HOW DEEP CONVOLUTIONAL NEURAL NETWORKS LOSE SPATIAL INFORMATION WITH TRAINING

Anonymous authors
Paper under double-blind review

ABSTRACT

A central question of machine learning is how deep nets learn tasks in high dimensions. An appealing hypothesis is that they build a representation of the data where information irrelevant to the task is lost. For image datasets, this view is supported by the observation that after (and not before) training, the neural representation becomes less and less sensitive to diffeomorphisms acting on images as the signal propagates through the net. This loss of sensitivity correlates with performance and surprisingly correlates with a *gain* of sensitivity to white noise acquired over training. These facts are unexplained, and as we demonstrate still hold when white noise is added to the images of the training set. Here we *(i)* show empirically for various architectures that stability to diffeomorphisms is achieved due to a combination of spatial and channel pooling; *(ii)* introduce a model scale-detection task which reproduces our empirical observations on spatial pooling; *(iii)* compute analytically how the sensitivity to diffeomorphisms and noise scale with depth due to spatial pooling. In particular, we find that both trends are caused by a diffusive spreading of the neuron's receptive fields through the layers.

1 INTRODUCTION

Deep learning algorithms can be successfully trained to solve a large variety of tasks (Amodei et al., 2016; Huval et al., 2015; Mnih et al., 2013; Shi et al., 2016; Silver et al., 2017), often revolving around classifying data in high-dimensional spaces. If there was little structure in the data, the learning procedure would be cursed by the dimension of these spaces: achieving good performances would require an astronomical number of training data (Luxburg & Bousquet, 2004). Consequently, real datasets must have a specific internal structure that can be learned with fewer examples. It has been then hypothesized that the effectiveness of deep learning lies in its ability of building 'good' representations of this internal structure, which are insensitive to aspects of the data not related to the task (Ansuini et al., 2019; Shwartz-Ziv & Tishby, 2017; Recanatesi et al., 2019), thus effectively reducing the dimensionality of the problem.

In the context of image classification, Bruna & Mallat (2013); Mallat (2016) proposed that neural networks lose irrelevant information by learning representations that are insensitive to small deformations of the input, also called diffeomorphisms. This idea was tested in modern deep networks by Petrini et al. (2021), who introduced the following measures

$$D_f = \frac{\mathbb{E}_{x,\tau}\|f(\tau(x)) - f(x)\|^2}{\mathbb{E}_{x_1,x_2}\|f(x_1) - f(x_2)\|^2}, \qquad G_f = \frac{\mathbb{E}_{x,\eta}\|f(x+\eta) - f(x)\|^2}{\mathbb{E}_{x_1,x_2}\|f(x_1) - f(x_2)\|^2}, \qquad R_f = \frac{D_f}{G_f}, \tag{1}$$

to probe the sensitivity of a function f—either the output or an internal representation of a trained network—to random diffeomorphisms τ of x, to large white noise perturbations η of magnitude $\|\tau(x) - x\|$, and in relative terms, respectively. Here the input images x, x_1 and x_2 are sampled uniformly from the test set. In particular, the test error of trained networks is correlated with D_f when f is the network output. Less intuitively, the test error is anti-correlated with the sensitivity to white noise G_f. Overall, it is the relative sensitivity R_f which correlates best with the error. This correlation is learned over training—as it is not seen at initialization—and built up layer by layer (Petrini et al., 2021).

Operations that grant insensitivity to diffeomorphisms in a deep network have been identified previously (e.g. Goodfellow et al. (2016), section 9.3, sketched in Figure 1). The first, *spatial* pooling,

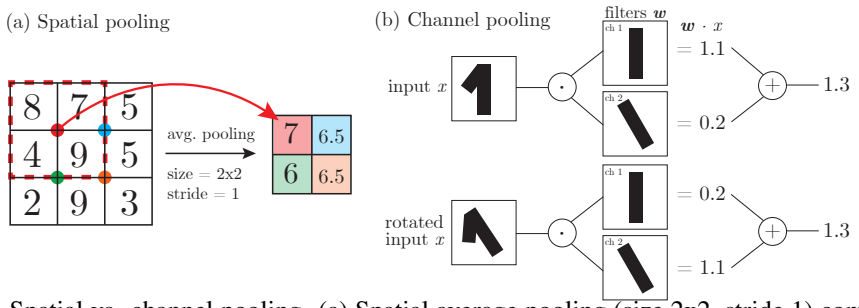

Figure 1: Spatial vs. channel pooling. (a) Spatial average pooling (size 2x2, stride 1) computed on a representation of size 3x3. One can notice that nearby pixel variations are smaller after pooling. (b) If the filters of different channels are identical up to e.g. a rotation of angle θ, then, averaging the output of the application of such filters makes the result invariant to input rotations of θ. This averaging is an example of channel pooling.

integrates local patches within the image, thus losing the exact location of its features. The second, *channel* pooling, requires the interaction of different channels, which allows the network to become invariant to any local transformation by properly learning filters that are transformed versions of one another. However, it is not clear whether these operations are actually learned by deep networks and how they conspire in building good representations. In this work we tackle this question by unveiling empirically the emergence of spatial and channel pooling and disentangling their role. We then focus on spatial pooling and show quantitatively, with the aid of an idealized scale-detection task, how its emergence relates to the behavior of the sensitivities. The code and details for reproducing experiments are available online at https://tinyurl.com/cnn-spatial-experiments.

2 Empirical observations on real data

First, we observe that the relative sensitivity to diffeomorphisms R_k of the k-th layer representation f_k decreases after each layer, as shown in Figure 2. This implies that spatial or channel pooling are carried out along the whole network. To disentangle their contribution we perform the following experiment: shuffle at random the connections between channels of successive convolutional layers, while keeping the weights unaltered. This procedure breaks any channel pooling while not affecting single filters. The values of R_k for deep networks after channel shuffling are reported in Figure 2 as dashed lines and compared with the original values of R_k in full lines. If only spatial pooling was present in the network, then the two curves would overlap. Conversely, if the decrease in R_k was all due to the interactions between channels, then the shuffled curves should be constant. Given that neither of these scenarios arises, we conclude that both kinds of pooling are being performed.

To bolster the evidence for the presence of spatial pooling, we analyze the filters of trained networks. Since spatial pooling can be built by having homogeneous filters, we test for its presence by looking at the frequency content of learned convolutional filters $\boldsymbol{w}^k$ at a given layer k, which are $F \times F$ matrices, with F the filter size. In particular, we consider the average squared projection of filters onto "Fourier modes" $\{\Psi_l\}_{l=1,\dots,F^2}$, taken as the eigenvectors of the discrete Laplace operator on the $F \times F$ filter grid. The square projections averaged over channels read

$$\gamma_{k,l} = \langle \left[\Psi_l \cdot \boldsymbol{w}^k\right]^2 \rangle_{channels}, \tag{2}$$

and are shown in Figure 3, 1st and 2nd row. When training a deep network such as VGG11 (with and without batch-norm) (Simonyan & Zisserman, 2015) on CIFAR10, filters of layers 2 to 6 become low-frequency with training.

3 A simple scale-detection task captures real-data observations

In order to isolate the contribution of spatial pooling and quantify its relation with the sensitivities to diffeomorphisms and noise, we introduce an idealized scale-detection task, where data are made of two active pixels and classified comparing the euclidean distance d between the two active pixels and some *characteristic scale* ξ. Namely, the label is $y = \text{sign}\,(\xi - d)$. A small diffeomorphism

of such images corresponds to a small displacement of the active pixels, which does not affect the label of the input. Therefore, we expect that a neural network trained on these tasks will lose any information on the exact location of the active pixels within the image, thus becoming insensitive to diffeomorphisms. Intuitively, spatial pooling up to the scale ξ is the most direct mean to achieve such insensitivity. The result of the integration depends on whether none, one or both the active pixels lie within the pooling window, thus it is still informative of the task. We will show empirically that this is indeed the solution reached by trained CNNs.

Firstly, we find the same correlations between test error and sensitivities of trained networks as found in Petrini et al. (2021): the test error correlates with sensitivity to diffeomorphisms and the anti-correlates with sensitivity to Gaussian noise (Figure 5, in Appendix C). Secondly, the internal representations of trained networks f_k become progressively insensitive to diffeomorphisms and sensitive to Gaussian noise through the layers, as shown in Figure 6 of Appendix C. Importantly, the curves relating sensitivities to the relative depth remain essentially unaltered if the channels of the networks are shuffled (shown as dashed lines in Figure 6). We conclude that, on the one hand channel pooling is negligible, and, on the other hand, all channels are approximately equal to the mean channel. Finally, direct inspection of the filters (Figure 3, bottom row) shows that the 0-frequency component grows much larger than the others over training for layers 1-7, which are the layers where R_k decreases the most in Figure 6. Filters are thus becoming nearly homogeneous, which means that the convolutional layers become effectively pooling layers (see Table 1 of Appendix B.).

4 Theoretical analysis of sensitivities in the scale-detection task

Since the neurons of CNNs trained on the scale-detection task behave effectively as a single pooling filter, we can understand quantitatively how the internal representations of the network become progressively more insensitive to diffeomorphisms and sensitive to Gaussian noise. We consider simple CNNs, made by stacking convolutional layers with filter size F, stride s and ReLU activation function, and we assume that the first few layers display a homogeneous positive filter and no bias. In this setting, the sensitivities D_k and G_k of the net representation f_k at the k-th hidden layer follow $G_k \sim A_k$ and $D_k \sim A_k^{-2}$, where A_k is the effective receptive field size of f_k. We verify empirically our predictions in Figure 4. Notice that the behaviour of the sensitivities as a function of depth does not change if all the filters at a given layer are replaced with their average (compare solid and dotted blue curves in Figure 4), confirming the assumption that all channels behave like the mean channel.

The mechanism underlying the layer-by-layer growth of the effective size A_k for CNNs of filter size F and stride 1 is essentially a diffusion process. Intuitively, applying a homogeneous filter to a representation is equivalent to making each pixel diffuse, i.e. distributing its intensity uniformly over a neighborhood of size F. With a single-pixel input δ_i located in location i, the effective receptive field of the k-th layer $f_k(\delta_i)$ is equivalent to a k-step diffusion of the pixel, thus it approaches a Gaussian distribution of standard deviation $\sqrt{k}$ centered at i. The size A_k is the standard deviation, thus $A_k \sim \sqrt{k}$. A detailed proof is presented in Appendix A.

Since the effective size A_k increases with k, the representation f_k becomes less and less sensitive to small displacements of the input active pixels, hence to diffeomorphisms, inducing the decrease of D_k. To analyze the sensitivity to Gaussian noise G_k, one must take into account the rectifying action of ReLU, which sets all the negative elements of its input to zero, including the zero-mean noise η. As the effective size A_k of the representation increases diffusively, the rectified noise piles up, causing an increase of G_k with k.

5 Conclusion

The meaning of an image often depends on local features, as evidenced by the fact that artists only need a small number of strokes to represent a visual scene. The exact locations of the features are not important in determining the image class, and indeed diffeomorphisms of limited magnitude leave the class unchanged. Here, we have shown that such invariance is learned in deep networks by performing spatial pooling and channel pooling. Modern neural networks learn these pooling operations—as they are not imposed by the architecture—suggesting that it is best to let the pooling

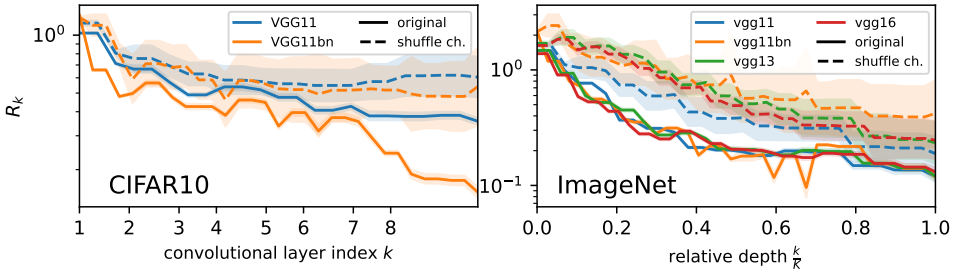

Figure 2: Relative sensitivity R_k as a function of depth for VGG architectures trained on CIFAR10 (left) and ImageNet (right). Full lines refer to the original networks, dashed lines to the ones with shuffled channels. K is the total depth of the networks. Experiments with additional architectures are reported in Appendix C, Figure 7.

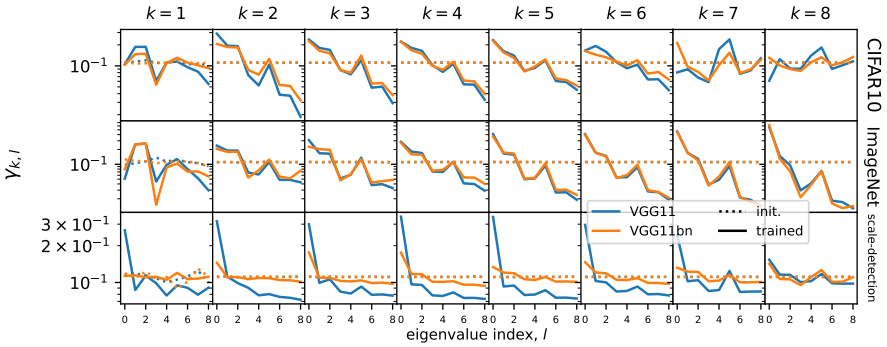

Figure 3: Projections of the filters onto the 9 eigenvectors of the (3×3)-grid Laplacian for VGG11 and VGG11bn at initialization (dotted) and trained (solid) on CIFAR10 (1st row), ImageNet, (2nd row) and the scale-detection task (3rd row). The x-axis reports low to high frequencies from left to right. The depth k of the representation increases from left to right.

adapt to the specific task considered. Interestingly, spatial pooling comes together with an increased sensitivity to random noise in the image, as captured in our simple artificial model of data.

It is commonly believed that the best architectures are those that extract the data features which are the most relevant for the task. The pooling operations studied here, which allow the network to forget the exact locations of these features, are probably more effective when features are better extracted. This point may be responsible for the observed strong correlations between the network performance and its stability to diffeomorphisms. Designing synthetic models of data whose features are combinatorial and stable to smooth transformations is very much needed to clarify this relationship, and ultimately understand how deep networks learn high-dimensional tasks with limited data.

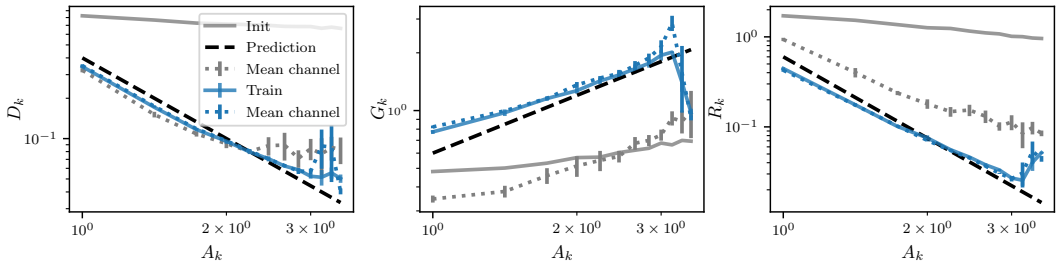

Figure 4: Sensitivities of internal representations f_k of simple CNNs with $s = 1$ and $F = 3$ against the k-th layer receptive field size A_k, at initialization (solid grey) and after training on the scale-detection task (solid blue). The sensitivities obtained by replacing each layer with the mean channel (blue dotted) overlap with the original sensitivities in the first part of the network.

REFERENCES

Dario Amodei, Sundaram Ananthanarayanan, Rishita Anubhai, Jingliang Bai, Eric Battenberg, Carl Case, Jared Casper, Bryan Catanzaro, Qiang Cheng, Guoliang Chen, et al. Deep speech 2: End-to-end speech recognition in english and mandarin. In *International conference on machine learning*, pp. 173–182, 2016.

Alessio Ansuini, Alessandro Laio, Jakob H Macke, and Davide Zoccolan. Intrinsic dimension of data representations in deep neural networks. In *Advances in Neural Information Processing Systems*, pp. 6111–6122, 2019.

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

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. The MIT Press, Cambridge, Massachusetts, November 2016. ISBN 978-0-262-03561-3.

Brody Huval, Tao Wang, Sameep Tandon, Jeff Kiske, Will Song, Joel Pazhayampallil, Mykhaylo Andriluka, Pranav Rajpurkar, Toki Migimatsu, Royce Cheng-Yue, et al. An empirical evaluation of deep learning on highway driving. *arXiv preprint arXiv:1504.01716*, 2015.

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

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

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

Leonardo Petrini, Alessandro Favero, Mario Geiger, and Matthieu Wyart. Relative stability toward diffeomorphisms indicates performance in deep nets. In *Advances in Neural Information Processing Systems*, volume 34, pp. 8727–8739. Curran Associates, Inc., 2021. URL `https://proceedings.neurips.cc/paper/2021/hash/497476fe61816251905e8baafdf54c23-Abstract.html`.

Stefano Recanatesi, Matthew Farrell, Madhu Advani, Timothy Moore, Guillaume Lajoie, and Eric Shea-Brown. Dimensionality compression and expansion in deep neural networks. *arXiv preprint arXiv:1906.00443*, 2019.

Hannes Risken. *The Fokker-Planck Equation Springer Series in Synergetics*. 1996.

Laurent Saloff-Coste and Pierre Bremaud. Markov chains: Gibbs fields, monte carlo simulation, and queues. *Journal of the American Statistical Association*, 95, 2000. ISSN 01621459. doi: 10.2307/2669802.

Baoguang Shi, Xiang Bai, and Cong Yao. 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, 2016.

Ravid Shwartz-Ziv and Naftali Tishby. Opening the black box of deep neural networks via information. *arXiv preprint arXiv:1703.00810*, 2017.

David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of go without human knowledge. *nature*, 550(7676):354–359, 2017.

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

APPENDIX

A STRIDE 1: PROOFS

In section 4 we consider a simple CNN with stride $s = 1$ and filter size F trained on scale-detection task 1. We fix the total depth of these networks to be $\tilde{K}$. We postulated in Sec. 4 that this network displays a one-channel solution with homogeneous filter $[1/F, ..., 1/F]$ and no bias. We can understand the representation $f_k(x)$ at layer k of an input datum x by using single-pixel inputs δ_i. Let us recall that these inputs have all components to 0 except the i-th, set to 1. Then, we have that a general datum x is given by $x \propto (\delta_i + \delta_j)$, where i and j are the locations of the active pixel in x. We have argued in the main text that the representation $f_k(\delta_i)$ is a Gaussian distribution with width $\sqrt{k}$. In this Appendix we prove this statement.

First, we observe that in this solution, since both the elements of the filters and those of the inputs are non-negative, the networks behaves effectively as a linear operator. In particular, each layer corresponds to the application of a $L \times L$ circulant matrix M, which is obtained by stacking all the L shifts of the following row vector,

$$[\underbrace{1, 1, ..., 1}_{F} \underbrace{0, 0, 0, ..., 0}_{L-F}]. \tag{3}$$

with periodic boundary conditions. The first row of such a matrix is fixed as follows. If F is odd the patch of size F is centered on the first entry of the first row, while if F is even we choose to have $(F/2)$ ones at left of the first entry and $(F/2) - 1$ at its right. The output f_k of the layer k is then the following: $f_k(\delta_i) = M^k \delta_i$.

Proposition A.1. *Let's consider the $L \times L$ matrix M and a given L vector δ_i, as defined above. For odd $F \geq 3$, in the limit of large depth $\tilde{K} \gg 1$ and large width $\tilde{L} \gg F\sqrt{\tilde{K}}$, we have that*

$$\begin{aligned} (M^k)_{ab}\delta_i &= \frac{1}{2\sqrt{\pi}\sqrt{D^{(1)}}\sqrt{k}} e^{-\frac{(a-i)^2}{4D^{(1)}k}}, \\ D^{(1)} &= \frac{1}{12F}(F-1)^3, \end{aligned} \tag{4}$$

while for even F:

$$\begin{aligned} (M^k)_{ab}\delta_i &= \frac{1}{2\sqrt{\pi}\sqrt{D^{(2)}}\sqrt{k}} e^{-\frac{(v^{(2)}k+a-i)^2}{4D_F^{(2)}k}}, \\ D^{(2)} &= \frac{1}{12F}\left(F^3 - 3F^2 + 6F - 4\right), \end{aligned} \tag{5}$$

with $v^{(2)} = (1-F)/(2F)$.

Proof. The matrix M can be seen as the stochastic matrix of a Markov process, where at each step the random walker has uniform probability $1/F$ to move in a patch of width F around itself. We write the following recursion relation for odd F,

$$\begin{aligned} p_{a,i}^{(k+1)} = \frac{1}{F}\Big(&p_{a-(F-1)/2,i}^{(k)} + ... \\ &+ p_{a,i}^{(k)} + ... + p_{a+(F-1)/2,i}^{(k)}\Big), \end{aligned} \tag{6}$$

and even F,

$$p_{a,i}^{(k+1)} = \frac{1}{F}\left(p_{a-F/2,i}^{(k)} + ... + p_{a,i}^{(k)} + ... + p_{a+(F/2-1),i}^{(k)}\right). \tag{7}$$

In any of these two cases, this is the so-called master equation of the random walk (Risken, 1996). In the limit of large image width L and large depth $\tilde{K}$, we can write the related equation for the continuous process $p_i(a, k)$, which is called Fokker-Planck equation in physics and chemistry (Risken, 1996) or forward Kolmogorov equation in mathematics (Saloff-Coste & Bremaud, 2000),

$$\partial_k p_{a,i}^{(k)} = v\partial_a p_{a,i}^{(k)} + D\partial_a^2 p_{a,i}^{(k)}. \tag{8}$$

where the drift coefficient v and the diffusion coefficient D are defined in terms of the probability distribution $W_i(x)$ of having a jump x starting from the location i

$$v = \int dx W_i(x) x, \qquad D = \int dx W_i(x) x^2. \tag{9}$$

In our case we have $W_i(x) = 1/F$ for $x \in [i - (F-1)/2, i + (F-1)/2]$ for odd F and $x \in [i - F/2, i + F/2 - 1]$ for even F, yielding the solutions for the Fokker-Planck equations for even and odd F reported in Equation 4 and Equation 5.

We can better characterize the limits of large image width L and large network depth $\tilde{K}$ as follows. The proof relies on the fact that a random walk, after a large number of steps, converges to a diffusion process. Here the number of steps is given by the depth $\tilde{K}$ of the network. Consequently, we need $\tilde{K} \gg 1$. Moreover, we want that the diffusion process is not influenced by the boundaries of the image, of width L. The average path walked by the random walker after $\tilde{K}$ steps is given by $F\sqrt{\tilde{K}}$. Then, we require $F\sqrt{\tilde{K}} \ll L$.

□

B EXPERIMENTAL SETUP

Table 1: Average over channels of filters in layer k, before and after training, for simple CNNs with $s = 1$ and $F = 3$ trained on task 1. The network learns filters which are much more homogeneous than initialization.

	Init.	After training
$k = 1$	$[0.0132, 0.0023, -0.0068]$	$[0.2928, 0.2605, 0.2928]$
$k = 2$	$[0.0014, -0.0007, -0.0009]$	$[0.0039, 0.0035, 0.0039]$
$k = 3$	$[-0.0006, -0.0001, 0.0010]$	$[0.0043, 0.0038, 0.0043]$
$k = 4$	$[3.4610e-05, 6.5687e-04, -9.1634e-04]$	$[0.0039, 0.0033, 0.0038]$
$k = 5$	$[-0.0006, 0.0002, -0.0009]$	$[0.0038, 0.0032, 0.0038]$
$k = 6$	$[0.0012, -0.0011, -0.0003]$	$[0.0038, 0.0031, 0.0038]$
$k = 7$	$[-0.0006, 0.0004, 0.0003]$	$[0.0041, 0.0032, 0.0040]$
$k = 8$	$[0.0005, -0.0012, 0.0010]$	$[0.0036, 0.0024, 0.0035]$
$k = 9$	$[0.0005, -0.0012, 0.0010]$	$[0.0021, 0.0016, 0.0017]$
$k = 10$	$[-0.0025, 0.0015, -0.0006]$	$[-0.0013, -0.0008, -0.0010]$
$k = 11$	$[-0.0006, 0.0005, 0.0009]$	$0.0002, 0.0002, 0.0002]$
$k = 12$	$[3.3418e-04, 3.3521e-05, 1.3936e-03]$	$[0.0009, 0.0008, 0.0009]$

All experiments are performed in PyTorch. The code with the instructions on how to reproduce experiments are found here: https://tinyurl.com/cnn-spatial-experiments.

B.1 DEEP NETWORKS TRAINING

In this section, we describe the experimental setup for the training of the deep networks deployed in Sections 1, 2 and 3.

For CIFAR10, fully connected networks are trained with the ADAM optimizer and learning rate $= 0.1$ while for CNNs SGD, learning rate $= 0.1$ and momentum $= 0.9$. In the latter case, the learning rate follows a cosine annealing scheduling. In all cases, the networks are trained on the cross-entropy loss, with a batch size of 128 and for 250 epochs. Early stopping at the best validation error is performed for selecting the networks to study. During training, we employ standard data augmentation consisting of random translations and horizontal flips of the input images. On the scale-detection task, we perform SGD on the hinge loss and halve the learning rate to 0.05. All results are averaged when training on 5 or more different networks initializations.

For ImageNet, we used pretrained models from Pytorch, `torchvision.models`.

B.2 SIMPLE CNNS TRAINING

In this section we present the experimental setup for the training of simple CNNs introduced in section 4, whose sensitivities to diffeomorphisms and Gaussian noise are shown in Figure 4.

We use CNNs with stride $s = 1$ and filter size $F = 3$. The width of the CNN is fixed to 1000 channels, while the depth to 12 layers. We use the Scale-Detection task introduced in section 3 with $\xi = 11$ and gap $g = 4$ and image size $L = 32$. For the training, we use $P = 48$ training points and Stochastic Gradient Descent (SGD) with learning rate 0.01 and batch size 8. We use weight decay for the L_2 norm of the filters weights with ridge 0.01. We stop the training after 500 times the interpolation time, which is the time required by the network to reach zero interpolation error of the training set. The goal of this procedure is to reach the solution with minimal norm. The generalization error of the trained CNNs is exactly zero: they learn spatial pooling perfectly. We show the sensitivities of the trained CNNs, averaged over 4 seeds, in Figure 4. We remark that to compute G_k we inserted Gaussian noise with already the ReLU applied on, since we observe that without it we would see a pre-asymptotic behaviour for G_k with respect to A_k.

To support the assumption done in section 4 that the trained CNNs are effectively behaving as one channel with homogeneous positive filters, we report the numerical values of the average filter over channels per layer in Table 1 for Task 1. They are positive in the first 9 hidden layers, where channel pooling is most pronounced.

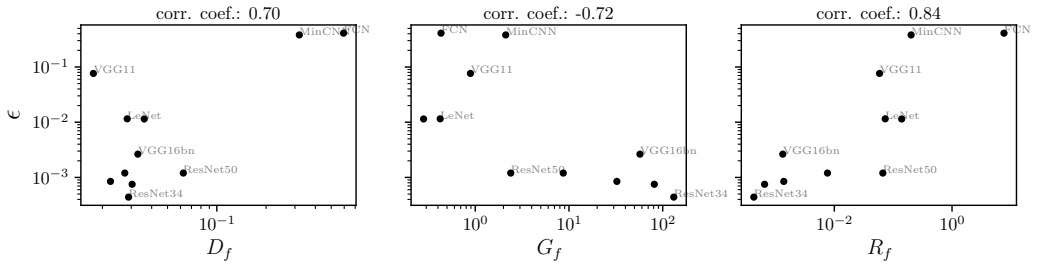

Figure 5: Generalization error ϵ versus sensitivity to diffeomorphisms D_f (left), noise G_f (center) and relative sensitivity R_f (right) for a wide range of architectures trained on scale-detection task 1 (train set size: 1024, image size: 32, $\xi = 14, g = 2$). As in real data, ϵ is positively correlated with D_f and negatively correlated with G_f. The correlation is the strongest for the relative measure R_f.

C ADDITIONAL FIGURES AND TABLES

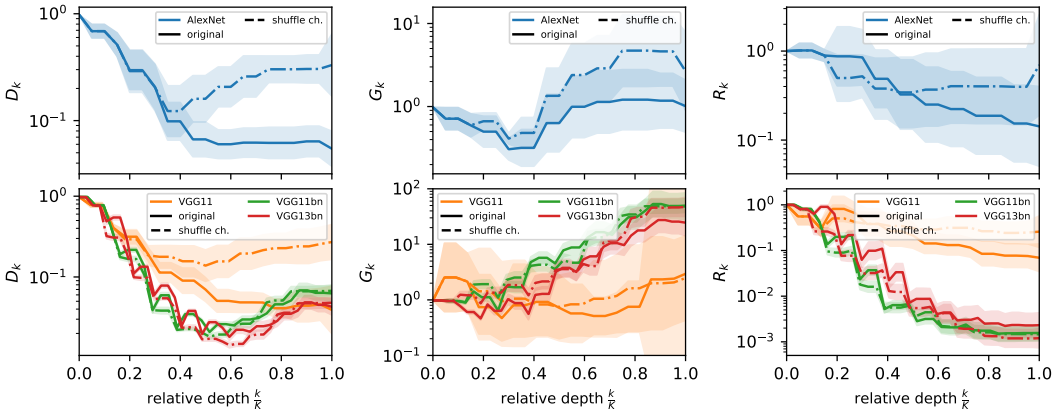

Figure 6: Sensitivities (D_k left, G_k middle and R_k right) of the internal representations vs relative depth for AlexNet (1st row) and VGG networks (2nd row) trained on scale-detection task. Dot-dashed lines show the sensitivities of networks with shuffled channels.

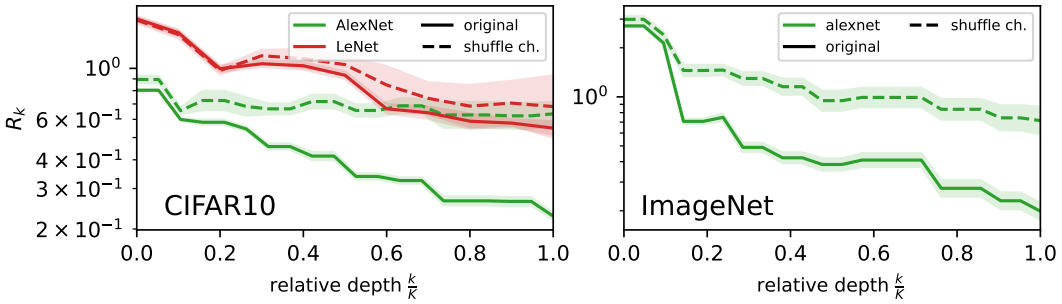

Figure 7: Analogous of Figure 2 for different network architectures: relative sensitivity R_k as a function of depth for LeNet and AlexNet architectures trained on CIFAR10 (left) and ImageNet (right). Full lines indicate experiments done on the original networks, dashed lines the ones after shuffling channels. K indicates the networks total depth.

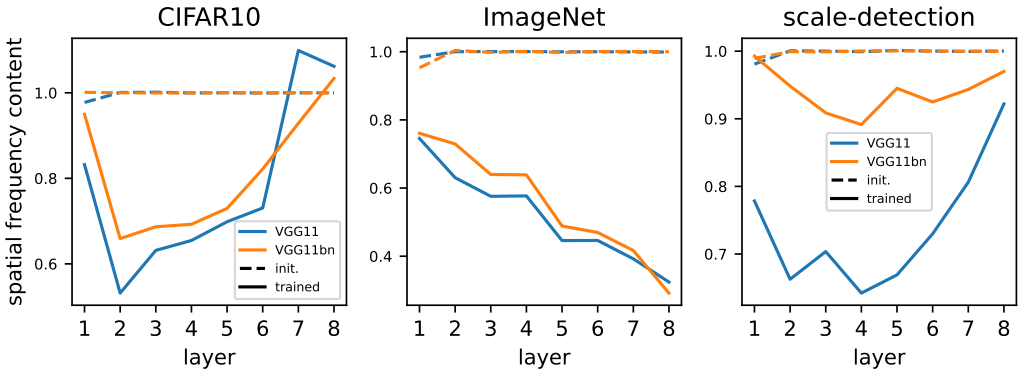

Figure 8: Spatial frequency content of filters for CIFAR10 (left), ImageNet (center) and the scale-detection task (right). The y-axis reports an aggregate measure among spatial frequencies: $N(\sum_{i=1}^{N}\lambda_l)^{-1}\langle\|\boldsymbol{w}_c^k\|^2\rangle_c^{-1}\sum_{l=1}^{F^2}\lambda_l\langle(\boldsymbol{\Psi}_l\cdot\boldsymbol{w}_c^k)^2\rangle_c$, where $\boldsymbol{\Psi}_l$ are the 3×3 Laplacian eigenvectors and λ_l the corresponding eigenvalues, $\boldsymbol{w}_c^k$ the c-th filter of layer k and $\langle\cdot\rangle_c$ denotes the average over c. This is an aggregate measure over frequencies, the frequencies distribution is reported in the main text, Figure 3.

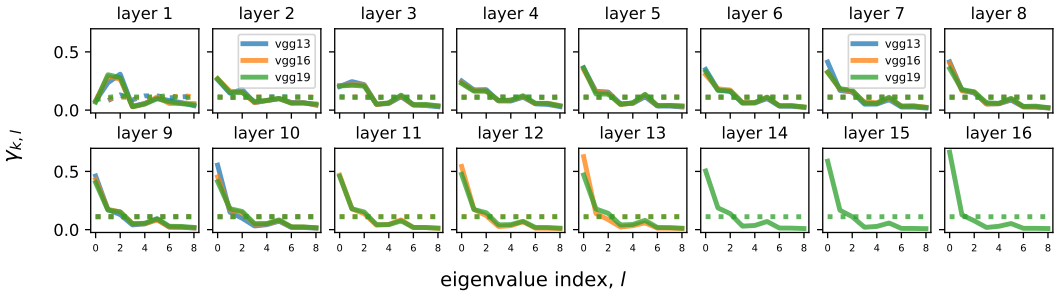

Figure 9: Analogous of Figure 3 for deep VGGs trained on ImageNet. Dotted and full lines respectively correspond to initialization and trained networks. The x-axis reports low to high frequencies from left to right. Deeper layers are reported in rightmost panels.

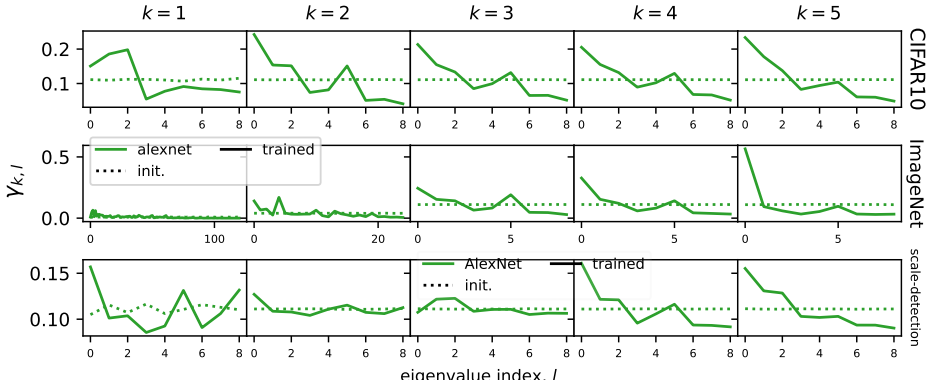

Figure 10: Analogous of Figure 3 for AlexNet trained on CIFAR10 (1st row), ImageNet (2nd row) and the scale detection task (3rd row). Dotted and full lines respectively correspond to initialization and trained networks. The x-axis reports low to high frequencies from left to right. Deeper layers are reported in rightmost panels.