s_0 + 1)$ for CNNs, while it is exponential in L for LCNs. We will argue why this is the case in section 6.

Benefit of Sparsity. We show that for locally connected nets, for reasonable assumptions and for a fixed dimension d, a higher sparsity is preferable as it leads to a smaller sample complexity. Introducing the fraction of informative 'pixels' in the image $F = (s_0 + 1)^{-L}$ as a measure of sparsity, we can reformulate the sample complexity P^*_{LCN} in terms of the input dimension d and image relevant fraction F, assuming that m and s are fixed but letting L and s_0 change. One obtains:

$$P^*_{\text{LCN}} \sim F^{\frac{\log m}{\log s} - \frac{1}{2}} d^{\frac{\log m}{\log s} + \frac{1}{2}}, \tag{5}$$

which is indeed a growing function of F, as long as $m > \sqrt{s}$. This result is illustrated in Figure 5 for the case where m takes its maximum value $m = v^{s-1}$. This indicates that neural networks can adapt to the sparsity of the task.

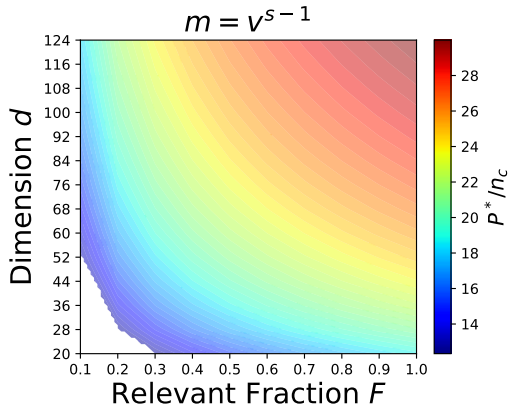

Figure 5. Sample complexity of LCN learning the SRHM for varying input dimension d and input relevant fraction F at the maximal case $m = v^{s-1}$, with $v = 10$ and $s = 5$, according to Eq. 5. The color map is in log scale. At fixed dimension d, a smaller F (hence higher sparsity) makes the task easier.

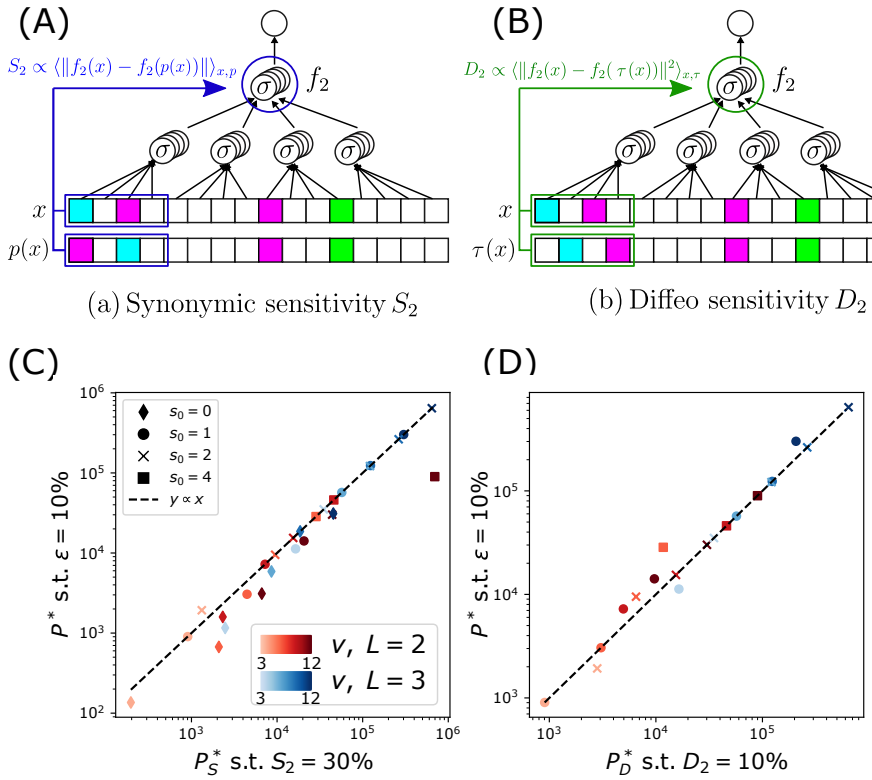

Figure 6. **Top.** Procedure to compute the sensitivity S_2 and D_2, defined in Eq. 6 and Eq. 7, illustrated for $L = 2$. We take a two-layer network, trained with P training points on the sparse hierarchical dataset with $L = 2$ levels. We apply either a (A) synonyms exchange p or a (B) diffeomorphism τ on the input data x. Note that in (A) p changes the features but not their position while in (B) τ changes the position of the features but not their value. Then we test the second net layer f_2 sensitivity to these transformations. **Bottom.** (C) Empirical sample complexity P^* to reach a 10% test error ε versus empirical sample complexity P_S^* to reach $S_2 = 30\%$ for $s = 2$, different depths L (red for $L = 2$, blue for $L = 3$), different vocabulary sizes v (different darkness), number of classes $n_c = v$, maximal $m = v^{s-1}$ and different s_0 (different markers). (D) Same as (C), for empirical sample complexity P^* to reach a 10% test error ε versus empirical sample complexity P_D^* to reach $D_2 = 10\%$. The sensitivity thresholds have been tuned based on the form of S_2 and D_2 versus P, reported in Appendix D, Figure 12 for $L = 2$. Both P_S^* and P_D^* are nearly equal to P^*.

5. Learning Invariant Representation

In section 4, we have shown that deep networks trained on the SRHM beat the curse of dimensionality, learning the task with a sample complexity polynomial in the input dimension. Here, we show that they manage to do so by learning representations that are insensitive to irrelevant aspects of the task.

The first invariance of the SRHM task corresponds to the fact that different combinations of s informative features are synonyms: to solve the task, it is not necessary to carry within the network the information of which actual synonym was present in the input. The second invariant considers transformations akin to diffeomorphisms, which shift the position of the informative features $\mu^{(1)}$. Formally, we measure such invariant by defining two operators acting on the input:

1. Synonymic exchange operator p. This operator takes in a datum x and substitutes each informative $s-$patch of features $\mu^{(1)}$, produced by a given $\mu^{(2)}$, with one of its $m-1$ synonyms, chosen uniformly at random, keeping the feature positions intact. Figure 6 (A) illustrates the action of p. The sensitivity of the representation at a given hidden layer f_k to the action of p is measured by the quantity S_k:

$$S_k = \frac{\langle ||f_k(x) - f_k(p(x))||^2 \rangle_{x,p}}{\langle ||f_k(x_1) - f_k(x_2))||^2 \rangle_{x_1,x_2}}, \qquad (6)$$

where x, x_1, and x_2 belong to a test set, and the average $\langle . \rangle$ is over the test set and on the random exchange of synonyms p. A visualization of S_k for $k = 2$ for a two-layer network is shown in Figure 6 (A). If the internal representation f_k has learnt the synonyms associated with production rule Eq. 2 at level $\ell = 2$, then S_k is small. The sensitivity of the network output to p, shown in Figure 1 (D, F), is defined similarly to (6).

2. A discretised diffeomorphism operator τ. This operator takes as input a datum x and randomly shifts its informative features according to the possible positions obtained when the features $\mu^{(2)}$ generate patches of features μ^1, as illustrated in Figure 2, panel (3). In this process, the nature of the feature remains the same. In Figure 6 (B) the action of τ on a datum x is shown. The sensitivity D_k of f_k to diffeomorphisms τ is defined as:

$$D_k = \frac{\langle ||f_k(x) - f_k(\tau(x))||^2 \rangle_{x,\tau}}{\langle ||f_k(x_1) - f_k(x_2))||^2 \rangle_{x_1,x_2}}. \quad (7)$$

If f_k has learnt the invariance to diffeomorphisms, then D_k is zero. This definition of sensitivity is akin to what is used for images (Petrini et al., 2021). The sensitivity of the network output f to τ, shown in Figure 1 (C, E), is defined analogously to (7).

It is possible to generalize the sensitivities Eq. 6 and Eq. 7 to measure how much the representation at any hidden layer k of a network, or its output, is sensitive to exchange of synonyms or diffeomorphisms related to the features $\mu^{(\ell)}$ produced by the latent $\mu^{(\ell+1)}$. We show in Appendix D and Appendix E that the representation at layer $k \geq 2$ becomes insensitive to transformations applied to levels $\ell \leq k-1$ of the hierarchy, for training set size $P \gg P^*$. At that point, the output becomes insensitive to transformations applied at any level ℓ of the hierarchy.

Here, we focus on S_2 and D_2 (corresponding to the case $k = 2$ and $\ell = 1$). We measure how these quantities depend on the size of the training set, as shown in Appendix D, Figure 12, and in Appendix E, Figure 15, for $L = 2$. Subsequently, we define sample complexities P_S^* or P_D^* associated with S_2 and D_2, using a similar procedure as for the test error: we measure the training set size where these quantities reach some threshold value.

Figure 6 (C) and (D) present our key result for LCNs:

$$P_S^* \approx P_{\text{LCN}}^*, \quad P_D^* \approx P_{\text{LCN}}^*, \quad (8)$$

further supported by Figure 11 in Appendix D for a different value of s. In essence, the insensitivities to diffeomorphisms and synonyms exchange are acquired for the same training set size, precisely when the network learns the task. This observation appears to be universal, as it holds for common convolutional architectures like VGG, ResNet, and EfficientNet as demonstrated in Appendix B, Figure 9, for the simple convolutional architectures pictured in Figure 3 (b) in Appendix E, Figure 14 and even for fully-connected networks in Appendix F, Figure 16.

Therefore, *deep networks learn to be insensitive to diffeomorphisms as they learn a hierarchical representation*, which is thought to be crucial to achieve high performance. This central result rationalizes the observed correlation between sensitivity to diffeomorphisms of the network output (thus indicative of building a hierarchical representation of the data,) and test error displayed in Figure 1 (A) for CIFAR10 and (C) in the SRHM.

The relationship between insensitivity to synonyms and to diffeomorphisms also appears when considering how these quantities evolve with the size of the training set P, as documented in Figure 1 (B, E, F). The results shown in Figure 1 (C,D,E,F) for the network output hold also for the sensitivities of internal layers, as shown in Figure 17 in Appendix G.

6. Sample Complexities Arguments

In the absence of sparsity $s_0 = 0$, the sample complexity essentially corresponds to the training set size necessary for detecting synonyms (Cagnetta et al., 2023b). Following this result, we present a simple heuristic argument for the sample complexity of the LCN architecture in the sparse case $s_0 > 0$. Crucially, this argument also explains why invariance to synonyms and to smooth transformations are learnt together.

- Any given 2-level latent variable $\mu^{(2)}$ closest to the input can generate different synonyms, distorted with different diffeomorphisms due to sparsity, as in Figure 2, panel (3).
- All these different representations produced by that latent variable $\mu^{(2)}$ have the same correlation with the class label.
- This correlation can be used to group together the representations produced by $\mu^{(2)}$, if the training set is larger than some sample complexity. We can estimate this sample complexity for LCNs as follows. A single weight connected to the input will not see any information for most data, as it will most often see the "pixel" or low-level feature $\mu^{(1)} = 0$, which is uninformative. With respect to the case $s_0 = 0$, the fraction of data with local information is diminished by a factor $(s_0+1)^{-L}$. To detect correlations beyond sampling noise, the sample complexity has thus to be increased by a factor $(s_0+1)^L$, as empirically observed in Eq. 3. We show in Appendix C that at this sample complexity a single step of GD can aggregate the equivalent representations produced by $\mu^{(2)}$.
- The hidden representation in the first hidden layer thus becomes insensitive to diffeomorphisms and synonyms at the same sample complexity, as shown in Figure 6 (C, D). Once the representations that encode for features $\mu^{(2)}$ have been detected, the problem is much simpler, corresponding to a generative model with $L-1$ levels

instead of L. Thus, representations of higher production rules μ^ℓ, $\ell > 2$ are also detected and represented together in the higher layers of the network beyond that characteristic training set size. Indeed, synonyms and insensitivity to diffeomorphisms are learnt at all levels of the hierarchy in one go, as illustrated in Appendix D, Figure 12.

Qualitatively, the same scenario holds for CNNs. One expects a different sample complexity since each weight is now connected to a fraction of the input that is independent of L. Yet, the quadratic dependence on $s_0 + 1$ remains to be understood.

7. Limitations

We predicted that good performance, insensitivity to diffeomorphism, and insensitivity to synonyms exchange should occur concomitantly as the training set size is increased. However to test this prediction on benchmark image datasets, we are currently missing an empirical procedure to measure insensitivity to synonyms. To perform such a measurement, one needs to be able to change what composes a given image at various scales. The recent result of (Sclocchi et al., 2024) suggest that change of low-level features can be obtained using diffusion-based generative models, and that the scale where the composition of the image changes is controlled by the magnitude of the noise that enters in these methods. It would be interesting to use this result to test our prediction. Another approach would be to test stability to actual synonyms in text data, and quantify how this property correlates with the performance of natural language processing systems.

8. Conclusions

Understanding the nature of real data is an elusive problem yet central to many quantitative questions of the field. We have unified in a common framework two different approaches to this problem: the first assumes the data to be combinatorial and hierarchical, while the second emphasizes the task insensitivity with respect to smooth transformations, relevant for example for image datasets. This framework was obtained by introducing models that display both properties, based on the notion that sparsity of informative features in space naturally endows stability to smooth transformations. These models explain the strong empirical correlation between stability to smooth transformations and performance, and further predict that both quantities should correlate with the insensitivity to discrete changes of low-level features in the data.

Finally, although we focused on classification problems, the generative models we introduced display a rich structure in the data distribution $P(x)$ itself, which is not only hierarchical but is invariant to smooth transformations. These models are thus promising to understand how unsupervised methods beat the curse of dimensionality, by composing or deforming features they have learnt from examples.

Impact Statement

This paper presents work whose goal is to advance the field of Machine Learning. There are many potential societal consequences of our work, none of which we feel must be specifically highlighted here.

Acknowledgements

The authors thank Francesco Cagnetta, Alessandro Favero, Leonardo Petrini and Antonio Sclocchi for fruitful discussions and helpful feedback on the manuscript. This work was supported by a grant from the Simons Foundation (#454953 Matthieu Wyart).

References

Abbe, E., Boix-Adsera, E., Brennan, M. S., Bresler, G., and Nagaraj, D. The staircase property: How hierarchical structure can guide deep learning. In Ranzato, M., Beygelzimer, A., Dauphin, Y., Liang, P., and Vaughan, J. W. (eds.), Advances in Neural Information Processing Systems, volume 34, pp. 26989–27002. Curran Associates, Inc., 2021.

Alaifari, R., Alberti, G. S., and Gauksson, T. ADef: an Iterative Algorithm to Construct Adversarial Deformations. September 2018. URL https://openreview.net/forum?id=Hk4dFjR5K7.

Alcorn, M. A., Li, Q., Gong, Z., Wang, C., Mai, L., Ku, W.-S., and Nguyen, A. Strike (With) a Pose: Neural Networks Are Easily Fooled by Strange Poses of Familiar Objects. In 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 4840–4849, Long Beach, CA, USA, June 2019. IEEE. ISBN 978-1-72813-293-8. doi: 10.1109/CVPR.2019.00498. URL https://ieeexplore.ieee.org/document/8954212/.

Allen-Zhu, Z. and Li, Y. How can deep learning performs deep (hierarchical) learning, 2023. URL https://openreview.net/forum?id=j2ymLjCr-Sj.

Athalye, A., Engstrom, L., Ilyas, A., and Kwok, K. Synthesizing Robust Adversarial Examples. In International Conference on Machine Learning, pp. 284–293. PMLR, July 2018. URL http://proceedings.mlr.press/v80/athalye18b.html. ISSN: 2640-3498.

Azulay, A. and Weiss, Y. Why do deep convolutional networks generalize so poorly to small image transformations? arXiv preprint arXiv:1805.12177, 2018.

Bach, F. Breaking the curse of dimensionality with convex neural networks. The Journal of Machine Learning Research, 18(1):629–681, 2017.

Batzner, S., Musaelian, A., Sun, L., Geiger, M., Mailoa, J. P., Kornbluth, M., Molinari, N., Smidt, T. E., and Kozinsky, B. E(3)-equivariant graph neural networks for data-efficient and accurate interatomic potentials. Nature Communications, 13(1):2453, May 2022. ISSN 2041-1723. doi: 10.1038/s41467-022-29939-5. URL https://doi.org/10.1038/s41467-022-29939-5.

Bietti, A. Approximation and learning with deep convolutional models: a kernel perspective, 2022.

Bietti, A., Venturi, L., and Bruna, J. On the sample complexity of learning under invariance and geometric stability, 2021.

Bietti, A., Bruna, J., Sanford, C., and Song, M. J. Learning single-index models with shallow neural networks, 2022.

Blum-Smith, B. and Villar, S. Machine learning and invariant theory, 2023. URL https://arxiv.org/abs/2209.14991.

Bommasani, R., Hudson, D. A., Adeli, E., and et al. On the opportunities and risks of foundation models, 2022.

Brown, T. B., Mann, B., Ryder, N., and et al. Language models are few-shot learners, 2020.

Bruna, J. and Mallat, S. Invariant scattering convolution networks. IEEE transactions on pattern analysis and machine intelligence, 35(8):1872–1886, 2013.

Cagnetta, F., Favero, A., and Wyart, M. What can be learnt with wide convolutional neural networks? In International Conference on Machine Learning, pp. 3347–3379. PMLR, 2023a.

Cagnetta, F., Petrini, L., Tomasini, U. M., Favero, A., and Wyart, M. How deep neural networks learn compositional data: The random hierarchy model, 2023b. URL https://arxiv.org/abs/2307.02129.

Castleman, K. R. Digital image processing. Prentice Hall Press, 1996.

Cohen, T. and Welling, M. Learning the irreducible representations of commutative lie groups. In Xing, E. P. and Jebara, T. (eds.), Proceedings of the 31st International Conference on Machine Learning, volume 32 of Proceedings of Machine Learning Research, pp. 1755–1763, Bejing, China, 22–24 Jun 2014. PMLR. URL https://proceedings.mlr.press/v32/cohen14.html.

Cohen, T. and Welling, M. Group equivariant convolutional networks. In Balcan, M. F. and Weinberger, K. Q. (eds.), Proceedings of The 33rd International Conference on Machine Learning, volume 48 of Proceedings of Machine Learning Research, pp. 2990–2999, New York, New York, USA, 20–22 Jun 2016. PMLR. URL https://proceedings.mlr.press/v48/cohenc16.html.

Cortes, C., Jackel, L. D., Solla, S., Vapnik, V., and Denker, J. Learning curves: Asymptotic values and rate of convergence. Advances in neural information processing systems, 6, 1993.

DeGiuli, E. Random language model. Physical Review Letters, 122(12):128301, 2019.

Dieleman, S., De Fauw, J., and Kavukcuoglu, K. Exploiting cyclic symmetry in convolutional neural networks. arXiv preprint arXiv:1602.02660, 2016.

Doimo, D., Glielmo, A., Ansuini, A., and Laio, A. Hierarchical nucleation in deep neural networks. Advances in Neural Information Processing Systems, 33:7526–7536, 2020.

Elesedy, B. Provably strict generalisation benefit for invariance in kernel methods. In Ranzato, M., Beygelzimer, A., Dauphin, Y., Liang, P., and Vaughan, J. W. (eds.), Advances in Neural Information Processing Systems, volume 34, pp. 17273–17283. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper_files/paper/2021/file/8fe04df45a22b63156ebabbb064fcd5e-Paper.pdf.

Elesedy, B. and Zaidi, S. Provably strict generalisation benefit for equivariant models. In Meila, M. and Zhang, T. (eds.), Proceedings of the 38th International Conference on Machine Learning, volume 139 of Proceedings of Machine Learning Research, pp. 2959–2969. PMLR, 18–24 Jul 2021. URL https://proceedings.mlr.press/v139/elesedy21a.html.

Engstrom, L., Tran, B., Tsipras, D., Schmidt, L., and Madry, A. Exploring the Landscape of Spatial Robustness. In International Conference on Machine Learning, pp. 1802–1811. PMLR, May 2019. URL http://proceedings.mlr.press/v97/engstrom19a.html. ISSN: 2640-3498.

Ensign, D., Neville, S., Paul, A., and Venkatasubramanian, S. The complexity of explaining neural networks through (group) invariants. In Hanneke, S. and Reyzin, L. (eds.),

Proceedings of the 28th International Conference on Algorithmic Learning Theory, volume 76 of Proceedings of Machine Learning Research, pp. 341–359. PMLR, 15–17 Oct 2017. URL https://proceedings.mlr.press/v76/ensign17a.html.

Favero, A., Cagnetta, F., and Wyart, M. Locality defeats the curse of dimensionality in convolutional teacher-student scenarios. Advances in Neural Information Processing Systems, 34:9456–9467, 2021.

Fawzi, A. and Frossard, P. Manitest: Are classifiers really invariant? In Procedings of the British Machine Vision Conference 2015, pp. 106.1–106.13, Swansea, 2015. British Machine Vision Association. ISBN 978-1-901725-53-7. doi: 10.5244/C.29.106. URL http://www.bmva.org/bmvc/2015/papers/paper106/index.html.

Finzi, M., Welling, M., and Wilson, A. G. G. A practical method for constructing equivariant multilayer perceptrons for arbitrary matrix groups. In Meila, M. and Zhang, T. (eds.), Proceedings of the 38th International Conference on Machine Learning, volume 139 of Proceedings of Machine Learning Research, pp. 3318–3328. PMLR, 18–24 Jul 2021. URL https://proceedings.mlr.press/v139/finzi21a.html.

Fukushima, K. Cognitron: A self-organizing multilayered neural network. Biological Cybernetics, 20(3): 121–136, September 1975. ISSN 1432-0770. doi: 10.1007/BF00342633. URL https://doi.org/10.1007/BF00342633.

Grenander, U. Elements of pattern theory. JHU Press, 1996.

He, K., Zhang, X., Ren, S., and Sun, J. Deep Residual Learning for Image Recognition. IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 770–778, June 2016. doi: 10.1109/CVPR.2016.90.

Hestness, J., Narang, S., Ardalani, N., Diamos, G., Jun, H., Kianinejad, H., Patwary, M., Ali, M., Yang, Y., and Zhou, Y. Deep learning scaling is predictable, empirically. arXiv preprint arXiv:1712.00409, 2017.

Ho, J., Jain, A., and Abbeel, P. Denoising diffusion probabilistic models. Advances in neural information processing systems, 33:6840–6851, 2020.

Hupkes, D., Dankers, V., Mul, M., and Bruni, E. Compositionality decomposed: how do neural networks generalise? (extended abstract). In Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI'20, 2021. ISBN 9780999241165.

Kanbak, C., Moosavi-Dezfooli, S.-M., and Frossard, P. Geometric Robustness of Deep Networks: Analysis and Improvement. In 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 4441–4449, Salt Lake City, UT, June 2018. IEEE. ISBN 978-1-5386-6420-9. doi: 10.1109/CVPR.2018.00467. URL https://ieeexplore.ieee.org/document/8578565/.

Kaplan, J., McCandlish, S., Henighan, T., Brown, T. B., Chess, B., Child, R., Gray, S., Radford, A., Wu, J., and Amodei, D. Scaling laws for neural language models. arXiv preprint arXiv:2001.08361, 2020.

Kayhan, O. S. and Gemert, J. C. v. On translation invariance in cnns: Convolutional layers can exploit absolute spatial location. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 14274–14285, 2020.

Kondor, R. and Trivedi, S. On the generalization of equivariance and convolution in neural networks to the action of compact groups. In Dy, J. and Krause, A. (eds.), Proceedings of the 35th International Conference on Machine Learning, volume 80 of Proceedings of Machine Learning Research, pp. 2747–2755. PMLR, 10–15 Jul 2018. URL https://proceedings.mlr.press/v80/kondor18a.html.

le Cun, Y. Generalization and network design strategies. In Connectionism in perspective, volume 19, pp. 143–155, 1989. URL https://api.semanticscholar.org/CorpusID:244797850.

Lecun, Y., Bottou, L., Bengio, Y., and Haffner, P. Gradient-based learning applied to document recognition. Proceedings of the IEEE, 86(11):2278–2324, November 1998. ISSN 1558-2256. doi: 10.1109/5.726791. Conference Name: Proceedings of the IEEE.

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

Luxburg, U. v. and Bousquet, O. Distance-based classification with lipschitz functions. The Journal of Machine Learning Research, 5(Jun):669–695, 2004.

Malach, E. and Shalev-Shwartz, S. A provably correct algorithm for deep learning that actually works, 2018.

Malach, E. and Shalev-Shwartz, S. The implications of local correlation on learning some deep functions. Advances in Neural Information Processing Systems, 33:1322–1332, 2020.

Mallat, S. Understanding deep convolutional networks. Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences, 374 (2065):20150203, 2016.

Mei, S., Misiakiewicz, T., and Montanari, A. Learning with invariances in random features and kernel models. In Belkin, M. and Kpotufe, S. (eds.), Proceedings of Thirty Fourth Conference on Learning Theory, volume 134 of Proceedings of Machine Learning Research, pp. 3351–3418. PMLR, 15–19 Aug 2021. URL https://proceedings.mlr.press/v134/mei21a.html.

Mei, S., Misiakiewicz, T., and Montanari, A. Generalization error of random feature and kernel methods: hypercontractivity and kernel matrix concentration. Applied and Computational Harmonic Analysis, 59:3–84, 2022.

Mossel, E. Deep learning and hierarchal generative models, 2018.

Petrini, L., Favero, A., Geiger, M., and Wyart, M. Relative stability toward diffeomorphisms indicates performance in deep nets. Advances in Neural Information Processing Systems, 34:8727–8739, 2021.

Poggio, T., Mhaskar, H., Rosasco, L., Miranda, B., and Liao, Q. Why and when can deep-but not shallow-networks avoid the curse of dimensionality: a review. International Journal of Automation and Computing, 14(5):503–519, 2017.

Rozenberg, G. and Salomaa, A. Handbook of Formal Languages. Springer, January 1997. doi: 10.1007/978-3-642-59126-6.

Ruderman, A., Rabinowitz, N. C., Morcos, A. S., and Zoran, D. Pooling is neither necessary nor sufficient for appropriate deformation stability in CNNs. arXiv:1804.04438 [cs, stat], May 2018. URL http://arxiv.org/abs/1804.04438. arXiv: 1804.04438.

Schmidt-Hieber, J. Nonparametric regression using deep neural networks with relu activation function. The Annals of Statistics, 48(4):1875–1897, 2020.

Sclocchi, A., Favero, A., and Wyart, M. A phase transition in diffusion models reveals the hierarchical nature of data, 2024. URL https://arxiv.org/abs/2402.16991.

Simonyan, K. and Zisserman, A. Very Deep Convolutional Networks for Large-Scale Image Recognition. ICLR, 2015.

Spigler, S., Geiger, M., and Wyart, M. Asymptotic learning curves of kernel methods: empirical data versus teacher–student paradigm. Journal of Statistical Mechanics: Theory and Experiment, 2020(12):124001, December 2020. ISSN 1742-5468. doi: 10.1088/1742-5468/abc61d. URL https://doi.org/10.1088/1742-5468/abc61d. Publisher: IOP Publishing.

Tan, M. and Le, Q. EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. In International Conference on Machine Learning, pp. 6105–6114. PMLR, May 2019. URL http://proceedings.mlr.press/v97/tan19a.html. ISSN: 2640-3498.

Tomasini, U. M., Petrini, L., Cagnetta, F., and Wyart, M. How deep convolutional neural networks lose spatial information with training. Machine Learning: Science and Technology, 4(4):045026, nov 2023. doi: 10.1088/2632-2153/ad092c. URL https://dx.doi.org/10.1088/2632-2153/ad092c.

Voulodimos, A., Doulamis, N., Doulamis, A., and Protopapadakis, E. Deep learning for computer vision: A brief review. Computational Intelligence and Neuroscience, pp. 1–13, 2018.

Xiao, C., Zhu, J.-Y., Li, B., He, W., Liu, M., and Song, D. Spatially Transformed Adversarial Examples. February 2018. URL https://openreview.net/forum?id=HyydRMZC-.

Xiao, L. Eigenspace restructuring: a principle of space and frequency in neural networks. In Conference on Learning Theory, pp. 4888–4944. PMLR, 2022.

Xiao, L. and Pennington, J. Synergy and symmetry in deep learning: Interactions between the data, model, and inference algorithm. In Chaudhuri, K., Jegelka, S., Song, L., Szepesvari, C., Niu, G., and Sabato, S. (eds.), Proceedings of the 39th International Conference on Machine Learning, volume 162 of Proceedings of Machine Learning Research, pp. 24347–24369. PMLR, 17–23 Jul 2022. URL https://proceedings.mlr.press/v162/xiao22a.html.

Zeiler, M. D. and Fergus, R. Visualizing and understanding convolutional networks. In Computer Vision – ECCV 2014, Lecture Notes in Computer Science, pp. 818–833, 2014.

Zhang, R. Making convolutional networks shift-invariant again. arXiv preprint arXiv:1904.11486, 2019.

Appendix

A. Sparsity B

In this Appendix, we show support for the robustness of the sample complexities Eq. 3 and Eq. 4 to the choice of the sparsity in Figure 2, panel (3). Indeed, we observe in Figure 7 and Figure 8 that the learning curves obtained implementing Sparsity B in the data are consistent with Eq. 3 and Eq. 4, obtained from data generated with Sparsity A.

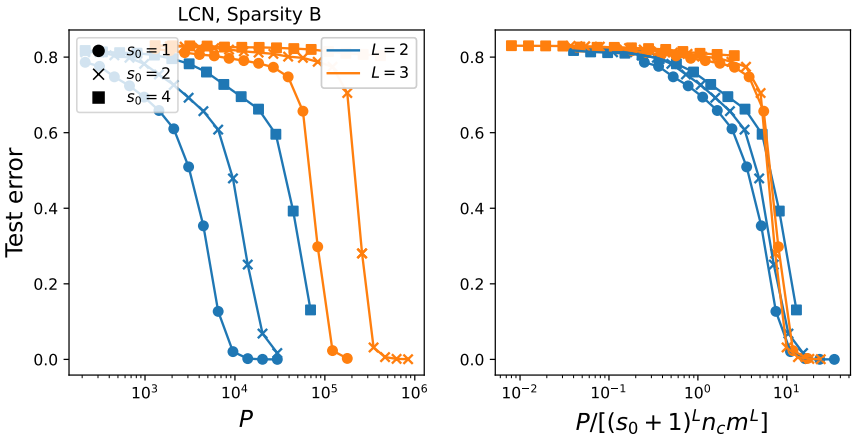

Figure 7. Left: test error versus number of training points P of LCN trained on the SRHM with sparsity B (described in Figure 2, panel (3)), with different L (different colors), different s_0 (different markers), $s = 2$ and $m = v = n_c = 6$. Right: same as left but rescaling P by Eq. 3.

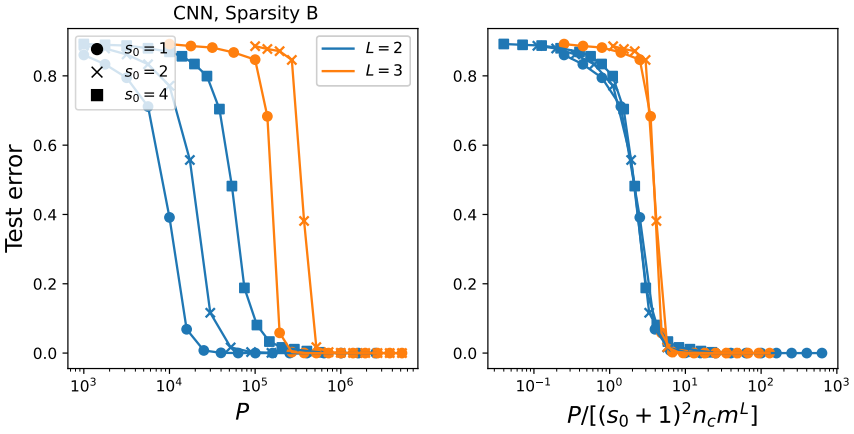

Figure 8. Left: test error versus number of training points P of CNN trained on the SRHM with sparsity B (described in Figure 2, panel (3)), with different L (different colors), different s_0 (different markers), $s = 2$ and $m = v = n_c = 6$. Right: same as left but rescaling P by prediction Eq. 4.

B. Common architectures learning the SRHM

Architectures. All networks implementations can be found at github.com/leonardopetrini/diffeo-sota/tree/main/models. In Table 1, adapted with permission of the authors from (Petrini et al., 2021), we list the structural choices of those networks.

Training scheme. We use Stochastic Gradient Descent (SGD) as optimizer, with batch size 4 and momentum 0.9. The learning rate lr has been optimized by extensive grid search, finding: $lr = 10^{-4}$. We use as train loss the cross entropy loss, and we stop the training when it reaches a threshold of 10^{-2}.

Figures. In Figure 9 we report the test error and the sensitivities to input transformations of 5 different common convolutional networks with respect to the number of training points P. The sensitivities are computed for 3 different layers at different relative depths.

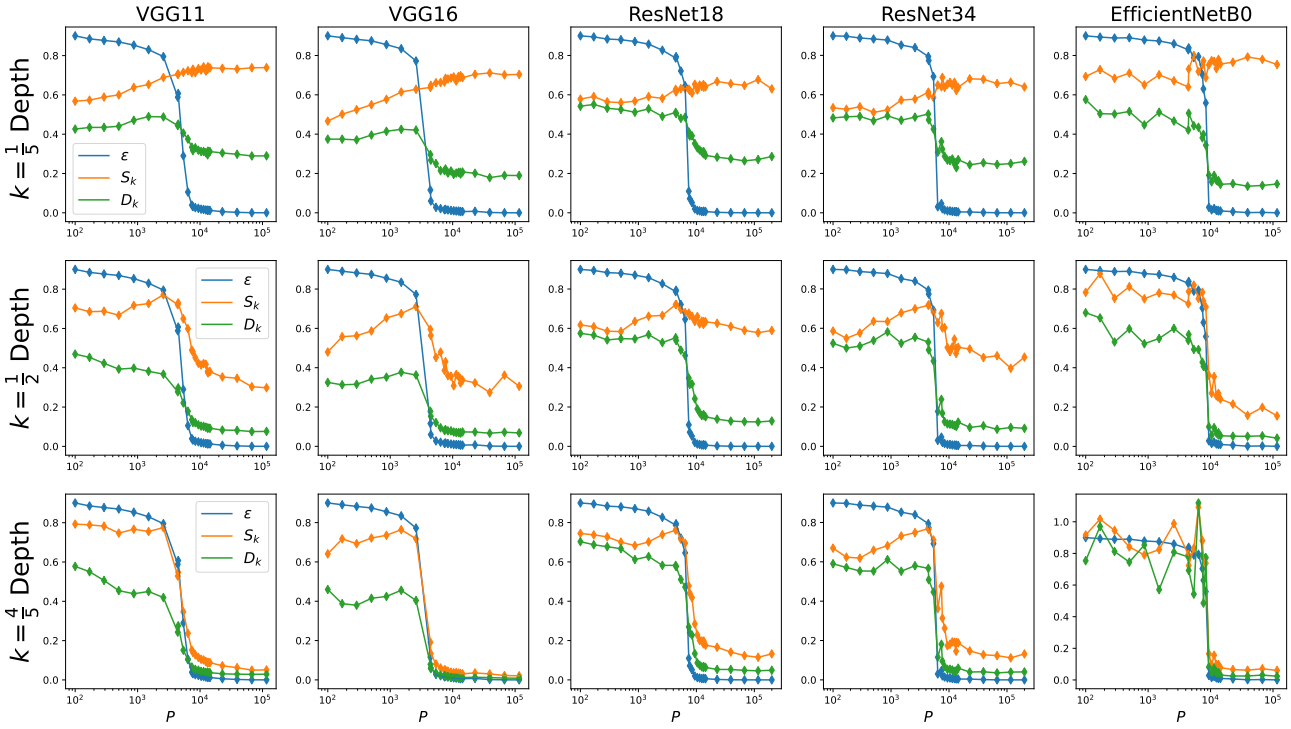

Figure 9. Test error (in blue), sensitivity to synonymic exchange (in orange), sensitivity to diffeomorphisms (in green) for increasing number of training points P of different common architectures (different columns). The sensitivities compute how much internal representations, at increasing relative depth k for increasing row, are sensitive to transformations applied at the input. Note that at 80% relative depth all the networks reach small sensitivities and test error for large enough P. We choose this relative depth for the results shown in Figure 1 (A,B). The sensitivities S_k and D_k refer to the case $\ell = 1$ in Eq. 13 and Eq. 14.

Table 1. **Network architectures, main characteristics.** For each column we list the salient structures of a different architectures used in Figure 9. Table adapted with permission of the authors from (Petrini et al., 2021)

structures	**VGG** (Simonyan & Zisserman, 2015)	**ResNet** (He et al., 2016)	**EfficientNetB0-2** (Tan & Le, 2019)
depth	11, 16	18, 34	18
num. parameters	9-15 M	11-21 M	5 M
FC layers	1	1	1
activation	ReLU	ReLU	swish
pooling	max	avg. (last layer only)	avg. (last layer only)
dropout	/	/	yes + dropconnect
batch norm	if 'bn' in name	yes	yes
skip connections	/	yes	yes (inv. residuals)

C. Learning the SRHM with Gradient Descent

In hierarchical generative models, it is known that the first step of gradient descent in some simplified setup can already be sufficient to group together lowest-level synonyms (Malach & Shalev-Shwartz, 2020; Cagnetta et al., 2023b). In particular,

the case of the SRHM with $s_0 = 0$ has been analyzed in (Cagnetta et al., 2023b). Here we show for LCNs with a given initialization that the statistics of the first step of gradient descent for $s_0 > 0$ is equivalent to the case $s_0 = 0$, with a training set size reduced by a factor $(s_0 + 1)^L$, thus supporting the result of Eq. 3 on sample complexity.

We consider an instance of the SRHM with L levels, from which we generate P training points x_k with label $y(x_k)$. We recall that training points have dimension $d = (s(s_0 + 1))^L$, with each one of the informative s^L features being represented with one-hot encoding of v features, while the $(s_0 + 1)^L$ uninformative features by empty columns of dimension v. The labels are represented with a one-hot encoding of the n_c labels. To learn the SRHM, we use as a network a LCN with L hidden layers, followed by a linear layer. Each one of the hidden layers f_k for $k \in \{1, ..., L\}$ is defined with respect to the previous layer f_{k-1}. We use filters with size and stride equal to $s(s_0 + 1)$, yielding a reduction of $s(s_0 + 1)$ in the spatial size of the hidden layer at each layer. Each entry of f_k is specified by two indices, one for the channel $c \in \{1, ..., H_k\}$ and one for the spatial location $i \in \{1, ..., (s(s_0 + 1))^{L-k}$, and it is given by:

$$[f_k(x)]_{c;i} = \phi\left(\frac{1}{\sqrt{H_{k-1}}}\sum_{c'=1}^{H_{k-1}} \vec{w}^k_{c,c',i} \cdot \vec{p}_i\left([f_{k-1}(x)]_{c'}\right)\right), \tag{9}$$

where ϕ is the ReLU non-linearity function and $\vec{w}^k_{c,c',i}$ are the filters of the $c-$th channel of the k-th layer. Each filter $\vec{w}^k_{c,c',i}$ connects channel c of layer k with the patch $\vec{p}_i$ channel c' of layer $k-1$. Note that the filters depend on the patch location i, differently with respect to CNNs (see Figure 3). The vector $\vec{p}_{i,j}\left([f_{k-1}(x)]_{c'}\right)$ denotes a $s(s_0 + 1)$-dimensional patch of $[f_{k-1}(x)]_{c'}$ centered in the spatial location i at channel $c' \in \{1, .., H_{k-1}\}$. The bottom layer $k = 1$ is directly looking at the input, hence we define $f_0(x)$ as the identity and the number of channels H_1 is equal to v. The output of the LCN is given by:

$$f(x)_\alpha = \frac{1}{H_L}\sum_{c=1}^{H_L} a_{\alpha,c}[f_L(x)]_c, \tag{10}$$

with $f(x)$ being a vector of dimension equal to the number of classes n_c and $\{a_{\alpha,c}\}$ the parameters of the last linear layer. We train such a LCN with GD on the cross-entropy loss

$$\mathcal{L} = \sum_{k=1}^{P}\left[-\sum_{\beta=1}^{n_c} y(x_l)_\beta \log\left(\frac{e^{(f(x_k))_\beta}}{\sum_{\beta'=1}^{n_c} e^{(f(x_k))_{\beta'}}}\right)\right]. \tag{11}$$

For simplicity we (i) take all the number of channels H_k equal to H for $k > 1$, (ii) send $H \to \infty$, (iii) fix the linear layer parameters $a_{\alpha,c}$ to be i.i.d. standard Gaussian variables and (iv) initialize all the weights $\vec{\omega}^k_{c,c',i}$ to be equal to the unitary vector $\vec{1}$ renormalized by $\sqrt{H}$. As a consequence of (iii), the network output $f(x)_\alpha$ is zero identically at initialization.

We now derive the weight updates after one step of GD. Let's focus on the weights at the bottom layer $[\omega^1_{c,c',i}]_{i_0}$, where i_0 denotes a position within the filter of size $s(s_0 + 1)$ and $i \in \{1, ..., [s(s_0 + 1)]^{L-1}\}$. Note that, since there is no weight sharing, each position z within the input dimension d is seen just one weight at the bottom layer, with $z = z(i, i_0) = (i - 1)(s(s_0 + 1)) + i_0$. Consequently, the weight update for a weight $[\omega^1_{c,c',i}]_{i_0}$ just depends on the content of such pixel location $z = z(i, i_0)$:

$$\begin{aligned}\frac{\partial \Delta[\omega^1_{c,c',i}]_{i_0}}{\partial t} &= -\nabla_{[\omega^1_{c,c',i}]_{i_0}}\mathcal{L} \\ &= -\frac{1}{P}\sum_{k=1}^{P}\left[\sum_{\alpha=1}^{n_c}\left(\frac{1}{n_c} - (y(x_k))_\alpha\right)\frac{1}{H}\sum_{h_L=1}^{H} a_{\alpha,h_L}\frac{1}{H}\sum_{h_{L-1}=1}^{H}\cdots\frac{1}{H}\sum_{h_2=1}^{H}[x_k]_{c',z(i,i_0)}\right] \\ &= -\frac{1}{P}\sum_{k=1}^{P}\left[\sum_{\alpha=1}^{n_c}\left(\frac{1}{n_c} - (y(x_k))_\alpha\right)\frac{1}{H}\sum_{h_L=1}^{H} a_{\alpha,h_L}\right][x_k]_{c',z(i,i_0)}.\end{aligned} \tag{12}$$

On average over all the training sets, the pixel at location $z(i, i_0)$ of the input datum x_k relates to an informative feature with probability $p_L = (s_0+1)^{-L}$. Consequently, on average just a fraction $P' = p_L P$ of training points will give a non-vanishing contribute to the weight update Eq. 12, which will be identical to the GD equation of an instance of the sparseless SRHM with P' training points. Thus, the sample complexity in the sparse case $s_0 > 0$ is equal to the sparseless sample complexity $n_c m^L$ increased by a factor $(s_0 + 1)^L$, as empirically observed in Eq. 3. At this sample complexity, a single step of GD is capable to group together representations with equal correlation with the label, as shown in (Cagnetta et al., 2023b).

D. Sample complexities and learnt representations for LCNs

Architectures. Shown in Figure 3 (a). All layers consist of 512 channels.

Training scheme. We use Stochastic Gradient Descent (SGD) as optimizer, with batch size 4 and momentum 0.9. The learning rate lr has been optimized by extensive grid search, finding: $lr = 0.01$ for $s = 2$ and $s_0 < 4$, $lr = 0.01$ for $s = 2$ and $s_0 \geq 4$ and $lr = 0.003$ for $s \geq 2$. We use as train loss the cross entropy loss, and we stop the training when it reaches a threshold of 10^{-3}.

Sample complexity figures. In Figure 10, realised with $s = 3$, we show additional support to the scaling of the sample complexity P^*_{LCN}, defined in Eq. 3. In Figure 11 we show for $s = 3$ that the insensitivity to diffeomorphism and to synonyms exchange are achieved at the same sample complexity P^*_{LCN} at which the task is learned, supporting Eq. 8.

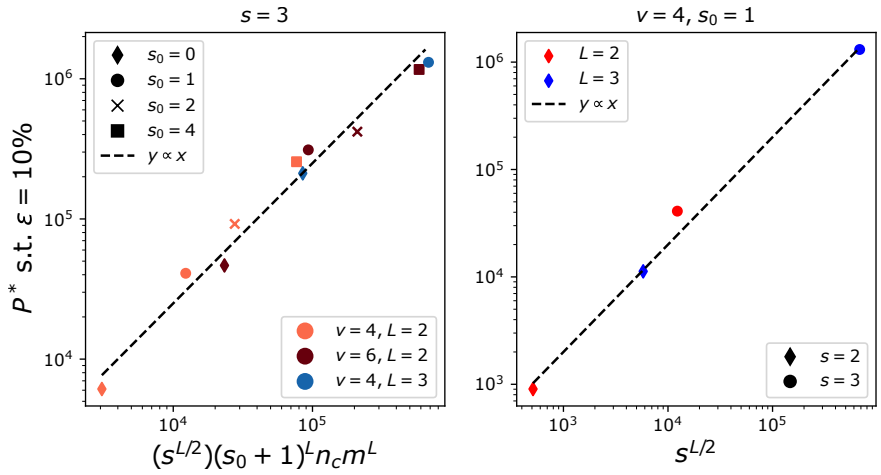

Figure 10. **LCN:** Left panel: empirical sample complexity P^* to reach a 10% test error ε versus prediction Eq. 3 for $s = 3$, different vocabulary sizes v and different depths L (different colors), number of classes $n_c = v$, maximal $m = v^{s-1}$ and different s_0 (different markers).Note an additional factor $s^{L/2}$ in the prediction, further supported by the right panel, realised at fixed $s_0 = 1$, $n_c = m = 4$, and varying s and L.

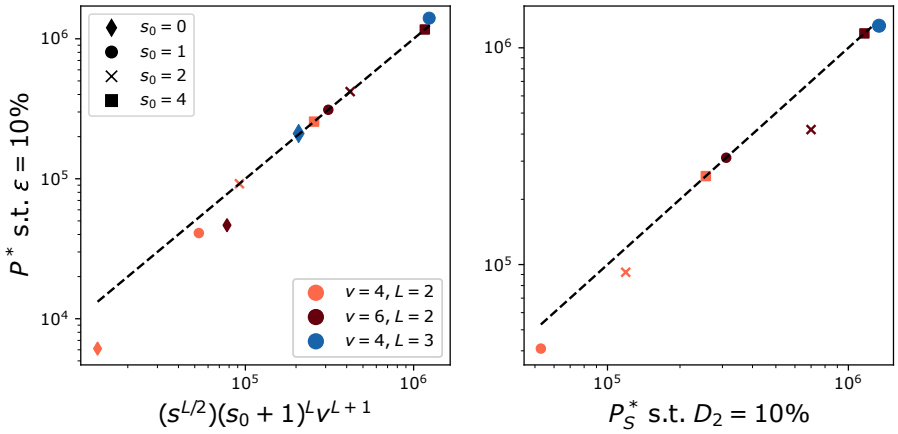

Figure 11. **LCN.** Left panel: empirical sample complexity P^* to reach a 10% test error ε versus empirical sample complexity P^*_S to reach $S_2 = 50\%$ (as defined in Eq. 6) for $s = 3$, different vocabulary sizes v and different depths L (different colors), number of classes $n_c = v$, maximal $m = v^{s-1}$ and different s_0 (different markers). Right panel: same as left for the sample complexity P^*_D to reach a $D_2 = 10\%$ (with D_2 as defined in Eq. 7). The thresholds have been tuned based on the form of S_2 and D_2 versus P.

Learnt representations figures. To assess whether the internal representation of the trained network have learnt the production rules Eq. 1 and Eq. 2, we define an operator p_l which takes in a datum x and substitutes each informative s

patches at generation level l with one of its $m-1$ synonyms, chosen uniformly at random, keeping the feature positions intact. The level l goes from $l=1$, the input, to L, the patch closest to the label. We define the sensitivity $S_{k,l}$ of the network representation f_k at an internal layer $k \in \{1, ..., L\}$ (with $k=1$ being the layer closest to the input while $k=L$ the one closest to the output) to synonymic exchange p_l as its average change before and after applying p_l on an input datum x:

$$S_{k,l} = \frac{\langle ||f_k(x) - f_k(p_l(x))||^2 \rangle_{x,p_l}}{\langle ||f_k(x_1) - f_k(x_2))||^2 \rangle_{x_1,x_2}}, \tag{13}$$

where x, x_1 and x_2 belong to a test set, while the average $\langle . \rangle$ is over the test set and on the random exchange of synonyms P_l. The same definition apply for the output of the deep network. The measure Eq. 6 is a particular case of Eq. 13 for $k=2$ and $l=1$, which can be visualized in Figure 6 (A).

To estimate the sensitivity to diffeomorphisms of trained networks we define an operator τ_l, which takes as input a datum x and shifts randomly its features according to the possible positions shown in Figure 2, panel (3), not impacting the feature values. Similarly to $S_{k,l}$, we define the sensitivity of f_k to diffeomorphisms τ_l at layer l as

$$D_{k,l} = \frac{\langle ||f_k(x) - f_k(\tau_l(x))||^2 \rangle_{x,\tau_l}}{\langle ||f_k(x_1) - f_k(x_2))||^2 \rangle_{x_1,x_2}}. \tag{14}$$

The measure Eq. 7 specializes Eq. 13 to $k=2$ and $l=1$, pictured in Figure 6 (B). T

he behaviours of $S_{k,l}$ and $D_{k,l}$ with respect to P are shown for different values of $s=L=2$ in Figure 12. We vary k and l in $\{1, ..., L\}$, and we analyse the sensitivity of the output too. For $k \geq l+1$ the sensitivities become significantly lower for large P. We checked the robustness of such observation for different values of s and L (not shown here).

E. Sample complexities and learnt representations for CNNs

Architectures. Shown in Figure 3 (b). All layers consist of 512 channels.

Training scheme. Same as LCN, except for the learning rate: $lr = 0.01$ for $s=2$ and $s_0 < 4$, $lr = 0.01$ for $s=2$ and $s_0 \geq 4$ and $lr = 0.003$ for $s \geq 2$.

Sample complexity figures. In Figure 13, realised with patch size $s=3$, we show additional support to the scaling of the sample complexity Eq. 3. In Figure 14 we show that, as for the LCNs, also for CNNs the insensitivity to diffeomorphism and to synonyms exchange are achieved at the same sample complexity P^*_{CNN} at which the task is learned, defined in Eq. 4.

Learnt representation figures. In Figure 15 we report the test error and the sensitivities to input transformations $S_{k,l}$ and $D_{k,l}$ defined in Eq. 13 and Eq. 14, with respect to P, for CNNs trained on the SRHM for different values of L and s. We vary k and l in $\{1, ..., L\}$, and we analyse the sensitivity of the output too. As for the LCNs, for $k \geq l+1$ the sensitivities become significantly lower for large P. We checked the robustness of such observation for different values of s and L (not shown here).

F. Sample complexities for FCNs

Architectures. The networks are made by stacking L full-connected layers, followed by a readout layer. All layers consist of 512 channels for $L=2$ and 256 channels for $L=3$.

Training scheme. Same as LCN, except for the learning rate: $lr = 0.01$ for $L=2$ and $lr = 0.003$ for $L=3$.

Sample complexity figures. In Figure 16, realised with patch size $s=2$, we show that, as for the LCNs and CNNs, also for full-connected networks (FCN) the insensitivity to diffeomorphism and to synonyms exchange are achieved at the same sample complexity at which the task is learned.

G. Correlation between test error and internal sensitivities

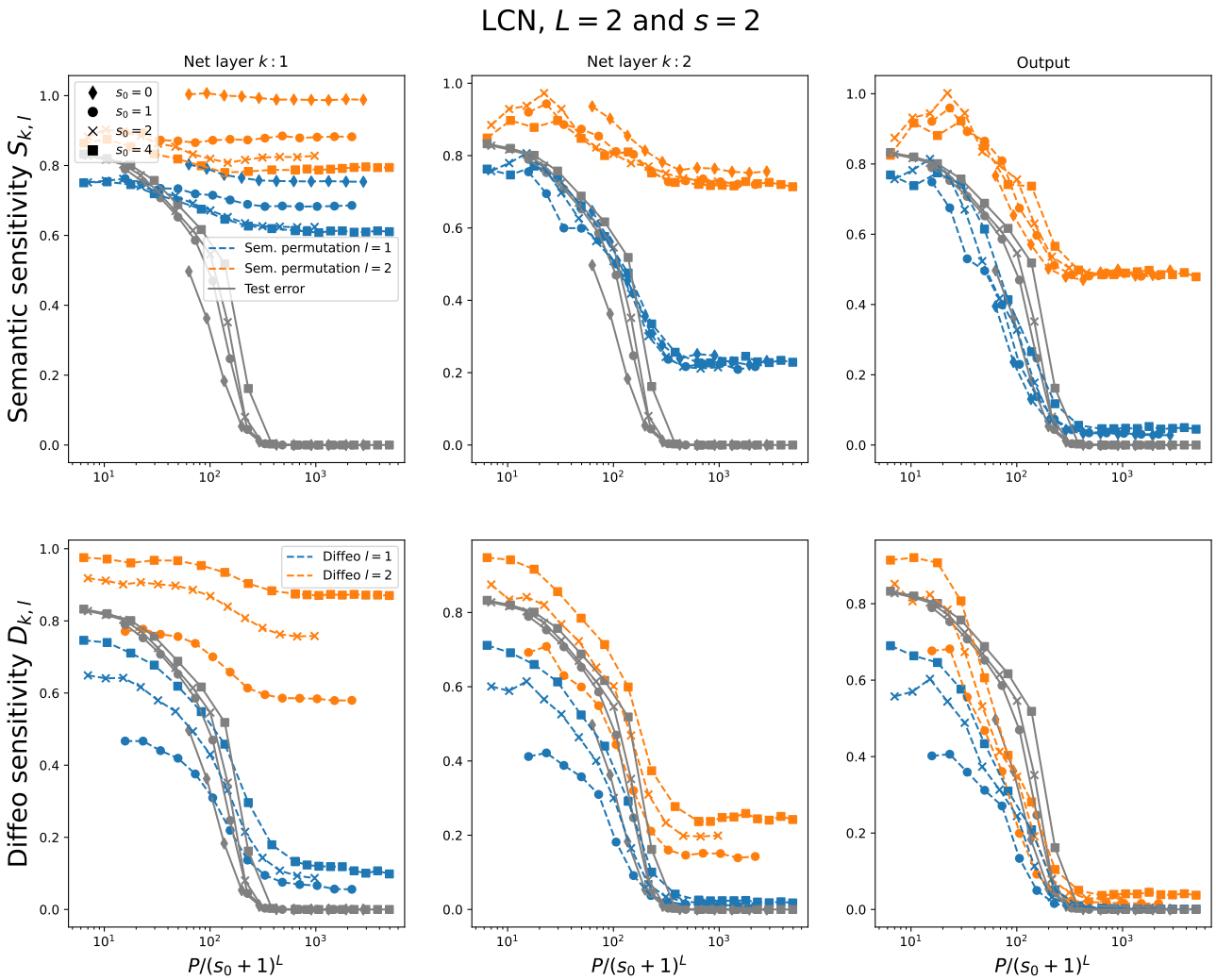

Figure 12. Top: sensitivity to synonyms exchange $S_{k,l}$ Eq. 13 of a $L-$layer LCN, trained on the SRHM with $L = 2$, $s = 2$ and $m = v = n_c = 8$, versus number of training points P rescaled by prediction Eq. 3. Going from left to right column the network layer k increases from the layer closest to the input to the output. For each column at fixed k there are plotted in color the sensitivities $S_{k,l}$ to synonyms exchange at the data level $l \in [1, ..., L]$, and in grey the test error. Different markers stand for different s_0. Bottom: same as top, but with the sensitivity to diffeomorphisms $D_{k,l}$ Eq. 14. The sensitivities S_2 and D_2 defined in Eq. 6 and Eq. 7 relate to the case $k = 2$ and $l = 1$ here.

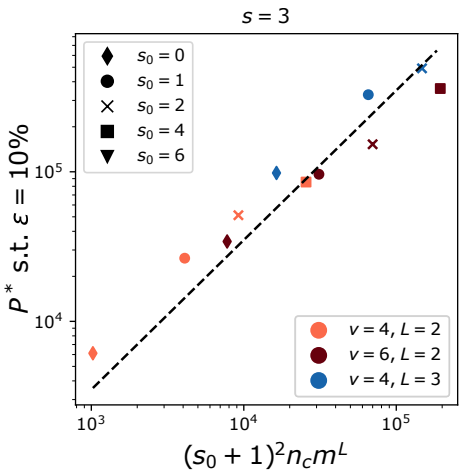

Figure 13. **CNN:** Empirical sample complexity P^* to reach a 10% test error ε versus prediction Eq. 4 for $s = 3$, for vocabulary size v and different depths L (different colors), maximal $m = v^{s-1}$ and different s_0 (different markers).

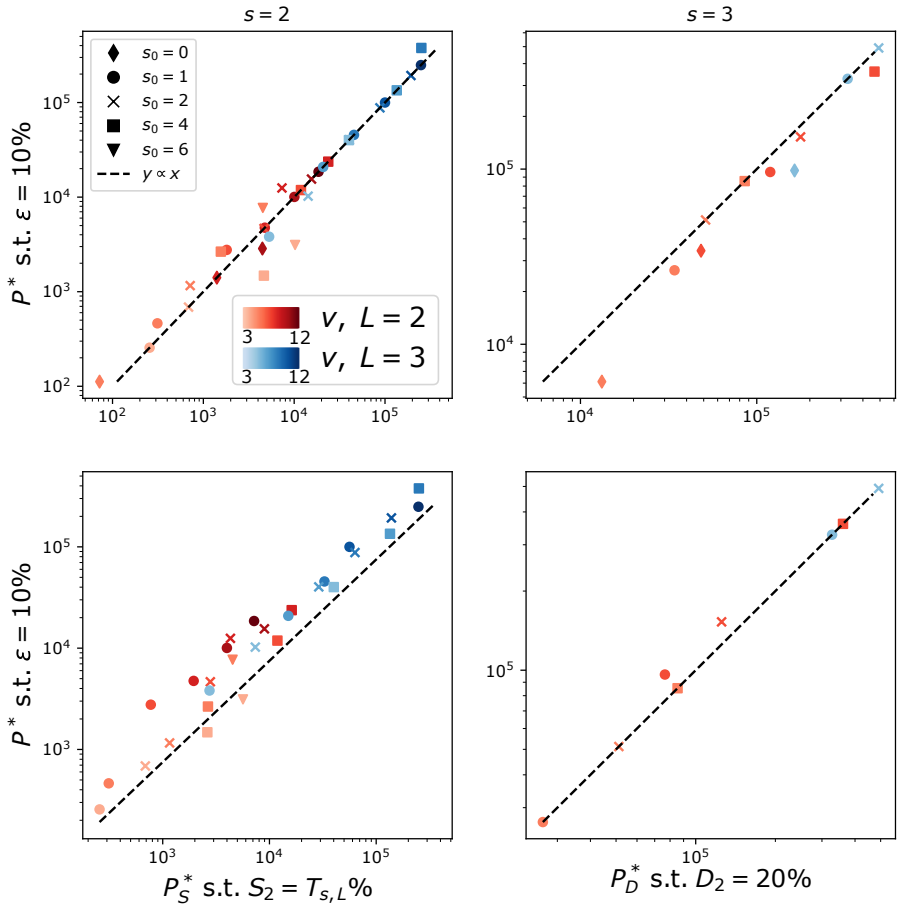

Figure 14. **CNN.** Left panels: empirical sample complexity P_S^* to reach $S_2 = T_{s,L}\%$ (with the threshold $T_{s,L} = 40$ if $L = 2$ and $s = 2$, 50 if $L = 2$ and $s = 3$, 20 if $L = 3$) versus empirical sample complexity P^* to reach a 10% test error ε for (Top) $s = 2$ and (Bottom) $s = 3$, for different depths L (red for $L = 2$, blue for $L = 3$) vocabulary sizes v (different darkness), maximal $m = v^{s-1}$ and different s_0 (different markers). Right panels: same as left panels, but with the sample complexity P_D^* to reach $D_2 = 20\%$. Both P_S^* and P_D^* align with P^*. The thresholds have been tuned based on the form of S_2 and D_2 versus P.

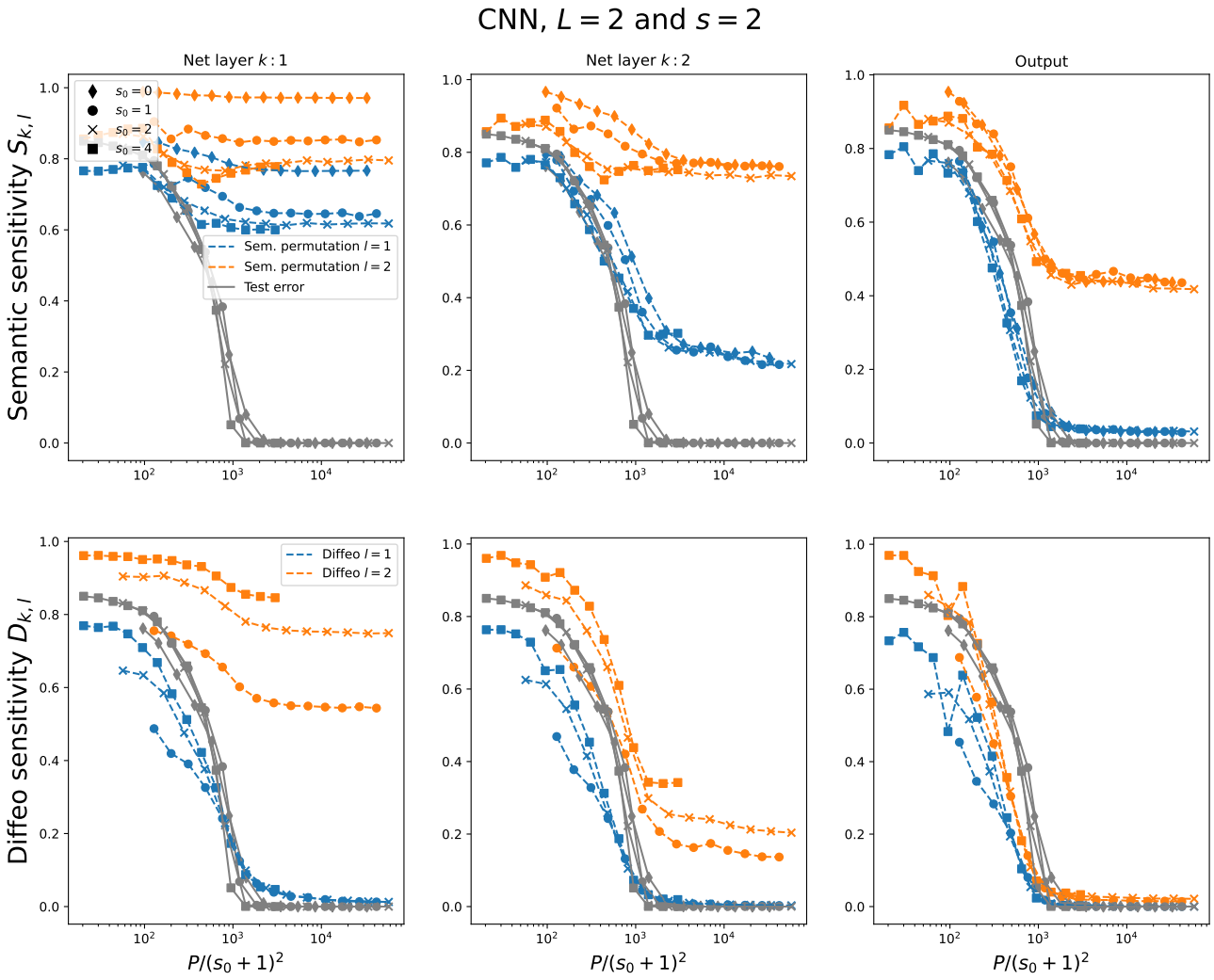

Figure 15. Top: sensitivity to synonyms exchange $S_{k,l}$ of a $L-$layer CNN, trained on the SRHM with $L = 2$, $s = 2$ and $m = v = 8$, versus number of training points P rescaled by prediction Eq. 4. Going from left to right column the network layer k increases from the layer closest to the input to the output. For each column at fixed k there are plotted in color the sensitivities $S_{k,l}$ to synonyms exchange at the data level $l \in [0, ..., L]$, and in grey the test error. Different markers stand for different s_0. Bottom: same as top, but with the sensitivity to diffeomorphisms $D_{k,l}$. The sensitivities S_2 and D_2 defined in Eq. 6 and Eq. 7 relate to the case $k = 2$ and $l = 1$ here.

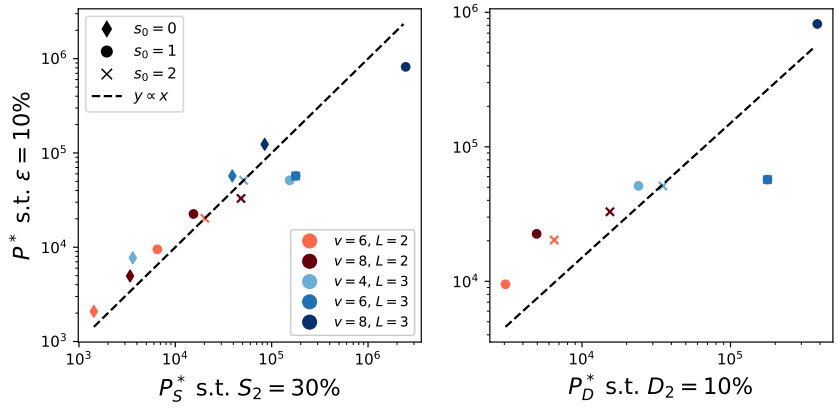

Figure 16. **FCN.** Left panel: empirical sample complexity P^* to reach a 10% test error ε versus empirical sample complexity P_S^* to reach $S_2 = 30\%$ for $s = 2$, different vocabulary sizes v and depths L (different colors), number of classes $n_c = v$, maximal $m = v^{s-1}$ and different s_0 (different markers). Right panel: same as left, for empirical sample complexity P^* to reach a 10% test error ε versus empirical sample complexity P_D^* to reach $D_2 = 10\%$.

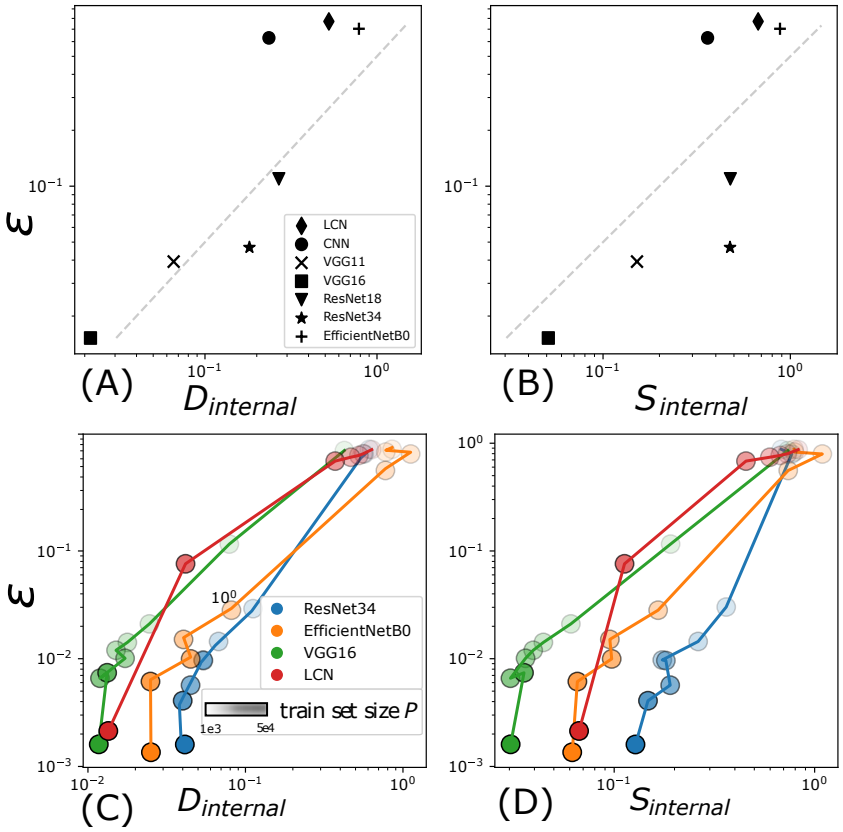

Figure 17. (A) Test error vs sensitivity D_{internal} to diffeomorphisms of an internal deep layer for a CNN trained with $P = 7400$ on the SHRM model, with parameters $L = s = s_0 = 2$ and $n_c = m = 10$. D_{internal} is defined as the change of the hidden representation induced by a diffeomorphism applied on the input, see Eq. 7. The internal layer is at 80% relative depth of the architecture, except for the 2 hidden-layer LCN, where it corresponds to the second layer. The test error shows a remarkable correlation with the sensitivities. A grey line, corresponding to a power-law, guides the eye. For details about the architectures and their training process, see Appendix B. (B) Same as (A) for sensitivity S_{internal} to synonymic exchanges, defined as the change of the hidden representation induced by an exchange of synonyms applied on the input, see Eq. 6. (C) and (D): as top panels (A) and (B), for increasing P (increasing opacity). The sensitivities of the network output yield the same observations, as shown in Figure 1.