Relative stability toward diffeomorphisms indicates performance in deep nets

Leonardo Petrini, Alessandro Favero, Mario Geiger, Matthieu Wyart

Institute of Physics
École Polytechnique Fédérale de Lausanne
1015 Lausanne, Switzerland
{name.surname}@epfl.ch

Abstract

Understanding why deep nets can classify data in large dimensions remains a challenge. It has been proposed that they do so by becoming stable to diffeomorphisms, yet existing empirical measurements support that it is often not the case. We revisit this question by defining a maximum-entropy distribution on diffeomorphisms, that allows to study typical diffeomorphisms of a given norm. We confirm that stability toward diffeomorphisms does not strongly correlate to performance on benchmark data sets of images. By contrast, we find that the *stability toward diffeomorphisms relative to that of generic transformations* R_f correlates remarkably with the test error ϵ_t. It is of order unity at initialization but decreases by several decades during training for state-of-the-art architectures. For CIFAR10 and 15 known architectures we find $\epsilon_t \approx 0.2\sqrt{R_f}$, suggesting that obtaining a small R_f is important to achieve good performance. We study how R_f depends on the size of the training set and compare it to a simple model of invariant learning.

1 Introduction

Deep learning algorithms LeCun et al. (2015) are now remarkably successful at a wide range of tasks Amodei et al. (2016); Huval et al. (2015); Mnih et al. (2013); Shi et al. (2016); Silver et al. (2017). Yet, understanding how they can classify data in large dimensions remains a challenge. In particular, the curse of dimensionality associated with the geometry of space in large dimension prohibits learning in a generic setting Luxburg and Bousquet (2004). If high-dimensional data can be learnt, then they must be highly structured.

A popular idea is that during training, hidden layers of neurons learn a representation Le (2013) that is insensitive to aspects of the data unrelated to the task, effectively reducing the input dimension and making the problem tractable Ansuini et al. (2019); Recanatesi et al. (2019); Shwartz-Ziv and Tishby (2017). Several quantities have been introduced to study this effect empirically. It includes (i) the mutual information between the hidden and visible layers of neurons Saxe et al. (2019); Shwartz-Ziv and Tishby (2017), (ii) the intrinsic dimension of the neural representation of the data Ansuini et al. (2019); Recanatesi et al. (2019) and (iii) the projection of the label of the data on the main features of the network Kopitkov and Indelman (2020); Oymak et al. (2019); Paccolat et al. (2021a), the latter being defined from the top eigenvectors of the Gram matrix of the neural tangent kernel (NTK) Jacot et al. (2018). All these measures support that the neuronal representation of the data indeed becomes well-suited to the task. Yet, they are agnostic to the nature of what varies in the data that need not being represented by hidden neurons, and thus do not specify what it is.

Recently, there has been a considerable effort to understand the benefits of learning features for one-hidden-layer fully connected nets. Learning features can occur and improve performance when the

Preprint. Under review.

true function is highly anisotropic, in the sense that it depends only on a linear subspace of the input space Bach (2017); Chizat and Bach (2020); Ghorbani et al. (2019, 2020); Paccolat et al. (2021a); Refinetti et al. (2021); Yehudai and Shamir (2019). For image classification, such an anisotropy would occur for example if pixels on the edge of the image are unrelated to the task. Yet, fully-connected nets (unlike CNNs) acting on images tend to perform best in training regimes where features are not learnt Geiger et al. (2021, 2020); Lee et al. (2020), suggesting that such a linear invariance in the data is not central to the success of deep nets.

Instead, it has been proposed that images can be classified in high dimensions because classes are invariant to smooth deformations or diffeomorphisms of small magnitude Bruna and Mallat (2013); Mallat (2016). Specifically, Mallat and Bruna could handcraft convolution networks, the *scattering transforms*, that perform well and are stable to smooth transformations, in the sense that $\|f(x) - f(\tau x)\|$ is small if the norm of the diffeomorphism τ is small too. They hypothesized that during training deep nets learn to become stable and thus less sensitive to these deformations, thus improving performance. More recent works generalize this approach to more common CNNs and discuss stability at initialization Bietti and Mairal (2019a,b). Interestingly, enforcing such a stability can improve performance Kayhan and Gemert (2020).

Answering if deep nets become more stable to smooth deformations when trained and quantifying how it affects performance remains a challenge. Recent empirical results revealed that small shifts of images can change the output a lot Azulay and Weiss (2018); Dieleman et al. (2016); Zhang (2019), in apparent contradiction with that hypothesis. Yet in these works, image transformations (i) led to images whose statistics were very different from that of the training set or (ii) were cropping the image, thus are not diffeophormisms. In Ruderman et al. (2018), a class of diffeomorphisms (low-pass filter in spatial frequencies) was introduced to show that stability toward them can improve during training, especially in architectures where pooling layers are absent. Yet, these studies do not address how stability affects performance, and how it depends on the size of the training set. To quantify these properties and to find robust empirical behaviors across architectures, we will argue that the evolution of stability toward smooth deformations needs to be compared relatively to that of any deformation, which turns out to vary significantly during training.

Note that in the context of adversarial robustness, attacks that are geometric transformations of small norm that change the label have been studied Alaifari et al. (2018); Alcorn et al. (2019); Athalye et al. (2018); Engstrom et al. (2019); Fawzi and Frossard (2015); Kanbak et al. (2018); Xiao et al. (2018). These works differ for the literature above and from out study below in the sense that they consider worst-case perturbations instead of typical ones.

1.1 Our Contributions

- We introduce a *maximum entropy distribution* of diffeomorphisms, that allow us to generate typical diffeomorphisms of controlled norm. Their amplitude is governed by a "temperature" parameter T.
- We define the *relative stability to diffeomorphisms index* R_f that characterizes the square magnitude of the variation of the output function f with respect to the input when it is transformed along a diffeomorphism, relatively to that of a random transformation of the same amplitude. It is averaged on the test set as well as on the ensemble of diffeomorphisms considered.
- We find that at initialization, R_f is close to unity for various data sets and architectures, indicating that initially the output is as sensitive to smooth deformations as it is to random perturbations of the image.
- Our central result is that after training, R_f correlates very strongly with the test error ϵ_t: during training, R_f is reduced by several decades in current State Of The Art (SOTA) architectures on four benchmark datasets including MNIST Lecun et al. (1998), FashionMNIST Xiao et al. (2017), CIFAR-10 Krizhevsky (2009) and ImageNet Deng et al. (2009). For more primitive architectures (whose test error is higher) such as fully connected nets or simple CNNs, R_f remains of order unity. For CIFAR10 we study 15 known architectures and find empirically that $\epsilon_t \approx 0.2\sqrt{R_f}$.
- R_f decreases with the size of the training set P. We compare it to an inverse power $1/P$ expected in simple models of invariant learning Paccolat et al. (2021a).

The library implementing diffeomorphisms on images is available online at github.com/pcsl-epfl/diffeomorphism.

The code for training SOTA nets can be found at github.com/leonardopetrini/diffeo-sota.

2 Maximum-entropy model of diffeomorphisms

2.1 Definition of maximum entropy model

We consider the case where the input vector x is an image. It can be thought as a function $x(s)$ describing intensity in position $s = (u, v) \in [0, 1]^2$, where u and v are the horizontal and vertical coordinates. To simplify notations we consider a single channel, in which case $x(s)$ is a scalar (but our analysis holds for colored images as well). We denote by τx the image deformed by τ, i.e. $[\tau x](s) = x(s - \tau(s))$. $\tau(s)$ is a vector field of components $(\tau_u(s), \tau_v(s))$. The deformation amplitude is measured by the norm

$$\|\nabla\tau\|^2 = \int_{[0,1]^2} ((\nabla\tau_u)^2 + (\nabla\tau_v)^2) du dv. \tag{1}$$

To test the stability of deep nets toward diffeomorphisms, we seek to build *typical* diffeomorphisms of controlled norm $\|\nabla\tau\|$. We thus consider the distribution over diffeomorphisms that maximizes the entropy with a norm constraint. It can be solved by introducing a Lagrange multiplier T and by decomposing these fields on their Fourier components, see e.g. Kardar (2007) or Appendix A. In this canonical ensemble, one finds that τ_u and τ_v are independent with identical statistics. For the picture frame not to be deformed, we impose fixed boundary conditions: $\tau = 0$ if $u = 0, 1$ or $v = 0, 1$. One then obtains:

$$\tau_u = \sum_{i,j\in\mathbb{N}^+} C_{ij} \sin(i\pi u) \sin(j\pi v) \tag{2}$$

where the C_{ij} are Gaussian variables of zero mean and variance $\langle C_{ij}^2 \rangle = T/(i^2 + j^2)$. If the picture is made of $n \times n$ pixels, the result is identical except that the sum runs on $0 < i, j \leq n$. For large n, the norm then reads $\|\nabla\tau\|^2 = (\pi^2/2)\, n^2 T$, and is dominated by high spatial frequency modes. It is useful to add another parameter c to cut-off the effect of high spatial frequencies, which can be simply done by constraining the sum in Eq.2 to $i^2 + j^2 \leq c^2$, one then has $\|\nabla\tau\|^2 = (\pi^3/8)\, c^2 T$.

Once τ is generated, pixels are displaced to random positions. A new pixelated image can then be obtained using standard interpolation methods. We use two interpolations, Gaussian and bi-linear[1], as described in Appendix C. As we shall see below, this choice does not affect our result as long as the diffeomorphism induced a displacement of order of the pixel size, or larger. Examples are shown in Fig.1 as a function of T and c.

2.2 Phase diagram of acceptable diffeomorphisms

Diffeomorphisms are bijective, which is not the case for our transformations if T is too large. When this condition breaks down, a single domain of the picture can break into several pieces, as apparent in Fig.1. It can be expressed as a condition on $\nabla\tau$ that must be satisfied in every point in space Lowe (2004), as recalled in Appendix B. This is satisfied locally with high probability if $\|\tau\|^2 \ll 1$, corresponding to $T \ll (8/\pi^3)/c^2$. In Appendix, we extract empirically a curve of similar form in the (T, c) plane at which a diffeomorphism is obtained with probability at least $1/2$. For much smaller T, diffeomorphisms are obtained almost surely.

Finally, for diffeomorphisms to have noticeable consequences, their associated displacement must be of the order of magnitude of the pixel size. Defining δ^2 as the average square norm of the pixel displacement at the center of the image, it is straightforward to obtain from Eq.2 that asymptotically for large c,

$$\delta^2 = \frac{\pi}{4} n^2 T \ln(c). \tag{3}$$

The line $\delta = 1/2$ is indicated in Fig.1, using empirical measurements that add pre-asymptotic terms to Eq.3. Overall, the green region corresponds to transformations that (i) are diffeomorphisms with high probability and (ii) produce significant displacements at least of the order of the pixel size.

[1]Throughout the paper, if not specified otherwise, bi-linear interpolation is employed.

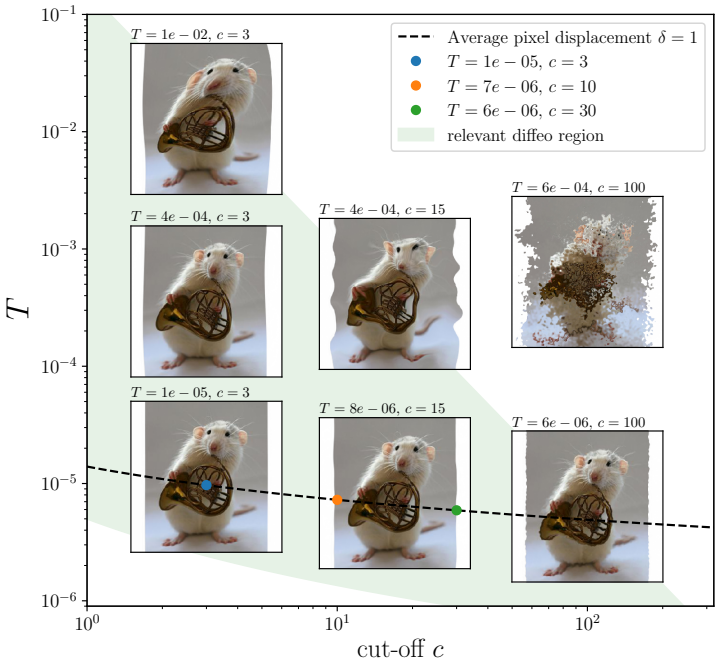

Figure 1: Samples of max-entropy diffeomorphisms for different temperatures T and high-frequency cut-offs c for an ImageNet data-point. The green region corresponds to well behaving diffeomorphisms (see Section 2.2). The dashed line corresponds to $\delta = 1$. The colored points on the line are those we focus our study in Section 3.

3 Measuring the relative stability to diffeomorphisms

Relative stability to diffeomorphisms To quantify how a deep net f learns to become less sensitive to diffeomorphisms than to generic data transformations, we define the relative stability to diffeomorphisms R_f as:

$$R_f = \frac{\langle \|f(\tau x) - f(x)\|^2 \rangle_{x,\tau}}{\langle \|f(x+\eta) - f(x)\|^2 \rangle_{x,\eta}}. \tag{4}$$

where the notation $\langle \rangle_y$ can indicate alternatively the mean or the median with respect to the distribution of y. In the numerator, this operation is made over the test set and over the ensemble of diffeomorphisms of parameters (T, c) (on which R_f implicitly depends). In the denominator, the average is on the test set and on all random directions η, whose magnitude follows $\|\eta\| = \langle \|\tau x - x\| \rangle$. An illustration of what R_f captures is shown in Fig.2. In the main text, we consider median quantities, as they reflect better the typical values of distribution. In Appendix E.3 we show that our results for mean quantities, for which our conclusions also apply.

Dependence of R_f on the diffeomorphism magnitude Ideally, R_f could be defined for infinitesimal transformations, as it would then characterize the magnitude of the gradient of f along smooth deformations of the images, normalized by the magnitude of the gradient in random directions. However, infinitesimal diffeomorphisms move the image much less than the pixel size, and their definition thus depends significantly on the interpolation method used. It is illustrated in the left panels of Fig.3, showing the dependence of R_f in terms of the diffeomorphism magnitude (here characterised by the mean displacement magnitude at the center of the image δ) for several interpolation methods. We do see that R_f becomes independent of the interpolation when δ becomes of order unity. In what follows we thus focus on $R_f(\delta = 1)$, which we denote R_f.

SOTA architectures become relatively stable to diffeomorphisms during training, but not at initialization The central panels of Fig.3 show R_f at initialization (shaded), and after training (full) for two SOTA architectures on four benchmark data sets. The first key result is that, at initialization, these architectures are as sensitive to diffeomorphisms as they are to random transformations. Stability to diffeomorphisms at initialization (guaranteed theoretically in some cases Bietti and Mairal (2019a,b)) thus does not appear to be indicative of successful architectures.

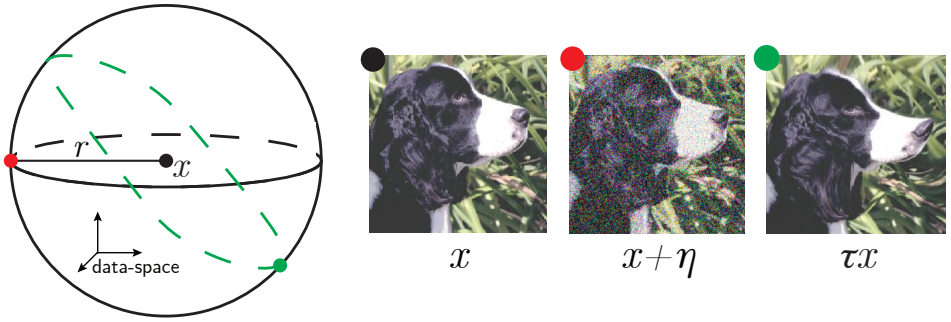

Figure 2: Illustrative drawing of the data-space $\mathbb{R}^{n\times n}$ around a data-point x (black point). We focus here on perturbations of fixed magnitude – i.e. on the sphere of radius r centered in x. The intersection between the images of x transformed via typical diffeomorphisms and the sphere is represented in dashed green. By contrast, the red point is an example of random transformation. For large n, it is equivalent to adding an i.i.d. Gaussian noise to all the pixel values of x. Figures on the right illustrate these transformations, the color of the dot labelling them corresponds to that of the left illustration. The relative stability to diffeomorphisms R_f characterizes how a net f varies in the green directions, normalized by random ones.

By contrast, for these SOTA architectures, invariance toward diffeomorphisms builds up during training on all the data sets probed. It is a significant effect, with values of R_f after training generally found in the range $R_f \in [10^{-2}, 10^{-1}]$.

Standard data augmentation techniques (translations, crops, and horizontal flips) are employed for training. However, the results we find only mildly depend on using such techniques, see Fig.12 in Appendix.

Learning stability to diffeos requires large training sets How many data are needed to learn stability toward diffeomorphisms? To answer this question, newly initialized networks are trained on different training-sets of size P. R_f is then measured for CIFAR10, as indicated in the right panels of Fig.3. Neural nets need a certain number of training points ($P \sim 10^3$) in order to become stable toward smooth deformations. Past that point, R_f monotonically decreases with P. In a range of P, this decrease is approximately compatible with the an inverse behavior $R_f \sim 1/P$ found in the simple model of Section 6. Additional results for MNIST and FashionMNIST can be found in Fig.13, Appendix E.3.

Simple architectures do not become stable to diffeomorphisms To test the universality of these results, we focus on two simple architectures: (i) a 4-hidden-layer fully connected (FC) network (FullConn-L4) where each hidden layer has 64 neurons and (ii) LeNet LeCun et al. (1989) that consists of two convolutional layers followed by local max-pooling and three fully-connected layers.

Measurements of R_f for these networks are shown in Fig.4. For the FC net, $R_f \approx 1$ at initialization (as observed for SOTA nets) but *grows* after training on the full data set, showing that FC nets do not learn to become stable to smooth deformations. It is consistent with the modest evolution of $R_f(P)$ with P, suggesting that huge training sets would be required to obtain $R_f < 1$. The situation is similar for the primitive CNN LeNet, which only becomes slightly insensitive ($R_f \approx 0.6$) in a single data set (CIFAR10), and otherwise remains larger than unity.

Layers' stability monotonically increases with depth Up to this point, we measured the relative stability of the output function for any given architecture. We now study how this stability builds up as the input data propagate through the hidden layers. In Fig.14 of Appendix E.3, we report R_f as a function of depth for both simple and deep nets. What we observe is $R_{f_0} \approx 1$ independently of depth at initialization, and monotonically decreases with depth after training. Overall, the gain in relative stability appears to be well-spread through the net, as is also found for stability alone Ruderman et al. (2018).

[2]With the only exception of the ImageNet results (central panel) in which only one trained network is considered.

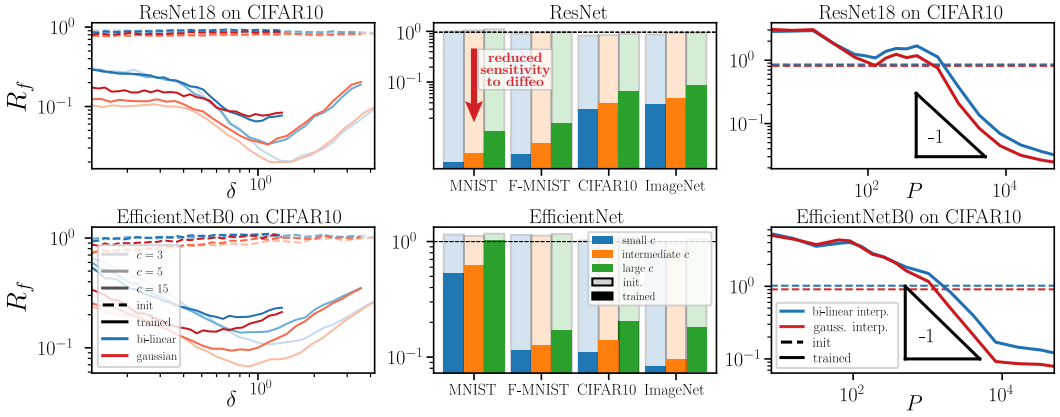

Figure 3: **Relative stability to diffeomorphisms R_f for SOTA architectures.** Left panels: R_f *vs.* diffeomorphism displacement magnitude δ at initialization (dashed lines) and after training (full lines) on the full data set of CIFAR10 ($P = 50k$) for several cut-off parameters c and two interpolations methods, as indicated in legend. ResNet is shown on the top and EfficientNet on the bottom. Central panels: $R_f(\delta = 1)$ for four different data-sets ($x-$axis) and two different architectures at initialization (shaded histograms) and after training (full histograms). The values of c (in different colors) are $(3, 5, 15)$ and $(3, 10, 30)$ for the first three data-sets and ImageNet, respectively. ResNet18 and EfficientNetB0 are employed for MNIST, F-MNIST and CIFAR10, ResNet101 and EfficientNetB2 for ImageNet. Right panels: $R_f(\delta = 1)$ *vs.* training set size P at $c = 3$ for ResNet18 (top) and EfficientNetB0 (bottom) trained on CIFAR10. The value of R_{f_0} at initialization is indicated with dashed lines. The triangles indicate the predicted slope $R_f \sim P^{-1}$ in a simple model of invariant learning, see Section 6. *Statistics*: Each point in the graphs[2] is obtained by training 16 differently initialized networks on 16 different subsets of the data-sets; each network is then probed with 500 test samples in order to measure stability to diffeomorphisms and Gaussian noise. The resulting R_f is obtained by log-averaging the results from single realizations.

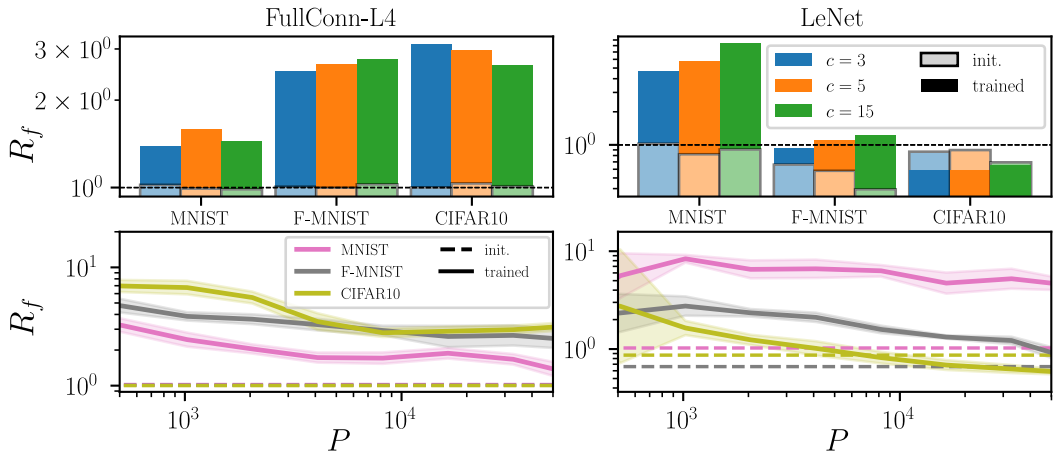

Figure 4: **Relative stability to diffeomorphisms R_f in primitive architectures.** Top panels: R_f at initialization (shaded) or for trained nets (full) for a fully connected net (left) or a primitive CNN (right) at $P = 50k$. Bottom panels: $R_f(P)$ for $c = 3$ and different data sets as indicated in legend. *Statistics:* see caption in the previous figure.

4 Relative stability to diffeomorphisms indicates performance

Thus, SOTA architectures appear to become stable to diffeomorphisms after training, unlike primitive architectures. This observation suggests that high performance requires such a stability to build up. To test further this hypothesis, we select a set of architectures that have been relevant in the state of the art progress over the past decade; we systematically train them in order to compare R_f to their test error ϵ_t. Apart from fully connected nets, we consider the already cited LeNet (5 layers and $\approx 60k$ parameters); then AlexNet Krizhevsky et al. (2012) and VGG Simonyan and Zisserman (2015), deeper (8-19 layers) and highly over-parametrized (10-20M (million) params.) versions of the latter. We introduce *batch-normalization* in VGGs and *skip connections* with ResNets. Finally, we go to EfficientNets, that have all the advancements introduced in previous models and achieve SOTA performance with a relatively small number of parameters (<10M); this is accomplished by designing an efficient small network and properly scaling it up. Further details about these architectures can be found in Table 1, Appendix E.2.

The results are shown in Fig.5. The correlation between R_f and ϵ_t is remarkably high (corr. coeff.[3] : 0.97), suggesting that generating low relative sensitivity to diffeomorphisms R_f is important to obtain good performance. In Appendix E.3 Fig.18, we also report how changing the train set size P affects the position of a network in the (ϵ_t, R_f) plane, for the four architectures considered in the previous section.

What architectures enable a low R_f value? The latter can be obtained with skip connections or not, and for quite different depths as indicated in Fig.5. Also, the same architecture (EfficientNetB0) trained by transfer learning from ImageNet – instead of directly on CIFAR10 – shows a large improvement both in performance and in diffeomorphisms invariance. Clearly, R_f is much better predicted by ϵ_t than by the specific features of the architecture indicated in Fig.5.

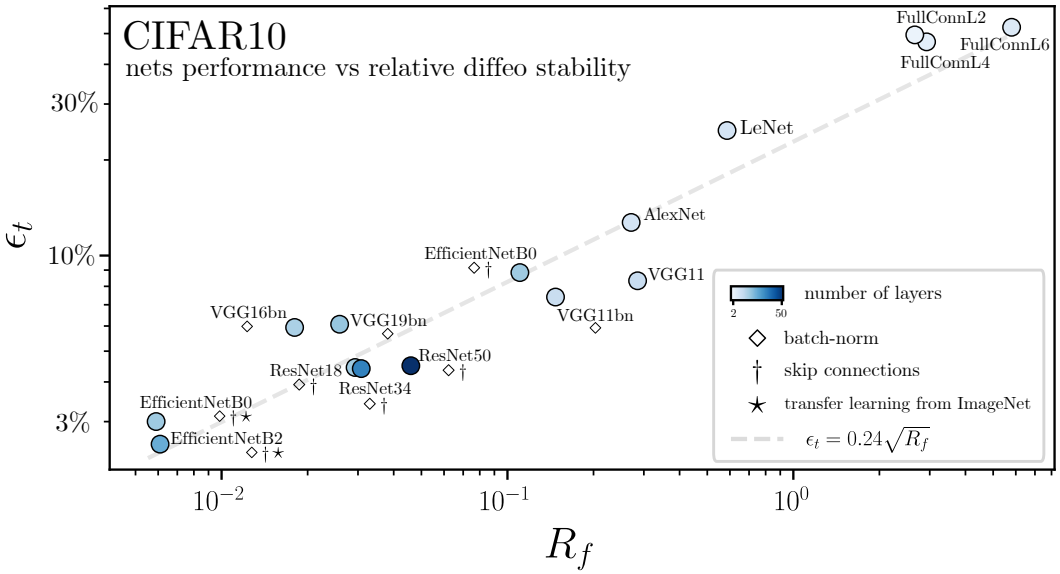

Figure 5: **Test error ϵ_t *vs.* relative stability to diffeomorphisms** R_f computed at $\delta = 1$ and $c = 3$ for common architectures when trained on the full 10-classes CIFAR10 dataset ($P = 50k$) with SGD and the cross-entropy loss; the EfficientNets achieving the best performance are trained by transfer learning from ImageNet ($\star$) – more details on the training procedures can be found in Appendix E.1. The color scale indicates depth, and the symbols the presence of batch-norm ($\diamond$) and skip connections ($\dagger$). Dashed grey line: power low fit $\epsilon_t \approx 0.2\sqrt{R_f}$. R_f strongly correlates to ϵ_t, much less so to depth or the presence of skip connections. *Statistics:* Each point is obtained by training 5 differently initialized networks; each network is then probed with 500 test samples in order to measure R_f. The results are obtained by log-averaging over single realizations. Error bars – omitted here – are shown in Fig.19, Appendix E.3.

5 Stability toward diffeomorphisms *vs.* noise

The relative stability to diffeomorphisms R_f can be written as $R_f = {}^{D_f}/_{G_f}$ where G_f characterizes the stability with respect to additive noise and D_f the stability toward diffeomorphisms:

$$G_f = \frac{\langle \|f(x+\eta) - f(x)\|^2 \rangle_{x,\eta}}{\langle \|f(x) - f(z)\|^2 \rangle_{x,z}}, \qquad D_f = \frac{\langle \|f(\tau x) - f(x)\|^2 \rangle_{x,\tau}}{\langle \|f(x) - f(z)\|^2 \rangle_{x,z}}. \tag{5}$$

Here, we chose to normalize these stabilities with the variation of f over the test set (to which both x and z belong), and η is a random noise whose magnitude is prescribed as above. Stability toward additive noise has been studied previously in fully connected architectures Novak et al. (2018) and for CNNs as a function of spatial frequency in Tsuzuku and Sato (2019); Yin et al. (2019). Note that although our definition of G_f is expressed in terms of finite variations, Appendix D shows that observations are made in a linear regime where $G_f \propto \|\eta\|^2$.

The decrease of R_f with growing training set size P could thus be due to an increase in the stability toward diffeomorphisms (i.e. D_f decreasing with P) or a decrease of stability toward noise (G_f increasing with P). To test these possibilities, we show in Fig.6 $G_f(P), D_f(P)$ and $R_f(P)$ for MNIST, Fashion MNIST and CIFAR10 for two SOTA architectures. The central results are that (i) stability toward noise is always reduced for larger training sets. This observation is natural: when more data needs to be fitted, the function becomes rougher. (ii) Stability toward diffeomorphisms does not behave universally: it can increase with P or decrease depending on the architecture and the training set. Additionally, G_f and D_f alone show a much smaller correlation with performance than R_f– see Figs.15,16,17 in Appendix E.3.

Our work thus supports that stability is not the good observable to characterize how deep nets learn invariance toward diffeomorphisms, as it includes two antagonist effects: the overall roughening of the learnt function with increasing P (that should reduce stability toward any transformations) and the relative stability toward diffeomorphisms captured by R_f, that universally decreases with P.

It may seem surprising at first that for a fixed size of the training set, the best architectures tend to display the weakest stability toward random perturbations of the image (see Figs 16,17), as such a stability may seem desirable. We offer the following interpretation: large gradients are required to fit such complex data sets, in particular near the boundary between distinct labels. Thanks to their stability toward diffeomorphisms, SOTA architectures can generalize these strong gradients to larger regions of data space when they detect their existence. They thus display larger gradients than less performant architectures, which require more data to build them.

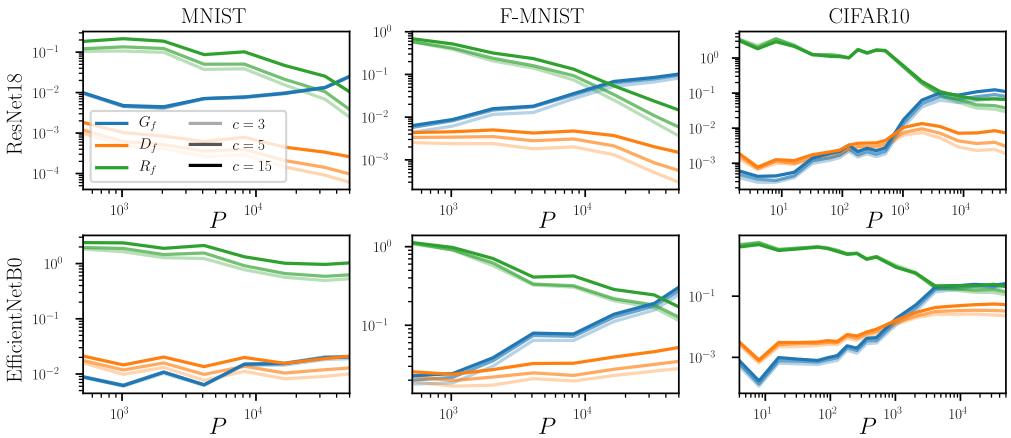

Figure 6: **Stability toward Gaussian noise (G_f) and diffeomorphisms (D_f) alone, and the relative stability R_f.** Columns correspond to different data-sets (MNIST, FashionMNIST and CIFAR10) and rows to architectures (ResNet18 and EfficientNetB0). Each panel reports G_f (blue), D_f (orange) and R_f (green) as a function of P and for different cut-off values c, as indicated in the legend. *Statistics:* cf. caption in Fig.3. Error bars – omitted here – are shown in Fig.20, Appendix E.3.

[3]Correlation coefficient: $\frac{\mathrm{Cov}(\log \epsilon_t, \log R_f)}{\sqrt{\mathrm{Var}(\log \epsilon_t)\mathrm{Var}(\log R_f)}}$.

6 A minimal model for learning invariants

In this section, we discuss the simplest model of invariance in data where stability to transformation builds up, that can be compared with our observations of R_f above. Specifically, we consider the "stripe" model Paccolat et al. (2021b), corresponding to a binary classification task for Gaussian-distributed data points $x = (x_\parallel, x_\perp)$ where the label function depends only on one direction in data space, namely $y(x) = y(x_\parallel)$. Layers of $y = +1$ and $y = -1$ regions alternate along the direction $x_\parallel$, separated by parallel planes. Hence, the data present $d - 1$ invariant directions in input-space denoted by $x_\perp$ as illustrated in Fig.7-left.

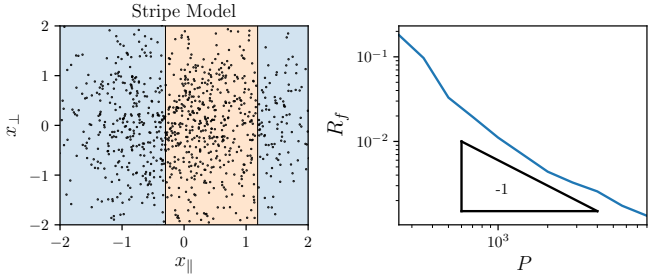

Figure 7: Left: example of the stripe model. Dots are data-points, the vertical lines represent the decision boundary and the color the class label. Right: Relative stability R_f for the stripe model in $d = 30$. The slope of the curve is -1, as predicted.

When this model is learnt by a one-hidden-layer fully connected net, the first layer of weights can be shown to align with the informative direction Paccolat et al. (2021a). The projection of these weights on the orthogonal space vanishes with the training set size P as $1/\sqrt{P}$, an effect induced by the sampling noise associated to finite training sets.

In this model, R_f can be defined as:

$$R_f = \frac{\langle \|f(x_\parallel, x_\perp + \nu) - f(x_\parallel, x_\perp)\|^2 \rangle_{x,\nu}}{\langle \|f(x+\eta) - f(x)\|^2 \rangle_{x,\eta}}, \tag{6}$$

where we made explicit the dependence of f on the two linear subspaces. Here, the isotropic noise ν is added only in the invariant directions. Again, we impose $\|\eta\| = \|\nu\|$. $R_f(P)$ is shown in Fig. 7-right. We observe that $R_f(P) \sim P^{-1}$, as expected from the weight alignment mentioned above.

Interestingly, Fig.3 for CIFAR10 and SOTA architectures support that the $1/P$ behavior is compatible with the observations for some range of P. In Appendix E.3, Fig.13, we show analogous results for MNIST and Fashion-MNIST. We observe the $1/P$ power-law scaling for ResNets. It suggests that for these architectures, learning to become invariant to diffeomorphisms may be limited by a naive measure of sampling noise as well. By contrast for EfficientNets, in which the decrease in R_f is more limited, a $1/P$ behavior cannot be identified.

7 Conclusion

We have introduced a novel empirical framework to characterize how deep nets become invariant to diffeomorphisms. It is jointly based on a maximum-entropy distribution for diffeomorphisms, and on the realization that stability of these transformations relative to generic ones R_f strongly correlates to performance, instead of just the diffeomorphisms stability considered in the past.

The ensemble of smooth deformations we introduced may have interesting applications. It could serve as a complement to traditional data-augmentation techniques (whose effect on relative stability is discussed in Fig.12 of the Appendix). It could also be used to build adversarial attacks along smooth transformations, in the spirit of Alaifari et al. (2018); Engstrom et al. (2019); Kanbak et al. (2018). It would be interesting to test if networks robust to such attacks are more stable in relative terms, and how such robustness affects their performance.

Acknowledgements

We thank Alberto Bietti, Joan Bruna, Francesco Cagnetta, Pascal Frossard, Jonas Paccolat, Antonio Sclocchi and Umberto M. Tomasini for helpful discussions. This work was supported by a grant from the Simons Foundation (#454953 Matthieu Wyart).

References

Alaifari, R., Alberti, G. S., and Gauksson, T. (2018). ADef: an Iterative Algorithm to Construct Adversarial Deformations.

Alcorn, M. A., Li, Q., Gong, Z., Wang, C., Mai, L., Ku, W.-S., and Nguyen, A. (2019). 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)*, pages 4840–4849, Long Beach, CA, USA. IEEE.

Amodei, D., Ananthanarayanan, S., Anubhai, R., Bai, J., Battenberg, E., Case, C., Casper, J., Catanzaro, B., Cheng, Q., Chen, G., et al. (2016). Deep speech 2: End-to-end speech recognition in english and mandarin. In *International conference on machine learning*, pages 173–182.

Ansuini, A., Laio, A., Macke, J. H., and Zoccolan, D. (2019). Intrinsic dimension of data representations in deep neural networks. In *Advances in Neural Information Processing Systems*, pages 6111–6122.

Athalye, A., Engstrom, L., Ilyas, A., and Kwok, K. (2018). Synthesizing Robust Adversarial Examples. In *International Conference on Machine Learning*, pages 284–293. PMLR. ISSN: 2640-3498.

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

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

Beale, P. (1996). *Statistical Mechanics*. Elsevier Science.

Bietti, A. and Mairal, J. (2019a). Group invariance, stability to deformations, and complexity of deep convolutional representations. *The Journal of Machine Learning Research*, 20(1):876–924.

Bietti, A. and Mairal, J. (2019b). On the inductive bias of neural tangent kernels. *arXiv preprint arXiv:1905.12173*.

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

Chizat, L. and Bach, F. (2020). Implicit Bias of Gradient Descent for Wide Two-layer Neural Networks Trained with the Logistic Loss. In *Conference on Learning Theory*, pages 1305–1338. PMLR. ISSN: 2640-3498.

Deng, J., Dong, W., Socher, R., Li, L., Kai Li, and Li Fei-Fei (2009). ImageNet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255. ISSN: 1063-6919.

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

Engstrom, L., Tran, B., Tsipras, D., Schmidt, L., and Madry, A. (2019). Exploring the Landscape of Spatial Robustness. In *International Conference on Machine Learning*, pages 1802–1811. PMLR. ISSN: 2640-3498.

Fawzi, A. and Frossard, P. (2015). Manitest: Are classifiers really invariant? In *Procedings of the British Machine Vision Conference 2015*, pages 106.1–106.13, Swansea. British Machine Vision Association.

Geiger, M., Petrini, L., and Wyart, M. (2021). Landscape and training regimes in deep learning. *Physics Reports*.

Geiger, M., Spigler, S., Jacot, A., and Wyart, M. (2020). Disentangling feature and lazy training in deep neural networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2020(11):113301. Publisher: IOP Publishing.

Ghorbani, B., Mei, S., Misiakiewicz, T., and Montanari, A. (2019). Limitations of lazy training of two-layers neural network. In *Advances in Neural Information Processing Systems*, pages 9111–9121.

Ghorbani, B., Mei, S., Misiakiewicz, T., and Montanari, A. (2020). When Do Neural Networks Outperform Kernel Methods? *Advances in Neural Information Processing Systems*, 33.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep Residual Learning for Image Recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778. ISSN: 1063-6919.

Huval, B., Wang, T., Tandon, S., Kiske, J., Song, W., Pazhayampallil, J., Andriluka, M., Rajpurkar, P., Migimatsu, T., Cheng-Yue, R., et al. (2015). An empirical evaluation of deep learning on highway driving. *arXiv preprint arXiv:1504.01716*.

Jacot, A., Gabriel, F., and Hongler, C. (2018). Neural tangent kernel: Convergence and generalization in neural networks. In *Proceedings of the 32Nd International Conference on Neural Information Processing Systems*, NIPS'18, pages 8580–8589, USA. Curran Associates Inc.

Kanbak, C., Moosavi-Dezfooli, S.-M., and Frossard, P. (2018). Geometric Robustness of Deep Networks: Analysis and Improvement. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4441–4449, Salt Lake City, UT. IEEE.

Kardar, M. (2007). *Statistical physics of fields*. Cambridge University Press.

Kayhan, O. S. and Gemert, J. C. v. (2020). 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*, pages 14274–14285.

Kopitkov, D. and Indelman, V. (2020). Neural Spectrum Alignment: Empirical Study. *Artificial Neural Networks and Machine Learning – ICANN 2020*.

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

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In Pereira, F., Burges, C. J. C., Bottou, L., and Weinberger, K. Q., editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc.

Le, Q. V. (2013). Building high-level features using large scale unsupervised learning. In *2013 IEEE international conference on acoustics, speech and signal processing*, pages 8595–8598. IEEE.

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

LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., and Jackel, L. D. (1989). Backpropagation Applied to Handwritten Zip Code Recognition. *Neural Computation*, 1(4):541–551.

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. Conference Name: Proceedings of the IEEE.

Lee, J., Schoenholz, S. S., Pennington, J., Adlam, B., Xiao, L., Novak, R., and Sohl-Dickstein, J. (2020). Finite versus infinite neural networks: an empirical study. *arXiv preprint arXiv:2007.15801*.

Loshchilov, I. and Hutter, F. (2016). SGDR: Stochastic Gradient Descent with Warm Restarts.

Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110.

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

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

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., and Riedmiller, M. (2013). Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*.

Novak, R., Bahri, Y., Abolafia, D. A., Pennington, J., and Sohl-Dickstein, J. (2018). Sensitivity and generalization in neural networks: an empirical study. *arXiv preprint arXiv:1802.08760*.

Oymak, S., Fabian, Z., Li, M., and Soltanolkotabi, M. (2019). Generalization guarantees for neural networks via harnessing the low-rank structure of the jacobian. *arXiv preprint arXiv:1906.05392*.

Paccolat, J., Petrini, L., Geiger, M., Tyloo, K., and Wyart, M. (2021a). Geometric compression of invariant manifolds in neural networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2021(4):044001. Publisher: IOP Publishing.

Paccolat, J., Spigler, S., and Wyart, M. (2021b). How isotropic kernels perform on simple invariants. *Machine Learning: Science and Technology*, 2(2):025020. Publisher: IOP Publishing.

Recanatesi, S., Farrell, M., Advani, M., Moore, T., Lajoie, G., and Shea-Brown, E. (2019). Dimensionality compression and expansion in deep neural networks. *arXiv preprint arXiv:1906.00443*.

Refinetti, M., Goldt, S., Krzakala, F., and Zdeborová, L. (2021). Classifying high-dimensional gaussian mixtures: Where kernel methods fail and neural networks succeed. *arXiv preprint arXiv:2102.11742*.

Ruderman, A., Rabinowitz, N. C., Morcos, A. S., and Zoran, D. (2018). Pooling is neither necessary nor sufficient for appropriate deformation stability in CNNs. *arXiv:1804.04438 [cs, stat]*. arXiv: 1804.04438.

Saxe, A. M., Bansal, Y., Dapello, J., Advani, M., Kolchinsky, A., Tracey, B. D., and Cox, D. D. (2019). On the information bottleneck theory of deep learning. *Journal of Statistical Mechanics: Theory and Experiment*, 2019(12):124020.

Shi, B., Bai, X., and Yao, C. (2016). An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition. *IEEE transactions on pattern analysis and machine intelligence*, 39(11):2298–2304.

Shwartz-Ziv, R. and Tishby, N. (2017). Opening the black box of deep neural networks via information. *arXiv preprint arXiv:1703.00810*.

Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A., et al. (2017). Mastering the game of go without human knowledge. *nature*, 550(7676):354–359.

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

Tan, M. and Le, Q. (2019). EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. In *International Conference on Machine Learning*, pages 6105–6114. PMLR. ISSN: 2640-3498.

Tsuzuku, Y. and Sato, I. (2019). On the structural sensitivity of deep convolutional networks to the directions of fourier basis functions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 51–60.

Xiao, C., Zhu, J.-Y., Li, B., He, W., Liu, M., and Song, D. (2018). Spatially Transformed Adversarial Examples.

Xiao, H., Rasul, K., and Vollgraf, R. (2017). Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms. *arXiv:1708.07747 [cs, stat]*. arXiv: 1708.07747.

Yehudai, G. and Shamir, O. (2019). On the power and limitations of random features for understanding neural networks. In *Advances in Neural Information Processing Systems*, pages 6598–6608.

Yin, D., Lopes, R. G., Shlens, J., Cubuk, E. D., and Gilmer, J. (2019). A fourier perspective on model robustness in computer vision. *arXiv preprint arXiv:1906.08988*.

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

A Maximum entropy calculation

Under the constraint on the borders, τ_u and τ_v can be expressed in a real Fourier basis as in Eq.2. By injecting this form into $\|\nabla\tau\|^2$ we obtain:

$$\|\nabla\tau\|^2 = \frac{\pi^2}{4} \sum_{i,j\in\mathbb{N}^+} (C_{ij}^2 + D_{ij}^2)(i^2+j^2) \tag{7}$$

where D_{ij} are the Fourier coefficients of τ_v. We aim at computing the probability distributions that maximize their entropy while keeping the expectation value of $\|\nabla\tau\|^2$ fixed. Since we have a sum of quadratic random variables, the equipartition theorem Beale (1996) applies: the distributions are normal and every quadratic term contributes in average equally to $\|\nabla\tau\|^2$. Thus, the variance of the coefficients follows $\frac{T}{i^2+j^2}$ where the parameter T determines the magnitude of the diffeomorphism.

B Boundaries of studied diffeomorphisms

Average pixel displacement magnitude δ We measure the mean square displacement at the center of the image δ^2 defined in Section 2.2, as shown in Fig.8. If $\delta \ll 1$, our results strongly depend on the choice of interpolation method. To avoid it, we only consider conditions for which $\delta \geq 1/2$, leading to

$$T > \frac{1}{\pi n^2 \log c}. \tag{8}$$

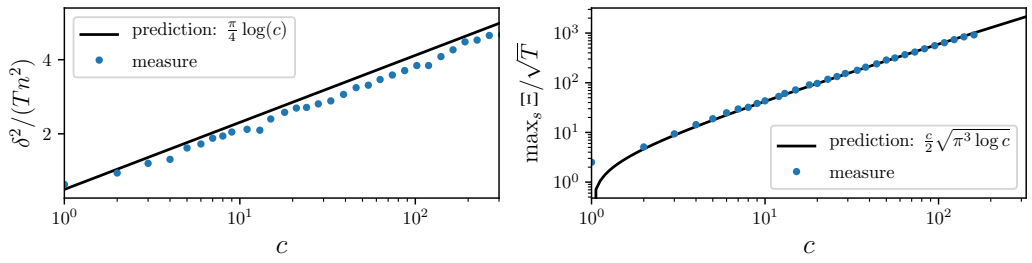

Figure 8: Left: The characteristic displacement $\delta(c,T)$ is observed to follow $\delta^2 \simeq \frac{\pi}{4} n^2 T \log c$. Right: measurement of $\max_s \Xi$ supporting Eq.13.

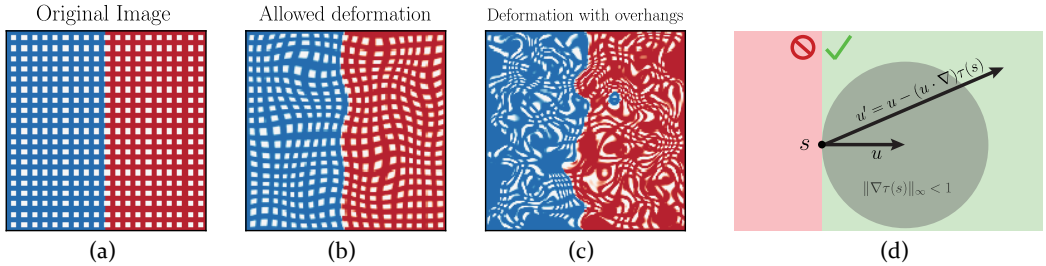

Figure 9: (a) Idealized image at $T = 0$. (b) Diffeomorphism of the image. (c) Deformation of the image at large T: colors get mixed-up together, shapes are not preserved anymore. (d) Allowed region for vector transformations under τ. For any point in the image s and any direction u, only displacement fields for which all the deformed direction u' is non-zero generate diffeomorphisms. The bound in Eq.12 ($u' \cdot u > 0$) correspond to the green region. The gray disc correspond to the bound $\|\nabla\tau\|_\infty < 1$.

Condition for diffeomorphism in the (T,c) plane For a given value of c, there exists a temperature scale beyond which the transformation is not injective anymore, affecting the topology of the image and creating spurious boundaries, see Fig.9a-c for an illustration. Specifically, consider a curve

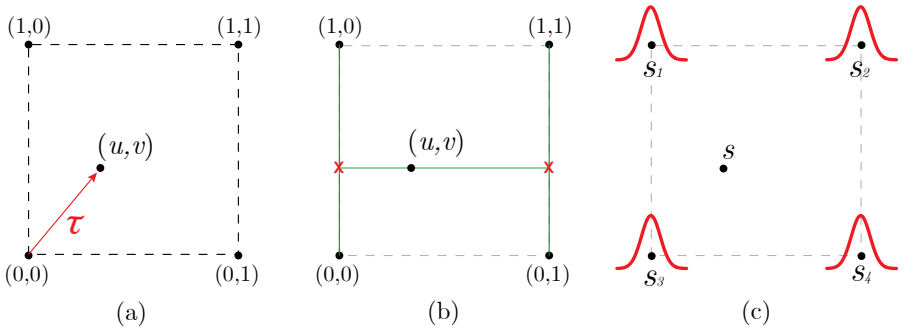

Figure 10: (a) We consider the region between four pixels as the square $[0,1]^2$ where, after the application of a deformation τ, the pixel $(0,0)$ is mapped into (u,v). (b) **Bi-linear interpolation**: the value of x in (u,v) is computed by two steps of linear interpolation. First, we compute x in the red crosses, by averaging values on the vertical axis. Then, a line interpolates horizontally the values in the red crosses to give the result. (c) **Gaussian interpolation**: we denote by s_i the pixel positions in the original grid. The interpolated value of s in any point of the image is given by a weighted sum of $n \times n$ Gaussian centered in each s_i – in red.

passing by the point s in the deformed image. Its tangent direction is u at the point s. When going back to the original image ($s' = s - \tau(s)$) the curve gets deformed and its tangent becomes

$$u' = u - (u \cdot \nabla)\tau(s). \tag{9}$$

A smooth deformation is bijective iff all deformed curves remain curves which is equivalent to have non-zero tangents everywhere

$$\forall\, s, u \neq 0 \quad \|u'\| \neq 0. \tag{10}$$

Imposing $\|u'\| \neq 0$ does not give us any constraint on τ. Therefore, we constraint τ a bit more and allow only displacement fields such that $u \cdot u' > 0$, which is a sufficient condition for Eq.10 to be satisfied – cf. Fig. 9d. By extremizing over u, this condition translates into

$$\tfrac{1}{2}\left(\sqrt{(\partial_x\tau_x - \partial_y\tau_y)^2 + (\partial_x\tau_y + \partial_y\tau_x)^2} - \partial_x\tau_x - \partial_y\tau_y\right) < 1 \tag{11}$$

or, equivalently,

$$\Xi = \tfrac{1}{2}\left(\sqrt{||\nabla\tau||^2 - 2\det(\nabla\tau)} - \mathrm{Tr}(\nabla\tau)\right) < 1, \tag{12}$$

were we identified by Ξ the l.h.s. of the inequality. We find that the median of the maximum of Ξ over all the image ($\|\Xi(s)\|_\infty$) can be approximated by (see Fig.8b):

$$\max_s \Xi \simeq \frac{c}{2}\sqrt{\pi^3 T \log c}. \tag{13}$$

The resulting constraint on T reads

$$T < \frac{4}{\pi^3 c^2 \log c}. \tag{14}$$

C Interpolation methods

When a deformation is applied to an image x, each of its pixels gets mapped, from the original pixels grid, to new positions generally outside of the grid itself – cf. Fig. 9a-b. A procedure (interpolation method) needs to be defined to project the deformed image back into the original grid.

For simplicity of notation, we describe interpolation methods considering the square $[0,1]^2$ as the region in between four pixels – see an illustration in Fig. 10a. We propose here two different ways to interpolate between pixels and then check that our measurements do not depend on the specific method considered.

Bi-linear Interpolation The bi-linear interpolation consists, as the name suggests, of two steps of linear interpolation, one on the horizontal, and one on the vertical direction – Fig. 10b. If we look at the square $[0,1]^2$ and we apply a deformation τ such that $(0,0) \mapsto (u,v)$, we have

$$x(u,v) = x(0,0)(1-u)(1-v) + x(1,0)u(1-v) + x(0,1)(1-u)v + x(1,1)uv. \tag{15}$$

Gaussian Interpolation In this case, a Gaussian function[4] is placed on top of each point in the grid – cf. Fig.10. The pixel intensity x can be evaluated at any point outside the grid by computing

$$x(s) = \frac{\sum_i x(s_i) G(s - s_i)}{\sum_i G(s - s_i)}. \tag{16}$$

In order to fix the standard deviation σ of G, we introduce the *participation ratio* n. Given $\Psi_i = G(s, s_i)|_{s=(0.5,0.5)}$, we define

$$n = \frac{\left(\sum_i \Psi_i^2\right)^2}{\sum_i \Psi_i^4}. \tag{17}$$

The participation ratio is a measure of how many pixels contribute to the value of a new pixel, which results from interpolation. We fix σ in such a way that the participation ratio for the Gaussian interpolation matches the one for the bi-linear ($n = 4$), when the new pixel is equidistant from the four pixels around. This gives $\sigma = 0.4715$.

Notice that this interpolation method is such that it applies a Gaussian smoothing of the image even if τ is the identity. Consequently, when computing observables for f with the Gaussian interpolation, we always compare $f(\tau x)$ to $f(\tilde{x})$, where $\tilde{x}$ is the smoothed version of x, in such a way that $f(\tau^{[T=0]} x) = f(\tilde{x})$.

Empirical results dependence on interpolation Finally, we checked to which extent our results are affected by the specific choice of interpolation method. In particular, blue and red colors in Figs3, 13 correspond to bi-linear and Gaussian interpolation, respectively. The interpolation method only affects the results in the small displacement limit ($\delta \to 0$).

Note: throughout the paper, if not specified otherwise, bi-linear interpolation is employed.

D Stability to additive noise *vs.* noise magnitude

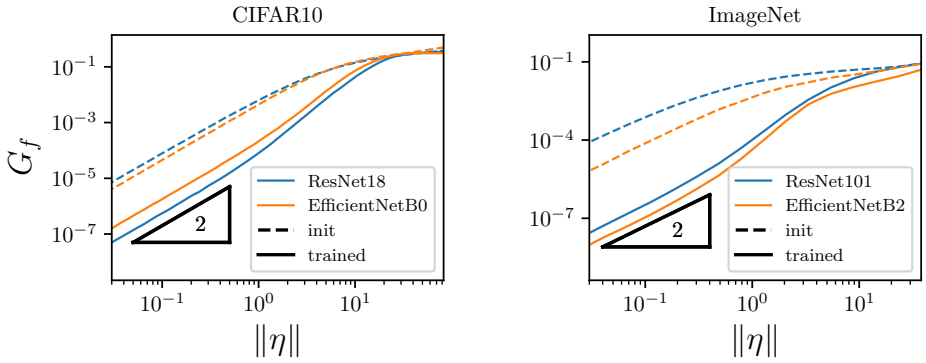

Figure 11: **Stability to isotropic noise** G_f as a function of the noise magnitude $\|\eta\|$ for CIFAR10 (left) and ImageNet (right). The color corresponds to two different classes of SOTA architecture: ResNet and EfficientNet. The slope 2 at small $\|\eta\|$ identifies the linear regime. For larger noise magnitudes, non-linearities appear.

We introduced in Section 5 the stability toward additive noise:

$$G_f = \frac{\langle \|f(x+\eta) - f(x)\|^2 \rangle_{x,\eta}}{\langle \|f(x) - f(z)\|^2 \rangle_{x,z}}. \tag{18}$$

We study here the dependence of G_f on the noise magnitude $\|\eta\|$. In the $\eta \to 0$ limit, we expect the network function to behave as its first-order Taylor expansion, leading to $G_f \propto \|\eta\|^2$. Hence, for small noise, G_f gives an estimate of the average magnitude of the gradient of f in a random direction η.

[4] $G(s) = (2\pi\sigma^2)^{-1/2} e^{-s^2/2\sigma^2}$.

Empirical results Measurements of G_f on SOTA nets trained on benchmark data-sets are shown in Figure 11. We observe that the effect of non-linearities start to be significant around $\|\eta\| = 1$. For large values of the noise – i.e. far away from data-points – the average gradient of f does not change with training.

Relative stability at $\delta = 1$ is computed in the linear region of $G_f(\|\eta\|)$ When comparing random transformations to diffeomorphisms at $\delta = 1$ we have $\|\eta\| = \langle \|\tau x - x\| \rangle \approx 10^{-1}$, corresponding to G_f within its linear regime.

E Numerical experiments

In this Appendix, we provide details on the training procedure, on the different architectures employed and some additional experimental results.

E.1 Image classification training set-up:

- Trainings are performed in `PyTorch`, the code can be found here github.com/leonardopetrini/diffeo-sota.
- Loss function: cross-entropy.
- Batch size: 128.
- Dynamics:
  - Fully connected nets: ADAM with `learning rate` $= 0.1$ and no scheduling.
  - Transfer learning: SGD with `learning rate` $= 10^{-2}$ for the last layer and 10^{-3} for the rest of the network, `momentum` $= 0.9$ and `weight decay` $= 10^{-3}$. Both learning rates decay exponentially during training with a factor $\gamma = 0.975$.
  - All the other networks are trained with SGD with `learning rate` $= 0.1$, `momentum` $= 0.9$ and `weight decay` $= 5 \times 10^{-4}$. The learning rate follows a cosine annealing scheduling Loshchilov and Hutter (2016).
- Early-stopping is performed – i.e. results shown are computed with the network obtaining the best validation accuracy out of 250 training epochs.
- For the experiments involving a training on a subset of the training date of size $P < P_{\max}$, the total number of epochs is accordingly re-scaled in order to keep constant the total number of optimizer steps.
- Standard data augmentation is employed: different random translations and horizontal flips of the input images are generated at each epoch. As a safety check, we verify that the invariance learnt by the nets is not purely due to such augmentation (Fig.12).
- Experiments are run on 16 GPUs NVIDIA V100. Individual trainings run in ~ 1 hour of wall time. We estimate a total of a few thousands hours of computing time for running the preliminary and actual experiments present in this work.

The stripe model is trained with an approximation of gradient flow introduced in Geiger et al. (2020), see Paccolat et al. (2021a) for details.

A note on computing stabilities at init. in presence of batch-norm We recall that batch-norm (BN) can work in either of two modes: *training* and *evaluation*. During training, BN computes the mean and variance on the current batch and uses them to normalize the output of a given layer. At the same time, it keeps memory of the running statistics on such batches, and this is used for the normalization steps at inference time (evaluation mode). When probing a network at initialization for computing stabilities, we put the network in evaluation mode, except for batch-norm (BN), which operates in train mode. This is because BN running mean and variance are initialized to 0 and 1, in such a way that its evaluation mode at initialization would correspond to not having BN at all, compromising the input signal propagation in deep architectures.

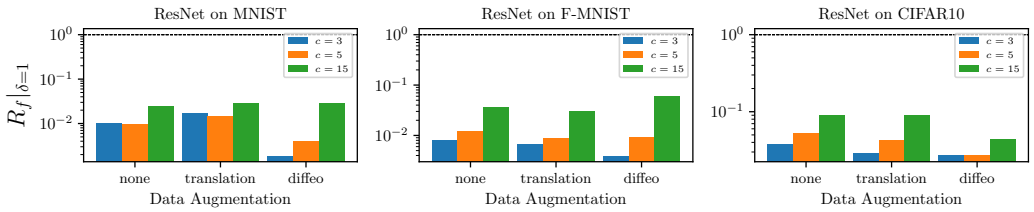

Figure 12: **Effect of data augmentation on R_f.** Relative stability to diffeomorphisms R_f after training with different data augmentations: "none" (1st group of bars in each plot) for no data augmentation, "translation" (2nd bars) corresponds to training on randomly translated (by 4 pixels) and cropped inputs, and "diffeo" (3rd bars) to training on randomly deformed images with max-entropy diffeomorphisms ($T = 10^{-2}$, $c = 1$). Results are averaged over 5 trainings of ResNet18 on MNIST (left), FashionMNIST (center), CIFAR10 (right). Colors indicate different cut-off values when probing the trained networks. Different augmentations have a small quantitative, and no qualitative effect on the results. As expected, augmenting the input images with smooth deformations makes the net more invariant to such transformations.

E.2 Networks architectures

All networks implementations can be found at github.com/leonardopetrini/diffeo-sota/tree/main/models. In Table 1, we report salient features of the network architectures considered.

Table 1: **Network architectures, main characteristics.** We list here (columns) the classes of net architectures used throughout the paper specifying some salient features (depth, number of parameters, etc...) for each of them.

features	**FullConn**	**LeNet** LeCun et al. (1989)	**AlexNet** Krizhevsky et al. (2012)
depth	2, 4, 6	5	8
num. parameters	200k	62k	23 M
FC layers	2, 4, 6	3	3
activation	ReLU	ReLU	ReLU
pooling	/	max	max
dropout	/	/	yes
batch norm	/	/	/
skip connections	/	/	/

features	**VGG** Simonyan and Zisserman (2015)	**ResNet** He et al. (2016)	**EfficientNetB0-2** Tan and Le (2019)
depth	11, 16, 19	18, 34, 50	18, 25
num. parameters	9-20 M	11-24 M	5, 9 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)

E.3 Additional figures

We present here:

- Fig.13: R_f as a function of P for MNIST and FashionMNIST with the corresponding predicted slope, omitted in the main text.
- Fig.14: Relative diffeomorphisms stability R_f as a function of depth for simple and deep nets.
- Figs15,16: diffeomorphisms and inverse of the Gaussian stability D_f and $1/G_f$ *vs.* test error for CIFAR10 and the set of architectures considered in Section 4.
- Fig.17: D_f, $1/G_f$ and R_f when using the mean in place of the median for computing averages $\langle\cdot\rangle$.
- Fig.18: curves in the (ϵ_t, R_f) plane when varying the training set size P for FullyConnL4, LeNet, ResNet18 and EfficientNetB0.
- Figs19, 20: error estimates for the main quantities of interest – often omitted in the main text for the sake of figures' clarity.

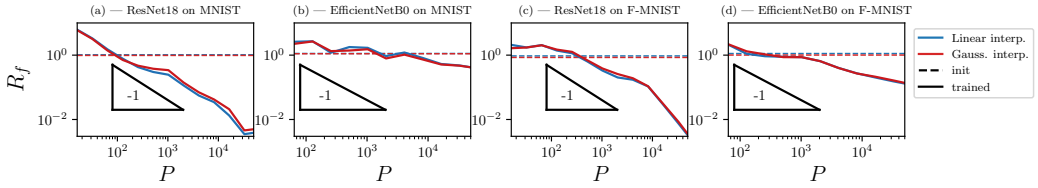

Figure 13: **Relative stability to diffeomorphisms $R_f(P)$ at $\delta = 1$.** Analogous to Figure 3-right but here we have MNIST (a-b) and FashionMNIST (c-d) in place of CIFAR10. Stability monotonically decreases with P. The triangles give a reference for the predicted slope in the stripe model – i.e. $R_f \sim P^{-1}$ – see Section 6. The slopes in case of ResNets are compatible with the prediction. For EfficientNets, the second panel of Fig.3 suggests that stability to diffeomorphisms is less important. Here, we also see that it builds up more slowly when increasing the training set size. Finally, blue and red colors indicate different interpolation methods used for generating image deformations, as discussed in Appendix C. Results are not affected by this choice.

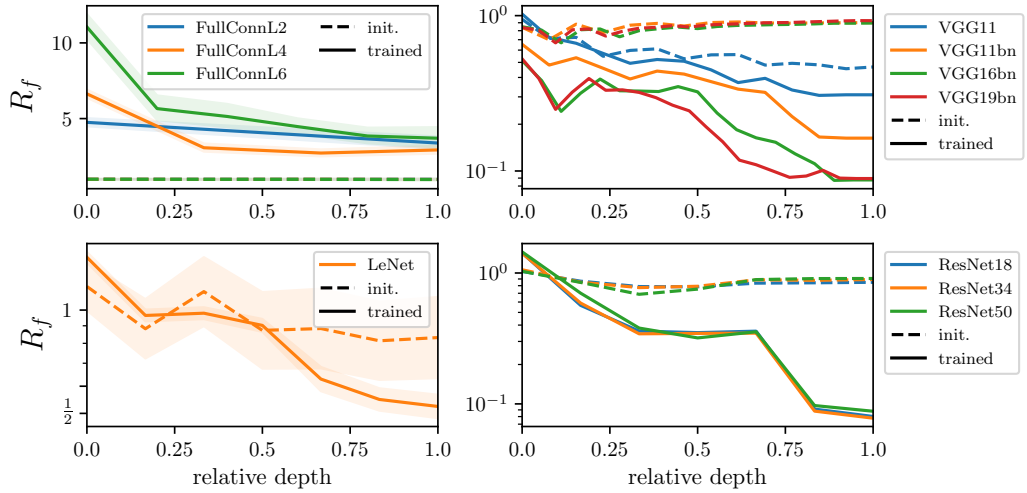

Figure 14: **Relative stability to diffeomorphisms as a function of depth.** R_f as a function of the layers relative depth (i.e. $\frac{\text{current layer depth}}{\text{total depth}}$) where "0" identifies the output of the 1st layer and "1" the last. The relative stability is measured for the output of layers (or blocks of layers) inside the nets for simple architectures (1st column) and deep ones (2nd column) at initialization (dashed) and after training (full lines). All nets are trained on the full CIFAR10 dataset. $R_{f_0} \approx 1$ independently of depth at initialization while it decreases monotonically as a function of depth after training. *Statistics:* Each point is obtained by training 5 differently initialized networks; each network is then probed with 500 test samples in order to measure R_f. The results are obtained by log-averaging over single realizations.

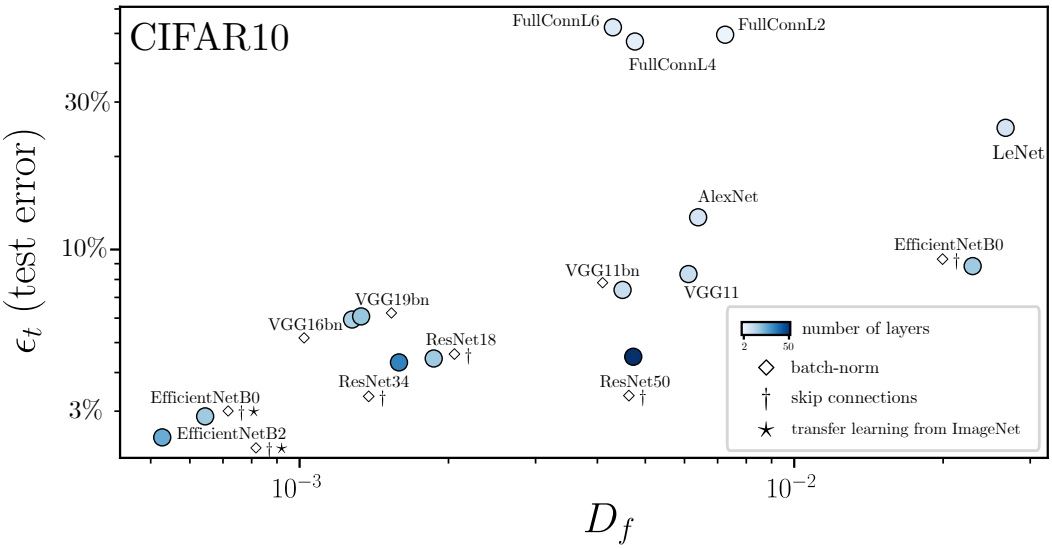

Figure 15: **Test error ϵ_t *vs.* stability to diffeomorphisms** D_f for common architectures when trained on the full 10-classes CIFAR10 dataset ($P = 50k$) with SGD and the cross-entropy loss; the EfficientNets achieving the best performance are trained by transfer learning from ImageNet ($\star$) – more details on the training procedures can be found in Appendix E.1. The color scale indicates depth, and the symbols the presence of batch-norm ($\diamond$) and skip connections ($\dagger$). D_f correlation with ϵ_t (corr. coeff.: 0.62) is much smaller than the one measured for R_f – see Fig.3. *Statistics:* Each point is obtained by training 5 differently initialized networks; each network is then probed with 500 test samples in order to measure D_f. The results are obtained by log-averaging over single realizations.

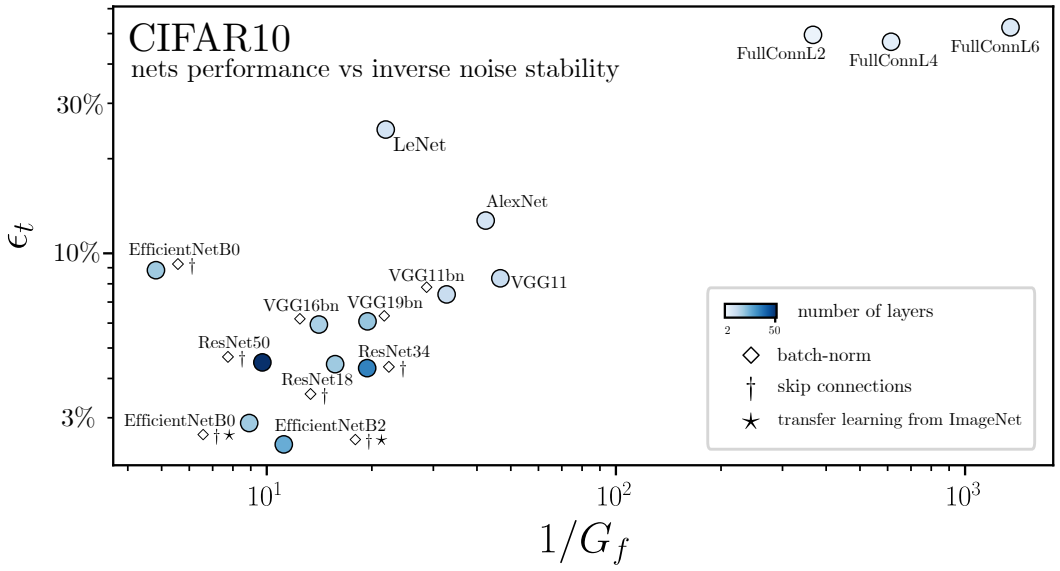

Figure 16: **Test error ϵ_t *vs.* inverse of stability to noise** $1/G_f$ for common architectures when trained on the full 10-classes CIFAR10 dataset ($P = 50k$) with SGD and the cross-entropy loss; the EfficientNets achieving the best performance are trained by transfer learning from ImageNet ($\star$) – more details on the training procedures can be found in Appendix E.1. The color scale indicates depth, and the symbols the presence of batch-norm ($\diamond$) and skip connections ($\dagger$). G_f correlation with ϵ_t (corr. coeff.: 0.85) is less important than the one measured for R_f – see Fig.3. *Statistics:* Each point is obtained by training 5 differently initialized networks; each network is then probed with 500 test samples in order to measure G_f. The results are obtained by log-averaging over single realizations.

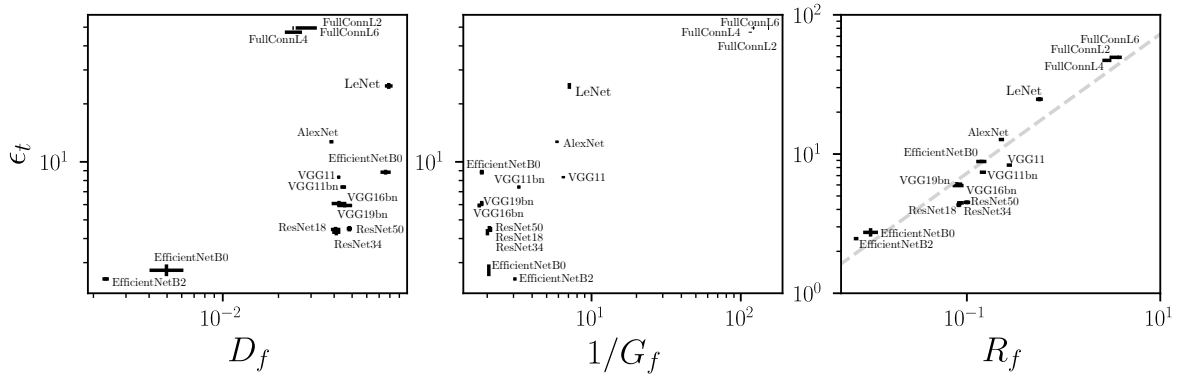

Figure 17: **Test error ϵ_t *vs.* D_f, $1/G_f$ and R_f where $\langle\cdot\rangle$ is the mean.** Analogous to Figs15-19, we use here the mean instead of the median to compute averages over samples and transformations.

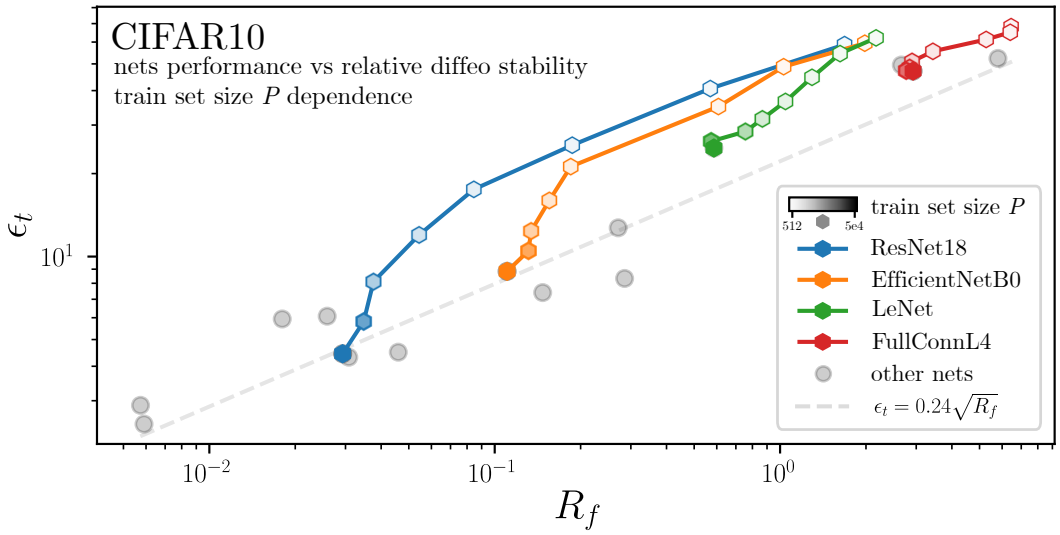

Figure 18: **Test error ϵ_t *vs.* relative stability to diffeomorphisms R_f for different training set sizes P.** Same data as Fig.5, we report here curves corresponding to training on different set sizes for 4 architectures. The other architectures considered together with the power-law fit are left in background. For a small training set, CNNs behave similarly. *Statistics:* Each point is obtained by training 5 differently initialized networks; each network is then probed with 500 test samples in order to measure R_f. The results are obtained by log-averaging over single realizations.

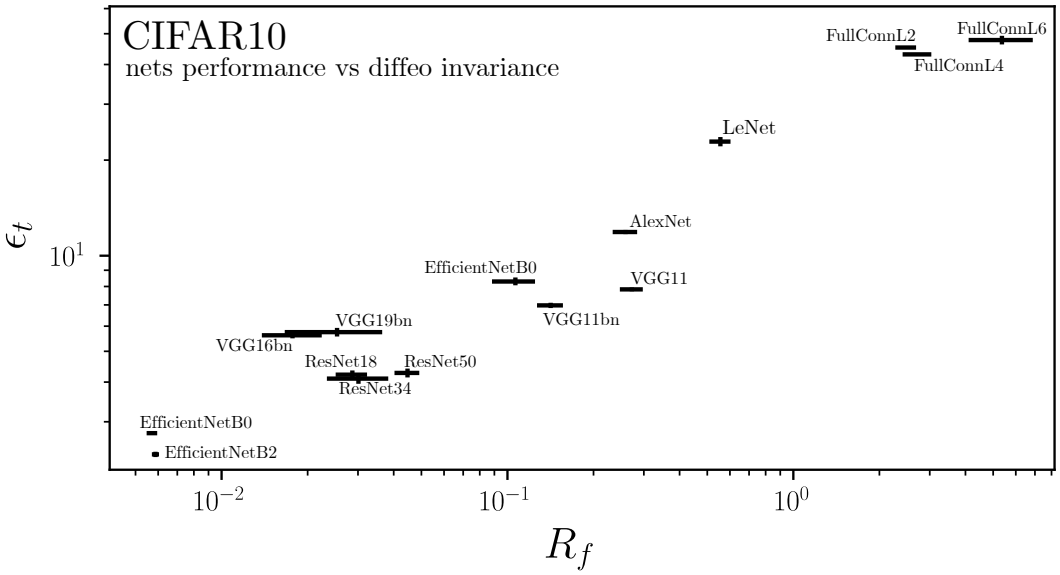

Figure 19: **Test error ϵ_t *vs.* relative stability to diffeomorphisms R_f with error estimates.** Same data as Fig.5, we report error bars here. *Statistics:* Each point is obtained by training 5 differently initialized networks; each network is then probed with 500 test samples in order to measure R_f. The results are obtained by log-averaging over single realizations.

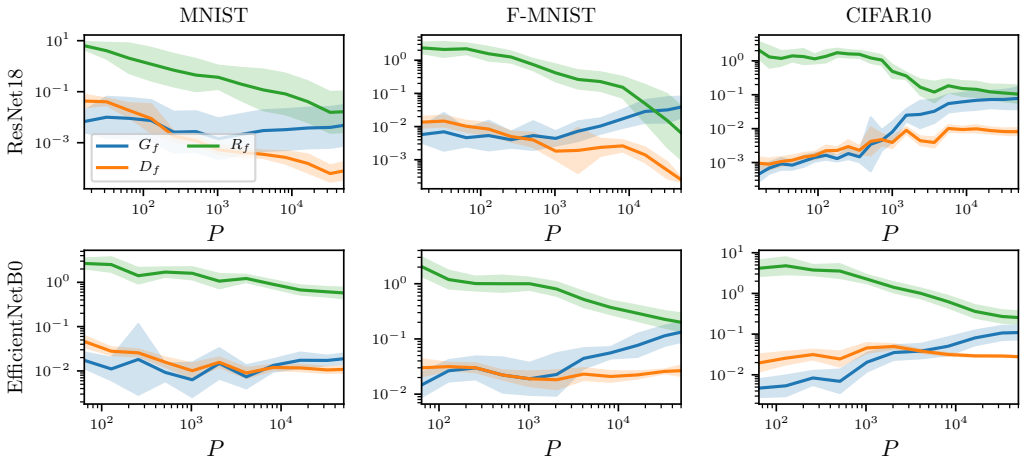

Figure 20: **Stability toward Gaussian noise (G_f) and diffeomorphisms (D_f) alone, and the relative stability R_f with the relative errors.** Analogous to Fig.6 in which error estimates are omitted to favour clarity. Here we fix the cut-off to $c = 3$ and show error estimates instead. Columns correspond to different data-sets (MNIST, FashionMNIST and CIFAR10) and rows to architectures (ResNet18 and EfficientNetB0). Each panel reports G_f (blue), D_f (orange) and R_f (green) as a function of P and for different cut-off values c, as indicated in the legend. *Statistics*: Each point in the graphs is obtained by training 16 differently initialized networks on 16 different subsets of the data-sets; each network is then probed with 500 test samples in order to measure stability to diffeomorphisms and Gaussian noise. The resulting R_f is obtained by log-averaging the results from single realizations. As we are plotting quantities in log scale, we report the relative error (shaded).