Scattering Networks for Hybrid Representation Learning

Edouard Oyallon, Sergey Zagoruyko, Gabriel Huang, Nikos Komodakis,
Simon Lacoste-Julien, Matthew Blaschko, Eugene Belilovsky

Scattering Networks for Hybrid Representation Learning

Edouard Oyallon, Sergey Zagoruyko, Gabriel Huang, Nikos Komodakis, Simon Lacoste-Julien, Matthew Blaschko, Eugene Belilovsky

Abstract—Scattering networks are a class of *designed* Convolutional Neural Networks (CNNs) with fixed weights. We argue they can serve as generic representations for modelling images. In particular, by working in scattering space, we achieve competitive results both for supervised and unsupervised learning tasks, while making progress towards constructing more interpretable CNNs. For supervised learning, we demonstrate that the early layers of CNNs do not necessarily need to be learned, and can be replaced with a scattering network instead. Indeed, using hybrid architectures, we achieve the best results with predefined representations to-date, while being competitive with end-to-end learned CNNs. Specifically, even applying a shallow cascade of small-windowed scattering coefficients followed by 1×1-convolutions results in AlexNet accuracy on the ILSVRC2012 classification task. Moreover, by combining scattering networks with deep residual networks, we achieve a single-crop top-5 error of 11.4% on ILSVRC2012. Also, we show they can yield excellent performance in the small sample regime on CIFAR-10 and STL-10 datasets, exceeding their end-to-end counterparts, through their ability to incorporate geometrical priors. For unsupervised learning, scattering coefficients can be a competitive representation that permits image recovery. We use this fact to train hybrid GANs to generate images. Finally, we empirically analyze several properties related to stability and reconstruction of images from scattering coefficients.

Index Terms—Scattering transform, Wavelets, Deep neural networks, Invariance.

1 Introduction

Natural image processing tasks are high dimensional problems that require introducing lower dimensional representations: in the case of image classification, they must reduce the non-informative image variabilities, whereas for image generation, it is desirable to parametrize them. For example, some of the main source of variability are often due to geometrical operations such as translations and rotations. Then, an efficient classification pipeline necessarily builds invariants to these variabilities, whereas mapping to those sources of variabilities is desirable in the context of image generation. Deep architectures build representations that lead to state-of-the-art results on image classification tasks [24]. These architectures are designed as very deep cascades of non-linear end-to-end learned modules [33]. When trained on large-scale datasets they have been shown to produce representations that are transferable to other datasets [60], [26], which indicates they have captured generic properties of a supervised task that consequently do not need to be learned. Indeed several works indicate geometrical structures in the filters of the earlier layers of Deep CNNs [30], [56]. However, understanding the precise operations performed by those early layers is a complicated [54], [40] and possibly intractable task. In this work we investigate if it is possible to replace these early layers by simpler cascades of non-learned operators that reduce and parametrize variability while retaining all the discriminative information.

Indeed, there can be several advantages to incorporating predefined geometric priors via a hybrid approach of combining predefined and learned representations. First, end-to-end pipelines can be data hungry and ineffective when the number of samples is low. Secondly, it could lead to more interpretable classification pipelines, which are amenable to analysis, and permits the performance of parallel transport along the Euclidean group. Finally, it can reduce the spatial dimensions and the required depth of the learned modules, improving their computational and memory requirements.

A potential candidate for an image representation is the SIFT descriptor [34], which was widely used before 2012 as a feature extractor in classification pipelines [46], [47]. This representation was typically encoded via an unsupervised Fisher Vector (FV) and fed to a linear SVM. However, several works indicate that this is not a generic enough representation on top of which to build further modules [32], [6]. Indeed end-to-end learned features produce substantially better classification accuracy.

A Scattering Transform [36], [11], [49] is an alternative that solves some of the issues with SIFT and other predefined descriptors. In this work, we show that contrary to other proposed descriptors [55], a Scattering Network can avoid discarding information. Indeed, a Scattering Transform is not quantized, and the loss of information is avoided thanks to a combination of wavelets and non linear operators. Furthermore, it is shown in [42] that a Scattering Network provides a substantial improvement in classification accuracy over SIFT. A Scattering Transform also provides certain mathematical guarantees, which CNNs generally lack. Finally, wavelets are often observed in the initial layers, as in the case of AlexNet [30]. Thus, combing the two approaches is natural.

This article is an extended version of [41]. Our main contributions are as follows. First, we design and develop a fast algorithm to compute a Scattering Transform to use in a deep learning context. We demonstrate that using supervised local descriptors obtained by shallow 1×1 convolutions with very small spatial

Edouard Oyallon is with CentraleSupelec CVN and INRIA Galen team, France
Matthew Blaschko is affiliated with KU Leuven in Leuven, Belgium
Sergey Zagoruyko and Nikos Komodakis are at Ecole des Points in France
Simon Lacoste-Julien, Eugene Belilovsky, and Gabriel Huang are members of MILA and DIRO at University of Montreal, Montreal, Canada

window sizes obtains AlexNet accuracy on the ImageNet classification task (Subsection 4.1). We show empirically that these encoders build explicit invariance to local rotations (Subsection 4.3). Second, we propose hybrid networks that combine scattering with modern CNNs (Section 5) and show that using scattering and a ResNet of reduced depth, we obtain similar accuracy to ResNet-18 on ImageNet (Subsection 5.1). Then, we study adversarial examples to the Scattering Transform with a linear classifier. We then develop a procedure to reconstruct an image from its scattering representation in Section 3.4 and show that this can be used to incorporate the scattering transform in a hybrid Generativel Adversarial Network in Section 6.2. Finally, we demonstrate in Subsection 5.3 that scattering permits a substantial improvement in accuracy in the setting of limited data.

Our highly efficient GPU implementation of the scattering transform is, to our knowledge, orders of magnitude faster than any other implementations, and allows training very deep networks while applying scattering on the fly. Our scattering implementation[1] and pre-trained hybrid models[2] are publicly available.

2 Related Work

Closely related to our work, [44] proposed a hybrid representation for large scale image recognition combining a predefined representation and Neural Networks (NN), that uses a Fisher Vector (FV) encoding of SIFT and leverages NNs as scalable classifiers. In contrast we use the scattering transform in combination with convolutional architectures and show hybrid results that well exceed those of [44].

A large body of recent literature has also considered unsupervised and self-supervised learning for constructing discriminative image features [3], [18] that can be used in subsequent image recognition pipelines. However, to the best of our knowledge on complex datasets such as imagenet these representations do not yet approach the accuracy of supervised methods or handcrafted unsupervised representations. In particular the FV encoding discussed above is an unsupervised representation that has outperformed any unsupervised learned representation on the imagenet dataset [47].

With regards to the algorithmic implementation of the Scattering Transform, former implementations [11], [1] were only scaled for CPU as they retain too many intermediate variables, which can be too large for GPU use. A major contribution of our work is to propose an efficient approach which fits in GPU memory, which subsequently allows a much faster computational time than the CPU implementations. This is essential for scaling to the ImageNet dataset.

Concurrent to our work the Scattering Transform was also recently used in a context of generative modeling [2]: it is shown that by inverting the scattering transform, it is possible to generate images in a similar fashion as GANs. We however adopt a rather different approach by building hybrid GANs that directly learn to generate Scattering coefficients, which we reconstruct back into images.

3 Scattering: A Baseline for Image Classification

We now describe the scattering transform and motivate its use as a generic input for supervised tasks. A scattering network belongs to the class of CNNs whose filters are fixed wavelets [42]. The construction of this network has strong mathematical foundations [36], meaning it is well understood, relies on few parameters, and is stable to a large class of geometric transformations. In general, the parameters of a scattering transform do not need to be adapted to the bias of the dataset [42], making its output a suitable generic representation.

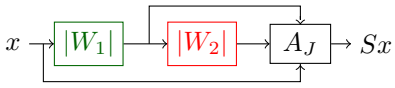


Fig. 1: A scattering network. A_J concatenates the averaged signals (cf. Section 3.1).

We then propose and motivate the use of supervised CNNs built on top of the scattering network. Finally we propose supervised encodings of scattering coefficients using 1x1 convolutions, which can retain interpretability and locality properties.

3.1 The Scattering Transform

In this section, we recall the definition of the scattering transform, introduced in [11], and clarify it by illustrating how to concretely apply it on a discrete image. In general, consider a signal $x(u)$, with u the spatial position index and an integer $J \in \mathbb{N}$, which is the spatial scale of our scattering transform. In particular, when x is a grayscale image, we write $x[p]$ its discretization, where $p_1, p_2 \leq N$. Let ϕ_J be a local averaging filter with a spatial window of scale 2^J (here, a Gaussian smoothing function). We obtain the *zeroth order* scattering coefficients $S^0x(u) = A_Jx(u) = x \star \phi_J(2^Ju)$ by applying[3] a local averaging operator A_J, followed by an appropriate downsampling of scale 2^J. The zeroth order scattering transform is approximately invariant to translations smaller than 2^J, but also results in a loss of high frequencies, which are necessary to discriminate signals. In our grayscale image example, S^0x is a feature map of resolution $\frac{N}{2^J} \times \frac{N}{2^J}$ with a single channel.

A solution to avoid the loss of high frequency information is to use wavelets. A wavelet is an integrable function with zero mean, which is localized both in Fourier and space domain [38]. A family of wavelets is obtained by dilating a complex mother wavelet ψ (here, a Morlet wavelet) such that $\psi_{j,\theta}(u) = \frac{1}{2^{2j}}\psi(r_{-\theta}\frac{u}{2^j})$, where $r_{-\theta}$ is the rotation by $-\theta$, and $j \geq 0$ is the scale of the wavelet. Thus, a given wavelet $\psi_{j,\theta}$ has its energy concentrated at a scale j in the angular sector θ. Let $L \in \mathbb{N}$ be an integer parametrizing a discretization of $[0, 2\pi]$. A wavelet transform is the convolution of a signal with the family of wavelets introduced above, followed by an appropriate downsampling:

$$W_1x(j_1, \theta_1, u) = \{x \star \psi_{j_1,\theta_1}(2^{j_1}u)\}_{j_1 \leq J, \theta_1 = 2\pi\frac{l}{L}, 1 \leq l \leq L}$$

Observe that j_1 and θ_1 have been discretized – the wavelet is chosen to be selective in angle and localized in the Fourier domain. With appropriate discretization [42], $\{A_Jx, W_1x\}$ is approximatively an isometry on the set of signals with limited bandwidth, which implies that the energy of the signal is preserved. This operator then belongs to the category of multi-resolution analysis operators, each filter being excited by a specific scale and angle, but with the output coefficients not being invariant to translation.

1. http://github.com/edouardoyallon/pyscatwave
2. http://github.com/edouardoyallon/scalingscattering

3. In this work, $\star$ denotes convolution, and has higher precedence than function evaluation.

To achieve invariance we cannot apply A_J directly to $W_1 x$ since it would result in a trivial invariant, namely zero.

To tackle this issue, we first apply a non-linear point-wise complex modulus to $W_1 x$, followed by an averaging A_J, and a downsampling of scale 2^J, which builds a non-trivial invariant. Here, the mother wavelet is analytic, thus $|W_1 x|$ is regular [5] which implies that the energy of $|W_1 x|$ in the Fourier domain is more likely to be contained in a lower frequency regime than $W_1 x$. Thus, A_J preserves more energy of $|W_1 x|$. It is possible to define

$$S^1 x = A_J |W_1| x,$$

which can also be written as:

$$S^1 x(j_1, \theta_1, u) = |x \star \psi_{j_1,\theta_1}| \star \phi_J(2^J u);$$

these are the *first-order* scattering coefficients. Following deep-learning terminology, each $S^1 x(j_1, \theta_1, \cdot)$ can be thought of as a one channel in a feature map. Again, the use of averaging builds an invariant to translation up to 2^J. In our grayscale image example, $S^1 x[p]$ is a feature map of resolution $\frac{N}{2^J} \times \frac{N}{2^J}$ with JL channels.

To recover some of the high-frequencies lost due to the averaging applied on the first order coefficients, we apply again a second wavelet transform W_2 (with the same filters as W_1) to each channel of the first-order scatterings, *before* the averaging step. This leads to the *second-order* scattering coefficients

$$S^2 x = A_J |W_2||W_1|,$$

which can also be written as

$$S^2 x(j_1, j_2, \theta_1, \theta_2, u) = ||x \star \psi_{j_1,\theta_1}| \star \psi_{j_2,\theta_2}| \star \phi_J(2^J u).$$

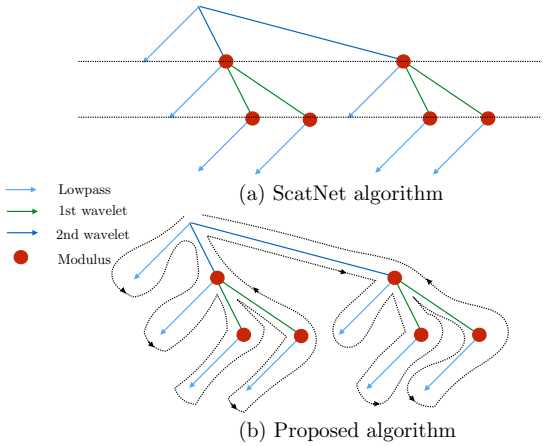
We only compute paths of increasing scale ($j_1 < j_2$) because non-increasing paths have been shown to bear no energy [11]. In our grayscale image example, $S^2 x[p]$ is a feature map of resolution $\frac{N}{2^J} \times \frac{N}{2^J}$ with $\frac{1}{2} J(J-1) L^2$ channels (one per increasing path).

We do not compute higher order scatterings, because their energy is negligible [11]. We call $Sx(u)$(or $S_J x(u)$) the final scattering coefficient corresponding to the concatenation of the order 0, 1 and 2 scattering coefficients, intentionally omitting the path index of each representation. A schematic diagram is shown in Figure 1. In the case of color images, we apply independently a scattering transform to each RGB channel of the image, which means $Sx(u)$ is a feature map with $3 \times \left(1 + JL + \frac{1}{2} J(J-1) L^2\right)$ channels, and the original image is down-sampled by a factor 2^J [11].

This representation has been proved to linearize small deformations of images [36], be non-expansive and almost complete [17], [10], which makes it an ideal input to a deep network algorithm, which can build invariants to this local variability via a first linear operator. We discuss its use as an initialization of a deep network in the next sections.

3.2 Efficient Implementation of Scattering Transforms

The implementation of a Scattering Network must be re-thought to benefit from GPU acceleration. Indeed, a GPU is a device which has a limited memory size in comparison with a CPU, and thus it is not possible to store intermediate computations. In this section, we show how to solve this problem of memory. We first describe the naive tree implementation [11], [42] and then our efficient GPU based implementation.



(a) ScatNet algorithm

(b) Proposed algorithm

Fig. 2: Trees of computations for a Scattering Transform. (a) corresponds to the traversal used in the ScatNet software package and (b) to our current implementation (PyScatWave).

3.2.1 Tree implementation of computations

We recall the algorithm to compute a Scattering Transform and its implementation in [11], [1] for order 2 Scattering with a scale of J and L different orientations for the wavelets. We explicitly show this algorithm is not appropriate to be scaled on a GPU. It corresponds to a level order traversal of the tree of computations of the Figure 2(a). Let us consider again a discretized input signal $x[p]$ of size N^2 which is a power of 2, and a spatial sampling of 1. For the sake of simplicity, we assume that an algorithm such as a symmetric padding has already been applied to x in order to avoid boundary effects that are inherent to periodic convolutions. The filter bank corresponds to $JL + 1$ filters:

$$\{\psi_{\theta,j}, \phi_J\}_{\theta, j \leq J}.$$

We only consider periodized filters, e.g.:

$$\tilde{\psi}_{\theta,j}(u) = \sum_{k_1,k_2} \psi_{\theta,j}(u + (Nk_1, Nk_2)).$$

A first wavelet transform must be applied on the input signal of the Scattering Transform. To this end, (a) a FFT of size N is applied. Then, (b) JL dot-wise multiplications with the resulting signal must be applied using the filters in the Fourier domain, $\{\hat{\tilde{\psi}}_{\theta,l}(\omega), \hat{\tilde{\phi}}_J(\omega)\}$. Each of the the resulting filtered $x \star \tilde{\psi}_{j,\theta}[p]$ or $x \star \tilde{\phi}_J[p]$ signals must be down-sampled by a factor of 2^j or 2^J, respectively, in order to reduce the computational complexity of the next operations. This is performed by (c) a periodization of the signal in the Fourier domain, which is equivalent to a down-sampling in the spatial domain, i.e. the resulting signal is $x \star \tilde{\psi}_{j,\theta}[2^j p]$ or $x \star \tilde{\phi}_J[2^J p]$. This last operation will lead to an aliasing, because there is a loss of information that can not be exactly recovered with Morlet filters. (a') An iFFT is then applied to each of the resulting filtered signals, which are of size $\frac{N^2}{2^j}, j \leq J$. (d) A modulus operator is applied to each of the signals, except to the low pass filter because it is a Gaussian. The set of filters to be reused at the next layer is $\{|x \star \tilde{\psi}_{j_1,\theta_1}[2^{j_1} p]|\}_{\theta_1 \leq L, j_1 < J, p_1 \leq \frac{N}{2^{j_1}}, p_2 \leq \frac{N}{2^{j_1}}}$ plus a low pass filter.

This requires the storage of $\mathcal{O}_{j_i,j_p} = L \sum_{j=j_i}^{J-1} \frac{N^2}{2^{2j_p}} + \frac{N^2}{2^{2J}}$ intermediate coefficients for the first layer, where $j_p = j_1, j_i = 0$.

Input size	J	ScatNetLight (in s)	PyScatWave (in s)
$32 \times 32 \times 3 \times 128$	2	2.5	0.03
$32 \times 32 \times 3 \times 128$	4	13	0.20
$128 \times 128 \times 3 \times 128$	2	16	0.26
$128 \times 128 \times 3 \times 128$	4	52	0.54
$256 \times 256 \times 3 \times 128$	2	160	0.71
$256 \times 256 \times 3 \times 128$	3		1.52
$256 \times 256 \times 3 \times 128$	4		1.73

TABLE 1: Comparison of the computation time of a Scattering Transform on CPU/GPU. PyScatWave (Algorithm 1) significantly improves performance in practice.

This step is iterated one more time, on each of the JL wavelet modulus signals, while only considering increasing paths. This means a wavelet transform and a modulus applied on a signal $|x \star \tilde{\psi}_{j_1,\theta_1}[2^j p]|$ lead to an additional storage requirement of $\mathcal{O}_{j_1,j_2}$. Consequently, the total number of coefficients stored for the second layer of the transform is $\sum_{j_1=0}^{J-1} L\mathcal{O}_{j_1,j_2}^2$ Finally, an averaging is applied on the second order wavelet modulus coefficients, which leads to a memory usage of $\frac{J(J-1)}{2} L^2 \frac{N^2}{2^{2J}}$ additional coefficients. Thus in total, the tree implementation requires a storage size of

$$\mathcal{O}_{0,j_1} + \sum_{j_1=0}^{J-1} L\mathcal{O}_{j_1,j_2}^2 + \frac{J(J-1)}{2} L^2 \frac{N^2}{2^{2J}}$$

The above approach is far too memory-consuming for a GPU implementation. For example, for $J = 2, 3, 4$, $L = 8$, and $N = 256$, which corresponds to the setting used on our ImageNet experiments, we numerically have approximately $2M, 2.5M, 2.6M$ parameters for a single tensor. A parameter is about 4 bytes, thus an image is about 8MB in the smallest case. In the case of batches of size 256 with color images, we thus need at least 6GB of memory simply to store the intermediate tensors used by the scattering, which does not take in account extra-memory used by libraries such as cuFFT for example. In particular, this reasoning demonstrates that a typical GPU with 12GB of memory can not efficiently process images in parallel with this naive approach.

3.2.2 Memory efficient implementation on GPUs

We now describe a GPU implementation which tries to minimize the memory usage during the computations. The procedures (a/a'), (b), (c) and (d) of the previous section can be efficiently implemented entirely on GPUs. They are fast, and can be implemented in batches, which permits parallel computations of the scattering representation. This is necessary for deep learning pipelines, which commonly use batches of data augmented samples.

To this end, we propose to perform an infix traversal of the tree of computations of the scattering. We introduce $\{\tilde{U}_j^1, \tilde{U}_j^2\}_{j \leq J}$, which are two sequences of temporary variables of length $\{\frac{N}{2^j}\}_{j \leq J}$ and a vector $\tilde{U}_0^0$ of length N. The total amount of memory that will be used is at most $5N^2$. Here, a color image of size $N = 256$ corresponds to at most approximately 0.98M coefficients. It divides the memory usage by at least 2 and permits us to scale processing to ImageNet. Algorithm 1 presents the algorithm we used in our implementation, dubbed PyScatWave. Table 1 demonstrates the speed-up for different values of tensors on a TitanX, compared with ScatNetLight [42].

We also note that in the case of training hybrid networks it is possible to store the computed scattering coefficients for a dataset via a cache. In this case, it is possible to obtain a speedup by a large factor since no extra computations are required to compute the earlier layers as optimization of the network proceeds. These early layers are often the most computationally expensive in comparison with deeper layers.

Algorithm 1: Pseudo-code of the algorithm used in PyScatWave.

function Scattering (x, J);
Input : Where x - image, J - scale
Output: $scattering(x, J)$
$\tilde{U}_0^0 = FFT(x)$;
$\tilde{U}_0^1 = \hat{\tilde{\phi}}_J \odot \tilde{U}_0^0$;
$S_J^0 x = iFFT(\text{periodize}(\tilde{U}_0^1, J))$;
for $\lambda_1 = (j_1, \theta_1)$ **do**
  $\tilde{U}_0^1 = \hat{\tilde{\psi}}_{\lambda_1} \odot \tilde{U}_0^0$;
  $\tilde{U}_{j_1}^1 = FFT(|iFFT(\text{periodize}(\tilde{U}_0^1, j_1)))$;
  $\tilde{U}_{j_1}^2 = \hat{\tilde{\phi}}_J \odot \tilde{U}_{j_1}^1$;
  $S_J^1 x[\lambda_1] = iFFT(\text{periodize}(\tilde{U}_{j_1}^2, J))$;
  for $\lambda_2 = (j_2, \theta_2)$, $j_1 < j_2$ **do**
    $\tilde{U}_{j_1}^2 = \hat{\tilde{\psi}}_{\lambda_2} \odot \tilde{U}_{j_1}^1$;
    $\tilde{U}_{j_2}^2 = FFT(|iFFT(\text{periodize}(\tilde{U}_{j_1}^2, j_2))|)$;
    $\tilde{U}_{j_2}^2 = \hat{\tilde{\phi}}_J \odot \tilde{U}_{j_2}^2$;
    $S_J^2 x[\lambda_1, \lambda_2] = iFFT(\text{periodize}(\tilde{U}_{j_2}^2, J))$;
  end
end

3.3 Reconstruction from the Scattering Coefficients

Reconstruction of an image from a scattering representation can be critical for permitting it's use in applications such as image generation. It also permits to obtain insights into the representation. We describe a simple method to reconstruct an image from its order 2 scattering representation. Several works [17], [10] proposed to synthesize textures and stochastic processes from their expected scattering coefficients. In the case of stationary processes, the final local averaging of a scattering transform allows the building of an unbiased estimator of the expected scattering coefficients, and the smallest variance is achieved using the largest windows size of invariance, i.e. the full image. This does not hold in the case of natural images, which do not correspond to stationary processes, and thus, global invariance to translation is not desirable because it loses spatial localization information. We show a straightforward approach can yield competitive reconstruction.

The method used [10], [9] consists in minimizing the ℓ_2 reconstruction error between an input image x and a candidate $\hat{x}$:

$$\hat{x} = \arg\inf_y \|S_J x - S_J y\|_2$$

This is achieved via a gradient descent, without however any (known) theoretical guarantees of convergence to the original signal. Computations are made possible thanks to the auto-differentiation tool of PyTorch. In this setting, we chose the optimizer Adam. The initial image is initialized as a white noise with variance 10^{-4} and is represented in the YUV space because it decorrelates approximatively the color channels and the intensity channels, and we observed it leads to better reconstruction. The

algorithm converges to a visually reasonable solution after 200 iterations, the loss reaching a plateau, and there is no extra-regularization or parametrization because empirically this has not yielded better reconstruction. Results are displayed in Figure 3 for different values of J and an image x of size 256^2. For each reconstruction, we evaluate its quality by computing the relative error of reconstruction with the original signal $\text{err}(x)$, and its distance in the scattering space $\text{err}(S_J)$,

$$\text{err}(x) = \frac{\|\hat{x} - x\|}{\|x\|} \quad \text{and} \quad \text{err}(S_J) = \frac{\|S_J\hat{x} - S_Jx\|}{\|S_Jx\|}.$$

We demonstrate good reconstruction in the case of $J = 2, 3, 4$ and we show that numerically, by $J \geq 5$, the obtained images are rather different from the original image due to the averaging loss. The attributes that are not well reconstructed are blurry and not at the appropriate spatial localization, which seems to indicate they have been lost by the spatial averaging. For $J < 7$, the Scattering coefficients are almost identical, however, for $J = 5, 6$ several corners and borders of the images are not well recovered, which indicates it is possible to find very different images with similar scattering coefficients. An open question is to understand if cascading more wavelet transforms could recover this information. For $J \geq 7$, the reconstructed signals are very different, only several textures seem to have been recovered and the color channels are decorrelated. Furthermore, the case $J = 7$ exhibits strong artifacts from the large scale wavelet, which is linked to the implementation of the wavelet transform.

Due to this lack of localization and ability to discriminate, in the following sections we combine CNNs with a scattering transform with scales $J < 5$, and therefore filters of width less than $2^5 = 32$ pixels.

3.4 Cascading a Supervised Architecture on Top of Scattering

We now motivate the use of a supervised architecture on top of a scattering network. Scattering transforms have yielded excellent numerical results [11] on datasets where the variabilities are completely known, such as MNIST or FERET. In these task, the problems encountered are linked to sample and geometric variance and handling these variances leads to solving these problems. However, in classification tasks on more complex image datasets, such variabilities are only partially known as there are also non geometrical intra-class variabilities. Although applying the scattering transform on datasets like CIFAR-10 or CalTech leads to nearly state-of-the-art results in comparison to other unsupervised representations, there is a large gap in performance when comparing to supervised representations [42]. CNNs fill in this gap. Thus we consider the use of deep neural networks utilizing generic scattering representations in order to learn more complex invariances than geometric ones alone.

Recent works [37], [12], [28] have suggested that deep networks could build an approximation of the group of symmetries of a classification task and apply transformations along the orbits of this group, like convolutions. This group of symmetry corresponds to some of the non-informative intra class variabilities, which must be reduced by a supervised classifier. [37] motivates that each layer corresponds to an approximated Lie group of symmetry, and this approximation is progressive in the sense that the dimension of these groups is increasing with depth. For instance, the main linear Lie group of symmetry of an image is the translation group, $\mathbb{R}^2$. In

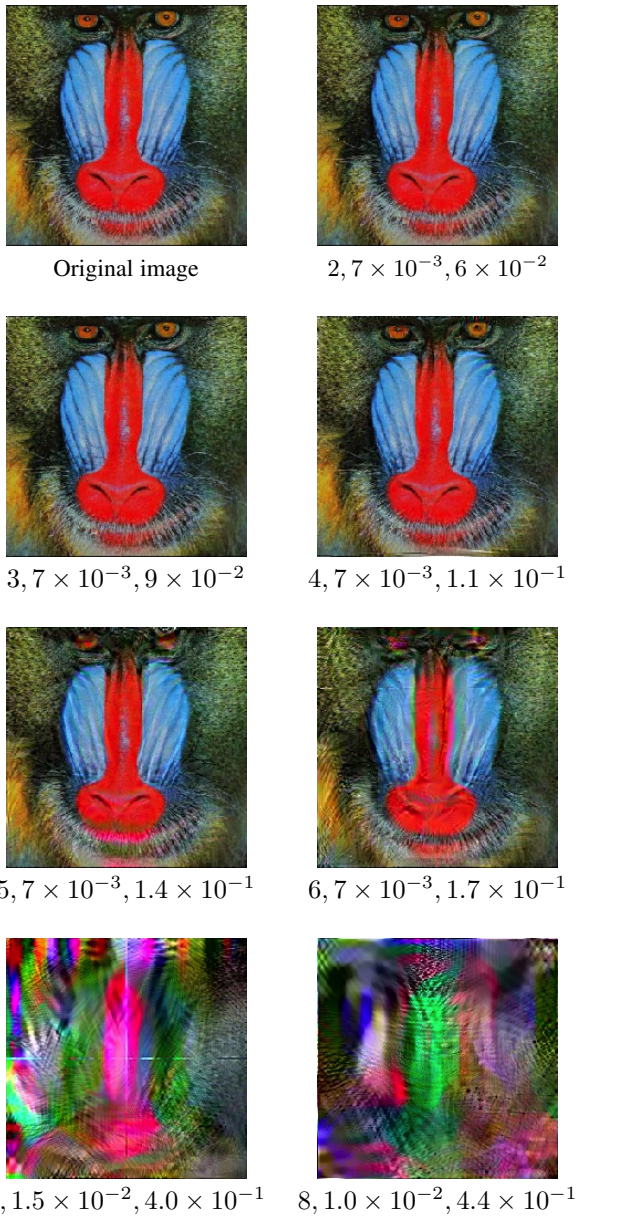


Fig. 3: Reconstructed images with subcaption indicating $J, \text{err}(S_J), \text{err}(x)$. See Section 3.4 for details of the reconstruction approach.

the case of a wavelet transform obtained by rotation of a mother wavelet, it is possible to recover a new subgroup of symmetry after a modulus non-linearity, the rotation SO_2, and the group of symmetry at this layer is the roto-translation group: $\mathbb{R}^2 \ltimes SO_2$. If no non-linearity was applied, a convolution along $\mathbb{R}^2 \ltimes SO_2$ would be equivalent to a spatial convolution. Discovering explicitly the next new and non-geometrical groups of symmetry is however a difficult task [28]; nonetheless, the roto-translation group seems to be a good initialization for the first layers. In this work, we investigate this hypothesis and avoid learning those well-known symmetries.

Thus, we consider two types of cascaded deep networks on top of scattering. The first, referred to as the *Shared Local Encoder* (SLE), learns a supervised local encoding of the scattering coefficients. We motivate and describe the SLE in the next sub-

section as an intermediate representation between unsupervised local pipelines, widely used in computer vision prior to 2012, and modern supervised deep feature learning approaches. The second, referred to as a hybrid CNN, is a cascade of a scattering network and a standard CNN architecture, such as a ResNet [24]. In the sequel we empirically analyse hybrid CNNs, which allow us to greatly reduce the spatial dimensions on which convolutions are learned and can reduce sample complexity.

4 Local Encoding of Scattering

First, we motivate the use of the Shared Local Encoder for natural image classifications. Then, we evaluate the supervised SLE on the Imagenet ILSVRC2012 dataset. This is a large and challenging natural color image dataset consisting of 1.2 million training images and $50,000$ validation images, divided into 1000 classes. We then show some unique properties of this network and evaluate its features on a separate task.

4.1 Shared Local Encoder for Scattering Representations

We now discuss the spatial support of different approaches, in order to motivate our local encoder for scattering. In CNNs constructed for large scale image recognition, the representations at a specific spatial location and depth depend upon large parts of the initial input image and thus mixes global information. For example, in [30], the effective spatial support of the corresponding filter is already 32 pixels (out of 224) at depth 2. The specific representations derived from CNNs trained on large scale image recognition are often used as representations in other computer vision tasks or datasets [57], [60].

On the other hand prior to 2012 local encoding methods led to state of the art performance on large scale visual recognition tasks [46]. In these approaches local neighborhoods of an image were encoded using method such as SIFT descriptors [34], HOG [15], and wavelet transforms [48]. They were also often combined with an unsupervised encoding, such as sparse coding [8] or Fisher Vectors (FVs) [46]. Indeed, many works in classical image processing or classification [29], [8], [46], [44] suggest that local encodings of an image are efficient descriptions. Additionally for some algorithms that rely on local neighbourhoods, the use of local descriptors is essential [34]. Observe that a representation based on local non overlapping spatial neighborhood is simpler to analyze, as there is no ad-hoc mixing of spatial information. Nevertheless, in large scale classification, this approach was surpassed by fully supervised learned methods [30].

We show that it is possible to apply a similarly local, yet supervised encoding algorithm to a scattering transform, as suggested in the conclusion of [44]. First observe that at each spatial position u, a scattering coefficient $S(u)$ corresponds to a descriptor of a local neighborhood of spatial size 2^J. As explained in the first Subsection 3.1, each of our scattering coefficients are obtained using a stride of 2^J, which means the final representation can be interpreted as a non-overlapping concatenation of descriptors. Let f be a cascade of fully connected layers that we identically apply on each $Sx(u)$. Then f is a cascade of CNN operators with spatial support size 1×1, thus we write $fSx \triangleq \{f(Sx(u))\}_u$. In the sequel, we do not make any distinction between the 1×1 CNN operators and the operator acting on $Sx(u), \forall u$. We refer to f as a *Shared Local Encoder*. We note that similarly to Sx, fSx corresponds to non-overlapping encoded descriptors. To learn a supervised classifier on a large scale image recognition task, we cascade fully connected layers on top of the SLE.

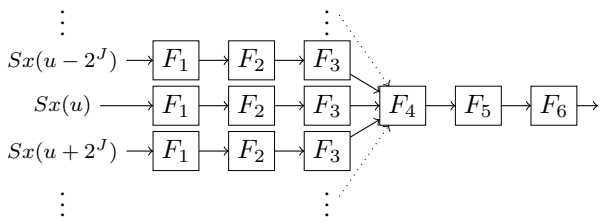


Fig. 4: Architecture of the SLE, which is a cascade of 3 1×1 convolutions followed by 3 fully connected layers. The ReLU non-linearities are included inside the F_i blocks.

Method	Top 1	Top 5
FV + FC [44]	55.6	78.4
FV + SVM [46]	54.3	74.3
AlexNet	56.9	**80.1**
Scat + SLE	**57.0**	79.6

TABLE 2: Top 1 and Top 5 percentage accuracy reported from one single crop on ILSVRC2012. We compare to other local encoding methods, and the Shared Local Encode (SLE) outperforms them (see Sec. 4.2 for experiment details). [44] single-crop result was provided by private communication.

Combined with a scattering network, the supervised SLE, has several advantages. Since the input corresponds to scattering coefficients whose channels are structured, the first layer of f is structured as well. We further explain and investigate this first layer in Subsection 4.3. Unlike standard CNNs, there is no linear combination of spatial neighborhoods of the different feature maps, thus the analysis of this network need only focus on the channel axis. Observe that if f was fed with raw images, for example in gray scale, it could not build any non-trivial operation except separating different level sets of these images.

In the next section, we investigate empirically this supervised SLE trained on the ILSVRC2012 dataset.

4.2 Shared Local Encoder on Imagenet

We first describe our training pipeline, which is similar to [59]. We trained our network for 90 epochs to minimize the standard cross entropy loss, using SGD with momentum 0.9 and a batch size of 256. We used a weight decay of 1×10^{-4}. The initial learning rate is 0.1, and is decreased by a factor of 10 at epochs 30, 50, 70, and 80. During the training process, each image is randomly rescaled, cropped, and flipped as in [24]. The final crop size is 224×224. At testing, we rescale the image to a size of 256×256, and extract a center crop of size 224×224.

We use an architecture which consists of a cascade of a scattering network, a SLE f, followed by fully connected layers. Figure 4 describes our architecture. We select the parameter $J = 4$ for our scattering network, which means the output representation has size $\frac{224}{2^4} \times \frac{224}{2^4} = 14 \times 14$ spatially and 1251 channels. f is implemented as 3 layers of 1x1 convolutions F_1, F_2, F_3 with layer size 1024. There are 2 fully connected layers of ouput size 1524. For all learned layers we use batch normalization [27] followed by a ReLU [30] non-linearity. We compute the mean and variance of the scattering coefficients on the whole of ImageNet, and standardized each spatial scattering coefficients with them.

Table 2 reports our numerical accuracies obtained with a single crop at testing, compared with local encoding methods, and AlexNet, which was the state-of-the-art approach in 2012. We obtain 20.4% at Top 5 and 43.0% Top 1 errors. The performance is analogous to the AlexNet [30]. In term of architecture, our hybrid model is analogous, and comparable to that of [46], [44], for which SIFT features are extracted followed by FV [47] encoding. Observe the FV is an unsupervised encoding compared to our supervised encoding. Two approaches are then used: the spatial localization is handled either by a Spatial Pyramid Pooling [31], which is then fed to a linear SVM, or the spatial variables are directly encoded in the FVs and classified with a stack of four fully connected layers. This last method is a major difference with ours, as the obtained descriptor does not have a spatial indexing anymore which are instead quantized. Furthermore, in both case, the SIFT are densely extracted which correspond to approximatively 2×10^4 descriptors, whereas in our case, only $14^2 = 196$ scattering coefficients are extracted. Indeed, we tackle the non-linear aliasing (due to the fact that the scattering transform is not oversampled) via random cropping during training, enabling invariance to small translations. In Top 1, [46] and [44] obtain error rates of 44.4% and 45.7%, respectively. Our method brings a substantial improvement of 1.4% and 2.7%, respectively.

The BVLC AlexNet[4] obtains a of 43.1% single-crop Top 1 error, which is nearly equivalent to the 43.0% of our SLE network. The AlexNet has 8 learned layers and as explained before, large receptive fields. On the contrary, our training pipeline consists in 6 learned layers with constant receptive field of size 16×16, except for the fully connected layers that build a representation mixing spatial information from different locations. This is a surprising result, as it seems to suggest contextual information is only necessary at the very last layers, to reach AlexNet accuracy.

We study briefly the local SLE, which only has a spatial extent of 16×16, as a generic local image descriptor. We use the Caltech-101 benchmark which is a dataset of 9144 images and 102 classes. We followed the standard protocol for evaluation [8] with 10 folds and evaluate per class accuracy with 30 training samples per class, using a linear SVM used with the SLE descriptors. Applying our raw scattering network leads to an accuracy of 62.8 ± 0.7, and the output features from F_1, F_2, and F_3 bring an absolute improvement of 13.7, 17.3, and 20.1, respectively. The accuracy of the final SLE descriptor is thus 82.9 ± 0.4, similar to that reported for the AlexNet final layer in [60] and sparse coding with SIFT [8]. However in both cases spatial variability is removed, either by Spatial Pyramid Pooling [31], or the cascade of large filters. By contrast, the concatenation of SLE descriptors are completely local. Similarly, the scattering network combined with ResNet-10 introduced in the next section, and followed by a linear SVM achieves 87.7 on Caltech-101, yet this descriptor is not local.

4.3 Interpreting SLE's first layer

Finding structure in the kernel of the layers of depth less than 2 [56], [60] is a complex task, and few empirical analyses exist that shed light on the structure [28] of deeper layers. A scattering transform with scale J can be interpreted as a CNN with depth J [42], whose channels indexes correspond to different scattering frequency indexes, which is a structuration. This structure is consequently inherited by the first layer F_1 of our SLE f. We analyse F_1 and show that it explicitly builds invariance to local rotations, and also that the Fourier bases associated to rotations are a natural bases of our operator. It is a promising direction to understand the nature of the next two layers.

4. https://github.com/BVLC/caffe/wiki/Models-accuracy-on-ImageNet-2012-val

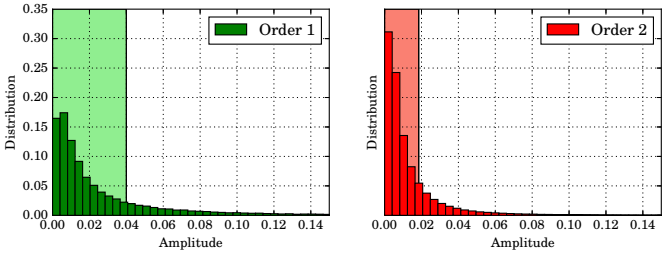


Fig. 5: Histogram of $\hat{F}_1$ amplitude for first and second order coefficients. The vertical lines indicate a threshold that is used in Subsection 4.3 to sparsify $\hat{F}_1$.

We first establish some mathematical notions linked to the rotation group that we use in our analysis. For the sake of clarity, we do not consider the roto-translation group. For a given input image x, let $r_\theta.x(u) \triangleq x(r_{-\theta}(u))$ be the image rotated by angle θ, which corresponds to the linear action of rotation on images. Observe the scattering representation is covariant with the rotation in the following sense:

$$\begin{aligned} S^1(r_\theta.x)(\theta_1, u) &= S^1x(\theta_1 - \theta, r_{-\theta}u) \triangleq r_\theta.(S^1x)(\theta_1, u), \\ S^2(r_\theta.x)(\theta_1, \theta_2, u) &= S^2x(\theta_1 - \theta, \theta_2 - \theta, r_{-\theta}u) \\ &\triangleq r_\theta.(S^2x)(\theta_1, \theta_2, u). \end{aligned}$$

Additionally, in the case of the second order coefficients, (θ_1, θ_2) is covariant with rotations, but $\theta_2 - \theta_1$ is an invariant to rotation that corresponds to a relative rotation.

The unitary representation framework [52] permits the building of a Fourier transform on a compact group, such as rotations. It is even possible to build a scattering transform on the roto-translation group [49]. Fourier analysis permits the measurement of the smoothness of the operator and, in the case of a CNN operator, it is a natural basis.

We can now numerically analyse the nature of the operations performed along angle variables by the first layer F_1 of f, with output size $K = 1024$. Let us define as $\{F_1^0 S^0 x, F_1^1 S^1 x, F_1^2 S^2 x\}$ the restrictions of F_1 to the order 0, 1, and 2 scattering coefficients respectively. Let $1 \leq k \leq K$ be an index of a feature channel and $1 \leq c \leq 3$ be the color index. In this case, $F_1^0 S^0 x$ is simply the weights associated to the smoothing $S_0 x$. $F_1^1 S^1 x$ depends only on (k, c, j_1, θ_1), and F_1^2 depends on $(k, c, j_1, j_2, \theta_1, \theta_2)$. We would like to characterize the smoothness of these operators with respect to the variables (θ_1, θ_2), because Sx is covariant to rotations.

To this end, we define by $\hat{F}_1^1$, $\hat{F}_1^2$ the Fourier transform of these operators along the variables θ_1 and (θ_1, θ_2) respectively. These operator are expressed in the tensorial frequency domain, which corresponds to a change of basis. In this experiment, we normalized each filter of F such that they have a ℓ_2 norm equal to 1, and each order of the scattering coefficients are normalized as well. Figure 5 shows the distribution of the amplitude of $\hat{F}_1^1, \hat{F}_2^2$. We observe that the distribution is shaped as a Laplace distribution, which is an indicator of sparsity.

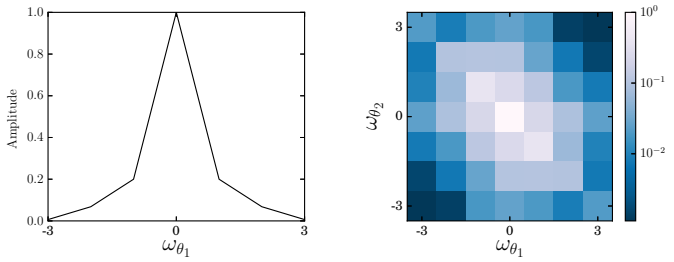


Fig. 6: Energy $\Omega_1\{F\}$ (left) and $\Omega_2\{F\}$ (right) from Eq. 1 for given angular frequencies.

To illustrate that this is a natural basis we explicitly sparsify this operator in its frequency basis and verify that empirically the network accuracy is minimally changed. We do this by thresholding by ϵ the coefficients of the operators in the Fourier domain. Specifically we replace the operators $\hat{F}_1^1$, $\hat{F}_1^2$ by $1_{|\hat{F}_1^1|>\epsilon}\hat{F}_1^1$ and $1_{|\hat{F}_1^2|>\epsilon}\hat{F}_1^2$. We select an ϵ that sets 80% of the coefficients to 0, which is illustrated in Figure 5. *Without retraining* our network performance degrades by only an absolute value of 2% worse on Top 1 and Top 5 ILSVRC2012. We have thus shown that this basis permits a sparse approximation of the first layer, F_1. We now show evidence that this operator builds an explicit invariant to local rotations.

To aid our analysis we introduce the following quantities:

$$\Omega_1\{F\}(\omega_1) \triangleq \sum_{k,j_1,c} |\hat{F}_1^1(k,c,j_1,\omega_{\theta_1})|^2, \quad (1)$$

$$\Omega_2\{F\}(\omega_{\theta_1},\omega_{\theta_2}) \triangleq \sum_{k,c,j_1,j_2} |\hat{F}_1^2(k,c,j_1,j_2,\omega_{\theta_1},\omega_{\theta_2})|^2.$$

They correspond to the energy propagated by F_1 for a given frequency, and quantify the smoothness of our first layer operator w.r.t. the angular variables. Figure 6 shows variation of $\Omega_1\{F\}$ and $\Omega_2\{F\}$ as a function of the frequencies. For example, if F_1^1 and F_1^2 were convolutional along θ_1 and (θ_1,θ_2), these quantities would correspond to their respective singular values. One sees that the energy is concentrated in the low frequency domain, which indicates that F_1 builds explicitly an invariant to local rotations.

5 Cascading a Supervised Deep CNN Architecture

We demonstrate that cascading modern CNN architectures on top of the scattering network can produce high performance classification systems. We apply hybrid convolutional networks on the Imagenet ILSVRC 2012 dataset as well as the CIFAR-10 dataset and show that they can achieve performance comparable to modern end-to-end learned approaches. We then evaluate the hybrid networks in the setting of limited data by utilizing a subset of CIFAR-10 as well as the STL-10 dataset and show that we can obtain substantial improvement in performance over analogous end-to-end learned CNNs.

5.1 Deep Hybrid CNNs on ILSVRC2012

We showed in the previous section that a SLE followed by FC layers can produce results comparable to AlexNet [30] on the ImageNet classification task. Here we consider cascading the scattering transform with a modern CNN architecture, such as ResNet [59], [24]. We take ResNet-18 [59] as a reference and construct a similar architecture with only 10 layers on top of the scattering network. We utilize a scattering transform with $J = 3$ such that the CNN is learned over a spatial dimension of 28×28 and a channel dimension of 651 (3 color channels of 217 each). ResNet-18 typically has 4 residual stages of 2 blocks each which gradually decrease the spatial resolution [59]. Since we utilize the scattering as a first stage we remove two blocks from our model. The network is described in Table 5.

We use the same optimization and data augmentation procedure described in Section 4.2 but with decreases in the learning rate at 30, 60, and 80 epochs. We find that when both methods are trained with the same settings of optimization and data augmentation, and when the number of parameters is similar (12.8M versus 11.7 M) the scattering network combined with a ResNet can achieve analogous performance (11.4% Top 5 for our model versus 11.1%), while utilizing fewer layers compared to a pure ResNet architecture. The accuracy is reported in Table 3 and compared to other modern CNNs.

Method	Top 1	Top 5	Params
AlexNet	56.9	80.1	61M
VGG-16 [23]	68.5	88.7	138M
Scat + Resnet-10 (ours)	68.7	88.6	12.8M
Resnet-18	68.9	88.8	11.7M
Resnet-200 [59]	**78.3**	**94.2**	64.7M

TABLE 3: ILSVRC-2012 validation accuracy (single crop) of hybrid scattering and 10 layer ResNet, a comparable 18 layer ResNet, and other well known benchmarks. We obtain comparable performance using a similar number of parameters while learning parameters at a spatial resolution of 28×28

Method	Accuracy
Unsupervised Representations	
CKN [35]	82.2
Roto-Scat + SVM [42]	82.3
ExemplarCNN [19]	84.3
DCGAN [45]	82.8
Scat + FC (ours)	**84.7**
Supervised and Hybrid	
Scat + WRN (ours)	93.1
Highway network [51]	92.4
All-CNN [50]	92.8
WRN 16 - 8 [59]	95.7
WRN 28 - 10 [59]	**96.0**

TABLE 4: Accuracy of scattering compared to similar architectures on CIFAR10. We set a new state-of-the-art in the unsupervised case and obtain competitive performance with hybrid CNNs in the supervised case.

Stage	Output size	Stage details
scattering	28×28	$J = 3, 651$ channels
conv1	28×28	[256]
conv2	28×28	$\begin{bmatrix} 256 \\ 256 \end{bmatrix}\times2$
conv3	14×14	$\begin{bmatrix} 512 \\ 512 \end{bmatrix}\times2$
avg-pool	1×1	$[14 \times 14]$

TABLE 5: Structure of Scattering and ResNet-10 architectures used in ImageNet experiments. Taking the convention of [59] we describe the convolution size and channels in the stage details.

Stage	Output size	Stage details
scattering	8×8, 24×24	$J = 2$
conv1	8×8, 24×24	16×k , 32×k
conv2	8×8, 24×24	$\begin{bmatrix} 32\times k \\ 32\times k \end{bmatrix} \times n$
conv3	8×8, 12×12	$\begin{bmatrix} 64\times k \\ 64\times k \end{bmatrix} \times n$
avg-pool	1×1	$[8 \times 8]$, $[12 \times 12]$

TABLE 6: Structure of Scattering and Wide ResNet hybrid architectures used in small sample experiments. Network width is determined by factor k. For sizes and stage details if settings vary, we list CIFAR-10 and then the STL-10 network information. All convolutions are of size 3×3 and the channel width is shown in brackets for both the network applied to STL-10 and CIFAR-10. For CIFAR-10 we use $n = 2$ and for the larger STL-10 we use $n = 4$.

This demonstrates both that the scattering networks does not lose discriminative power and that it can be used to replace early layers of standard CNNs. We also note that learned convolutions occur over a drastically reduced spatial resolution without resorting to pre-trained early layers, which can potentially lose discriminative information or become too task specific.

5.2 Deep Hybrid CNNs on CIFAR-10

We now consider the popular CIFAR-10 dataset consisting of color images composed of 5×10^4 images for training, and 1×10^4 images for testing divided into 10 classes. We use a hybrid CNN architecture with a ResNet built on top of the scattering transform.

For the scattering transform we used $J = 2$ which means the output of the scattering stage will be 8×8 spatially and 243 in the channel dimension. We follow the training procedure prescribed in [59] utilizing SGD with momentum of 0.9, batch size of 128, weigh decay of 5×10^{-4}, and modest data augmentation by using random cropping and flipping. The initial learning rate is 0.1, and we reduce it by a factor of 5 at epochs 60, 120 and 160. The models are trained for 200 epochs in total. We used the same optimization and data augmentation pipeline for training and evaluation in both case. We utilize batch normalization techniques at all layers which lead to a better conditioning of the optimization [27]. Table 4 reports the accuracy in the unsupervised and supervised settings and compares them to other approaches.

We compare to state-of-the-art approaches on CIFAR-10, all based on end-to-end learned CNNs. We use a similar hybrid architecture to the successful wide residual network (WRN) [59]. Specifically we modify the WRN of 16 layers, which consists of 4 convolutional stages. With k denoting the widening factor, after the scattering output we use a first stage of $32 \times k$. We add intermediate 1×1 convolutions to increase the effective depth, without substantially increasing the number of parameters. Finally we apply a dropout of 0.2 as specified in [59]. Using a width of 32 we achieve an accuracy of 93.1%. This is superior to several benchmarks but performs worse than the original ResNet [24] and the wide ResNet [59]. We note that training procedures for learning directly from images, including data augmentation and optimization settings, have been heavily optimized for networks trained directly on natural images, while we use them largely out of the box.

Method	**100**	**500**	**1000**	**Full**
WRN 16-8	34.7 ± 0.8	46.5 ± 1.4	60.0 ± 1.8	**95.7**
VGG 16 [58]	25.5 ± 2.7	46.2 ± 2.6	56 ± 1.0	92.6
Scat + WRN	**38.9 ± 1.2**	**54.7±0.6**	**62.0±1.1**	93.1

TABLE 7: Mean accuracy of a hybrid scattering in a limited sample situation on CIFAR-10 dataset. We find that including a scattering network is significantly better in the smaller sample regime of 500 and 100 samples.

5.3 Limited samples setting

A major application of a hybrid representation is in the setting of limited data. Here the learning algorithm is limited in the variations it can observe or learn from the data, such that introducing a geometric prior can substantially improve performance. We evaluate our algorithm on the limited sample setting using a subset of CIFAR-10 and the STL-10 dataset.

5.3.1 CIFAR-10

We take subsets of decreasing size of the CIFAR dataset and train both baseline CNNs and counterparts that utilize the scattering as a first stage. We perform experiments using subsets of 1000, 500, and 100 samples, which are split uniformly amongst the 10 classes.

We use as a baseline the Wide ResNet [59] of depth 16 and width 8, which shows near state-of-the-art performance on the full CIFAR-10 task in the supervised setting. This network consists of 4 stages of progressively decreasing spatial resolution detailed in [59, Table 1]. We construct a comparable hybrid architecture that removes a single stage and all strides, as the scattering already down-sampled the spatial resolution. This architecture is described in Table 6. Unlike the baseline, referred from here-on as WRN 16-8, our architecture has 12 layers and equivalent width, while keeping the spatial resolution constant through all stages prior to the final average pooling. We also incorporate the numerical results obtained via a VGG of depth 16 [58] for the sake of comparison.

We use the same training settings for our baseline, WRN 16-8, and our hybrid scattering and WRN-12. The settings are the same as those described for CIFAR-10 in the previous section, with the only difference being that we apply a multiplier to the learning rate schedule and to the maximum number of epochs. The multiplier is set to 10, 20, and 100 for the 1000, 500, and 100 sample cases, respectively. For example the default schedule of 60, 120, and 160 epochs becomes 600, 1200, and 1600 for the case of 1000 samples and a multiplier of 10. Finally in the case of 100 samples we use a batch size of 32 in lieu of 128.

Table 7 corresponds to the averaged accuracy over 5 different subsets, with the corresponding standard error. In this small sample setting, a hybrid network outperforms the purely CNN based baselines, particularly when the sample size is smaller. This is not surprising as we incorporate a geometric prior in the representation.

5.3.2 STL-10

The STL-10 dataset consists of color images of size 96×96, with only 5000 labeled images in the training set divided equally in 10 classes and 8000 images in the test set. The larger size of the images and the small number of available samples make this a challenging image classification task. The dataset also provides

Method	Accuracy
Supervised methods	
Scat + WRN 20-8	**76.0 ± 0.6**
CNN[53]	70.1 ± 0.6
Unsupervised methods	
Exemplar CNN [19]	75.4 ± 0.3
Stacked what-where AE [61]	74.33
Hierarchical Matching Pursuit (HMP) [7]	64.5±1
Convolutional K-means Network [13]	60.1±1

TABLE 8: Mean accuracy of a hybrid CNN on the STL-10 dataset. We find that our model is better in all cases even compared to those utilizing the large unsupervised part of the dataset.

100,000 unlabeled images for unsupervised learning. We do not utilize these images in our experiments, yet we find we are able to outperform all methods which learn unsupervised representations using these unlabeled images, obtaining very competitive results on the STL-10 dataset.

We apply a hybrid convolutional architecture, similar to the one applied in the small sample CIFAR task, adapted to the size of 96×96. The architecture is described in Table 6 and is similar to that used in the CIFAR small sample task. We use the same data augmentation as with the CIFAR datasets. We apply SGD with learning rate 0.1 and learning rate decay of 0.2 applied at epochs 1500, 2000, 3000, 4000. Training is run for 5000 epochs. We use at training and evaluation the predefined 10 folds of 1000 training images each, as given in [61]. The averaged result is reported in Table 8. Unlike other approaches, we do not use the 4000 remaining training images to perform hyper-parameter tuning on each fold, as this is not representative of small sample situations. Instead we train the same settings on each fold. The best reported result in the purely supervised case is a CNN [53], [19] whose hyper parameters have been automatically tuned using 4000 images for validation achieving 70.1% accuracy. The other competitive methods on this dataset utilize the unlabeled data to learn in an unsupervised manner before applying supervised methods. We also evaluate on the full training set of 5000 images obtaining an accuracy of 87.6%, which is quite higher than 81.3% [25] using unsupervised learning and the full training set. These techniques add several hyper parameters and require an additional engineering process. Applying a hybrid network is on the other hand straightforward and is very competitive with all the existing approaches without using any unsupervised learning. In addition to showing that hybrid networks perform well in the small sample regime, these results, along with our unsupervised CIFAR-10 result, suggest that completely unsupervised feature learning on image data may still not outperform supervised methods and predefined representations for downstream discriminative tasks. One possible explanation is that in the case of natural images, unsupervised learning of more complex variabilities than geometric ones (e.g the rototranslation group) might be ill-posed.

6 Unsupervised and Hybrid Unsupervised Learning with the Scattering Transform

This section describes the use of the Scattering Transform as an unsupervised representation and as part of hybrid unsupervised learning. First we evaluate the scattering as an unsupervised representation using the CIFAR-10 and ImageNet datasets, then we show that it can be used inside common unsupervised learning schemes by proposing a hybrid GAN combined with a Scattering Transform, which synthesizes Scattering Coefficients from random Gaussian noise on 32×32 color images from ImageNet. Using the reconstruction proposed in Section 3.4 we show that we can generate images from this GAN model.

6.1 Scattering as an Unsupervised Representation

We first consider the CIFAR-10 dataset used in Section 5.2 and perform an experiment that allows us to evaluate the scattering transform as an unsupervised representation with a complex non-convolutional classifier. In a second experiment, we consider the linear classification task on ILSVRC 2012 often used to evaluate unsupervised representations [3].

For CIFAR-10, as in Section 5.2, we used $J = 2$ which means the output of the scattering stage will be 8×8 spatially and 243 in the channel dimension. This task has been commonly evaluated on CIFAR-10 with a non-linear classifier [42] and we thus consider the use of a MLP. We follow the training procedure prescribed in [59] utilizing SGD with momentum of 0.9, batch size of 128, weigh decay of 5×10^{-4}, and modest data augmentation of the dataset by using random cropping and flipping. The initial learning rate is 0.1, and we reduce it by a factor of 5 at epochs 60, 120 and 160. The models are trained for 200 epochs in total. We used the same optimization and data augmentation pipeline for training and evaluation in both cases. We utilize batch normalization at all layers which leads to a better conditioning of the optimization [27]. Table 4 reports the accuracy in the unsupervised and supervised settings and compares them to other approaches. Combining the scattering transform with a NN classifier consisting of 3 hidden layers, with width 1.1×10^4, we show that one can obtain a new state of the art classification for the case of unsupervised convolutional layers. More numerical comparisons with other unsupervised methods, such as random networks, can be found in [42]. Scattering based approaches outperform all methods utilizing learned and not-learned unsupervised features, further demonstrating the discriminative power of the scattering network representation.

For the ILSVRC-2012 dataset we use a common evaluation based on training a linear classifier on top of the unsupervised representation [3]. We used a standard training protocol with cross-entropy loss on top of a scattering transform produced with $J = 4$. We apply standard data augmentation, optimizing with stochastic gradient descent with momentum 0.9, weight decay set to $1e-7$, and learning rate drops at epochs 20, 40, and 60. The results are shown in Table 9 and are compared with unsupervised and self-supervised baselines. Observe that a Scattering Transform improves significantly from a random baseline [3], and that it recognize a large number of images even when only considering the top result. The accuracy of a random baseline is still high, because the small support of the convolutional operators already incorporates some geometric structures in this type of pipeline. Modern learned unsupervised representations however can improve on this result.

In order to test the robustness of the Scattering Network w.r.t. adversarial examples, we used the simple sign gradient attack [20]. We build adversarial examples that fool our linear layer, which means for a given x classified as c that we desire to force the classifer to erroneously classify as $\tilde{c} \neq c$, we find the smallest ϵ_x such that:

$$\text{class}(x + \epsilon_x) = \tilde{c}.$$

Method	Top 1
Scattering + 1 FC	17.4 %
Random CNN [3]	12.9 %
Pathak et al [43]	22.3%
Doersch et al [16]	31.7%
Donahue et al [18]	31.0%
Noroozi and Favaro et al [39]	34.7%
Arandjelovic et al [3]	32.6 %

TABLE 9: Comparison of the top-1 accuracy from unsupervised and self-supervised representation, on the ImageNet dataset, evaluated as ours with a linear classifier. We compare to a reported result of a similar architecture and random initialization. We also show result of learned unsupervised representations for reference. Baselines for the linear classification results are taken from [3].

(a) Original image x, well classified with output probability: `0.35, tiger cat` (b) Adversarial sample $\tilde{x}$, wrongly classified with output probability `0.02, magnetic compass` (c) $x-\tilde{x}$ (magnified for a better visualization)

Fig. 7: Adversarial examples obtained from a Scattering Transform followed by a linear classifier on ImageNet.

In our case, candidates for ϵ are given by vectors collinear to the gradient sign in the direction of $\tilde{c}$ as explained in [21]. Results are shown in Figure 7.

It shows that being only 1-Lipschitz is not sufficient to be visually robust to such artifacts, when combined only with a linear classifier; using non-linear classifier, such as a CNN, designed to be robust to predefined noises could permit to tackle this issue.

6.2 Hybrid Unsupervised Learning with Scattering GAN

In this section we propose to construct a Generative Adversarial Network (GAN) in the space of scattering coefficients. This essentially constructs a hybrid generator and discriminator. The GAN is a state-of-the-art generative modeling framework. The use of the learned generator on top of a scattering transform can be well motivated if we consider the scattering transform as good a model of low level texture [11]. Furthermore, as extensive data augmentation is often not required, it is possible to store scattering representations that have a smaller spatial resolution, permitting us to try rapidly a variety of architectures. We demonstrate in the following that a scattering representation can be used as the initialization of a generative model, similar to the classification case.

We follow the Deep Convolutional Generative Adversarial Network architectures proposed in [45] in order to generate signals in the scattering space. We consider color images from the resized ImageNet dataset of size 32×32 in the YUV space that are processed by a Scattering Transform with $J = 2$. The scattering coefficients were renormalized to lie between -1 and 1. Their

Generator	
random uniform	Input size 100
2x2 Trans. Conv.	stride 1, batch norm, LeakyReLU, 256 out
4x4 Trans. Conv.	stride 2, pad 1,batchnorm, LeakyReLU,128 out
4x4 Trans. Conv.	stride 2, pad 1,batchnorm, tanh, 243x8x8 out
Discriminator	
random uniform	Input size 243x8x8
4x4 Conv.	stride 1, batchnorm, LeakyReLU, 128 out
4x4 Conv.	stride 2, pad 1,batchnorm, LeakyReLU,256 out
4x4 Conv.	stride 2, pad 1, LeakyReLU, 256 out
1x1 Conv.	stride 1,batchnorm, 256 out
Fully connected layer	

TABLE 10: Architecture of the Discriminator and Generator of the Scattering-DCGAN.

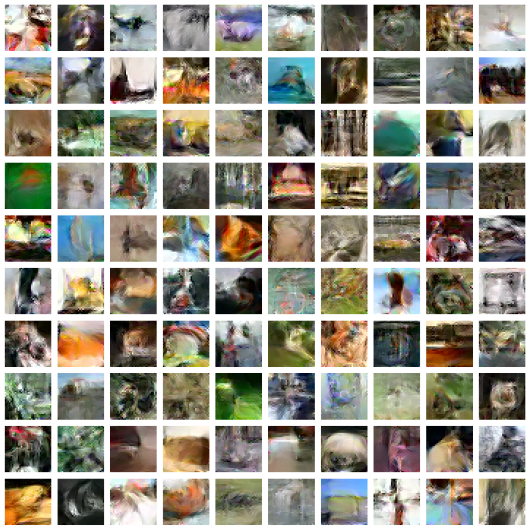


Fig. 8: Samples generated by the Scattering-DCGAN. See Section 6.2 for details.

scattering representations are then fed to the generator and discriminators of our Scattering-DCGAN. In particular, the generator aims to synthesize scattering coefficients from a Gaussian noise with $d = 100$. They are represented in Table 10. Moreover we apply the recently proposed Wasserstein distance based objective [22], [4].

We now describe our training procedure. We run the Adam optimizer for both the discriminator and generator during 600k iterations without observing significant instabilities during the optimization. The discriminator is trained during 5 successive iterations and the generator only 1, as done in [22], because we observed it leads to more realistic images. The generator takes as input a latent variable of 100 dimensions.

Section 3.4 shows that the scattering transform can be used to reconstruct images. We thus recover images generated from our model from the generated scattering coefficients, and they are shown in Figure 8. These images are qualitatively similar to other baselines, and it shows how one can use the scattering transform with more complex models. Generating coherent Scattering coefficients that leads to real images is challenging: the non-surjectivity of the scattering transform is due to physical constraints(e.g. interactions between different coefficients), yet we however did not incorporate this knowledge in our architectures.

Scattering approximator	
3x3 Convolution	stride 1, batch norm, ReLU, 128 output
3x3 Convolution	stride 2, batch norm, ReLU, 128 output
3x3 Convolution	stride 1, batch norm, ReLU, 128 output
3x3 Convolution	stride 1, batch norm, ReLU, 256 output
3x3 Convolution	stride 2, batch norm, ReLU, 243 output
Cascaded CNN	
3x3 Convolution $\times 10$	stride 1, batch norm, ReLU, 128 output
3x3 Convolution $\times 10$	stride 1, batch norm, ReLU, 256 output
Averaging layer	
Fully connected layer	

TABLE 11: Architecture of the Scattering approximator.

7 Learning Scattering

Many theoretical arguments of deep learning rely on the universal approximation theorem [14]. The flexibility of this deep learning frameworks raises the following question: can we approximate the first scattering layers by a deep network?

In order to explore this question, we consider a 5-layer convnet as a candidate to replace our scattering network on CIFAR10. Its architecture is described in Table 11, and it has the same output size as a scattering network. It has two downsampling steps, in order to mimic the behavior of a scattering network with $J = 2$. We build a hybrid architecture, i.e. scattering followed by a Cascaded CNN, described in Table 11 that leads to 91.4% on CIFAR10. Then we replace the scattering part by the CNN of Table 11, i.e. the Scattering Approximator. We train it, keeping the weights of the Cascaded CNN layers constant and equal to the optimal solution found with the scattering. Instead of minimizing a loss between the output of a scattering network and this network, we target the best input for the fixed convnet given the classification task.

This architecture can achieve 1% accuracy below the original pipeline, which indicates it is possible to learn the Scattering representation. Using a shallower network seems to degrade the performances, but we did not investigate this question further. In any case, the learned network will not have any guarantee of stability properties present in the original scattering transform.

8 Conclusion

This work demonstrates a competitive approach for large scale visual tasks, based on scattering networks, in particular for ILSVRC2012. When compared with unsupervised representations on CIFAR-10 or small data regimes on CIFAR-10 and STL-10, we demonstrate state-of-the-art results. We build a supervised Shared Local Encoder (SLE) that permits the scattering networks to surpass other local encoding methods on ILSVRC2012. This network of just 3 learned layers permits a deteailed analysis of the performed operations. We additionally prove that it is possible to synthetize images from a GAN in the Scattering space.

Our work also suggests that pre-defined features are still of interest and can provide valuable insights into deep learning techniques and to allow them to be more interpretable. Combined with appropriate learning methods, they enable stronger theoretical guarantees, which are necessary to engineer better deep models and stable representations.

Acknowledgment

The authors would like to thank Mathieu Andreux, Tomás Angles, Joan Bruna, Carmine Cella, Bogdan Cirstea, Michael Eickenberg, Stéphane Mallat, Louis Thiry for helpful discussions and support. The authors would also like to thank Rafael Marini and Nikos Paragios for use of computing resources. We would like to thank Florent Perronnin for providing important details of their work. This work is funded by the ERC grant InvariantClass 320959, via a grant for PhD Students of the Conseil régional d'Ile-de-France (RDM-IdF), Internal Funds KU Leuven, FP7-MC-CIG 334380, DIGITEO 2013-0788D - SOPRANO, NSERC Discovery Grant RGPIN-2017-06936, an Amazon Research Award to Matthew Blaschko, and by the Research Foundation - Flanders (FWO) through project number G0A2716N. We thank also the CVN (CentraleSupelec) for providing financial support.

References

[1] J. Andén, L. Sifre, S. Mallat, M. Kapoko, V. Lostanlen, and E. Oyallon. Scatnet. *Computer Software. Available: http://www. di. ens. fr/data/software/scatnet/.[Accessed: December 10, 2013]*, 2, 2014.

[2] T. Angles and S. Mallat. Generative networks as inverse problems with scattering transforms. *International Conference on Learning Representations*, 2018.

[3] R. Arandjelović and A. Zisserman. Look, listen and learn. In *IEEE International Conference on Computer Vision*, 2017.

[4] M. Arjovsky, S. Chintala, and L. Bottou. Wasserstein generative adversarial networks. In *Proceedings of the 34th International Conference on Machine Learning*, 2017.

[5] S. Bernstein, J.-L. Bouchot, M. Reinhardt, and B. Heise. Generalized analytic signals in image processing: comparison, theory and applications. In *Quaternion and Clifford Fourier Transforms and Wavelets*, pages 221–246. Springer, 2013.

[6] L. Bo, X. Ren, and D. Fox. Multipath sparse coding using hierarchical matching pursuit. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 660–667, 2013.

[7] L. Bo, X. Ren, and D. Fox. Unsupervised feature learning for RGB-D based object recognition. In *Experimental Robotics*, pages 387–402. Springer, 2013.

[8] Y.-L. Boureau, N. Le Roux, F. Bach, J. Ponce, and Y. LeCun. Ask the locals: multi-way local pooling for image recognition. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 2651–2658. IEEE, 2011.

[9] J. Bruna. *Scattering representations for recognition*. PhD thesis, Ecole Polytechnique X, 2013.

[10] J. Bruna and S. Mallat. Audio texture synthesis with scattering moments. *arXiv preprint arXiv:1311.0407*, 2013.

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

[12] J. Bruna, A. Szlam, and Y. LeCun. Learning stable group invariant representations with convolutional networks. *arXiv preprint arXiv:1301.3537*, 2013.

[13] A. Coates and A. Y. Ng. Selecting receptive fields in deep networks. In *Advances in Neural Information Processing Systems*, pages 2528–2536, 2011.

[14] G. Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4):303–314, 1989.

[15] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 886–893. IEEE, 2005.

[16] C. Doersch, A. Gupta, and A. A. Efros. Unsupervised visual representation learning by context prediction. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1422–1430, 2015.

[17] I. Dokmanić, J. Bruna, S. Mallat, and M. de Hoop. Inverse problems with invariant multiscale statistics. *arXiv preprint arXiv:1609.05502*, 2016.

[18] J. Donahue, P. Krähenbühl, and T. Darrell. Adversarial feature learning. *arXiv preprint arXiv:1605.09782*, 2016.

[19] A. Dosovitskiy, J. T. Springenberg, M. Riedmiller, and T. Brox. Discriminative unsupervised feature learning with convolutional neural networks. In *Advances in Neural Information Processing Systems*, pages 766–774, 2014.

[20] I. Goodfellow, J. Shlens, and C. Szegedy. Explaining and harnessing adversarial examples. In *International Conference on Learning Representations*, 2015.

[21] I. J. Goodfellow, J. Shlens, and C. Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

[22] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. Courville. Improved training of Wasserstein GANs. *arXiv preprint arXiv:1704.00028*, 2017.
[23] S. Han, J. Pool, J. Tran, and W. Dally. Learning both weights and connections for efficient neural network. In *Advances in Neural Information Processing Systems*, pages 1135–1143, 2015.
[24] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2016.
[25] E. Hoffer, I. Hubara, and N. Ailon. Deep unsupervised learning through spatial contrasting. *arXiv preprint arXiv:1610.00243*, 2016.
[26] M. Huh, P. Agrawal, and A. A. Efros. What makes ImageNet good for transfer learning? *arXiv preprint arXiv:1608.08614*, 2016.
[27] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32nd International Conference on International Conference on Machine Learning*, pages 448–456, 2015.
[28] J.-H. Jacobsen, E. Oyallon, S. Mallat, and A. W. Smeulders. Multiscale hierarchical convolutional networks. *arXiv preprint arXiv:1703.01775*, 2017.
[29] J. J. Koenderink and A. J. Van Doorn. The structure of locally orderless images. *International Journal of Computer Vision*, 31(2-3):159–168, 1999.
[30] A. Krizhevsky, I. Sutskever, and G. E. Hinton. ImageNet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.
[31] S. Lazebnik, C. Schmid, and J. Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *Computer vision and pattern recognition, 2006 IEEE computer society conference on*, volume 2, pages 2169–2178. IEEE, 2006.
[32] Q. V. Le, W. Y. Zou, S. Y. Yeung, and A. Y. Ng. Learning hierarchical invariant spatio-temporal features for action recognition with independent subspace analysis. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 3361–3368. IEEE, 2011.
[33] Y. LeCun, K. Kavukcuoglu, C. Farabet, et al. Convolutional networks and applications in vision. In *ISCAS*, pages 253–256, 2010.
[34] D. G. Lowe. Object recognition from local scale-invariant features. In *Computer vision, 1999. The proceedings of the seventh IEEE international conference on*, volume 2, pages 1150–1157. Ieee, 1999.
[35] J. Mairal, P. Koniusz, Z. Harchaoui, and C. Schmid. Convolutional kernel networks. In *Advances in neural information processing systems*, pages 2627–2635, 2014.
[36] S. Mallat. Group invariant scattering. *Communications on Pure and Applied Mathematics*, 65(10):1331–1398, 2012.
[37] S. Mallat. Understanding deep convolutional networks. *Phil. Trans. R. Soc. A*, 374(2065):20150203, 2016.
[38] S. G. Mallat. A theory for multiresolution signal decomposition: The wavelet representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(7):674–693, July 1989.
[39] M. Noroozi and P. Favaro. Unsupervised learning of visual representations by solving jigsaw puzzles. In *European Conference on Computer Vision*, pages 69–84. Springer, 2016.
[40] E. Oyallon. Building a regular decision boundary with deep networks. 2017.
[41] E. Oyallon, E. Belilovsky, and S. Zagoruyko. Scaling the Scattering Transform: Deep Hybrid Networks. In *International Conference on Computer Vision (ICCV)*, Venice, Italy, Oct. 2017.
[42] E. Oyallon and S. Mallat. Deep roto-translation scattering for object classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2865–2873, 2015.
[43] D. Pathak, P. Krahenbuhl, J. Donahue, T. Darrell, and A. A. Efros. Context encoders: Feature learning by inpainting. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2536–2544, 2016.
[44] F. Perronnin and D. Larlus. Fisher vectors meet neural networks: A hybrid classification architecture. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3743–3752, 2015.
[45] A. Radford, L. Metz, and S. Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.
[46] J. Sánchez and F. Perronnin. High-dimensional signature compression for large-scale image classification. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 1665–1672. IEEE, 2011.
[47] J. Sánchez, F. Perronnin, T. Mensink, and J. Verbeek. Image classification with the fisher vector: Theory and practice. *International journal of computer vision*, 105(3):222–245, 2013.
[48] T. Serre and M. Riesenhuber. Realistic modeling of simple and complex cell tuning in the hmax model, and implications for invariant object recognition in cortex. Technical report, DTIC Document, 2004.
[49] L. Sifre and S. Mallat. Rotation, scaling and deformation invariant scattering for texture discrimination. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1233–1240, 2013.
[50] J. T. Springenberg, A. Dosovitskiy, T. Brox, and M. Riedmiller. Striving for simplicity: The all convolutional net. *arXiv preprint arXiv:1412.6806*, 2014.
[51] R. K. Srivastava, K. Greff, and J. Schmidhuber. Highway networks. *arXiv preprint arXiv:1505.00387*, 2015.
[52] M. Sugiura. *Unitary representations and harmonic analysis: an introduction*, volume 44. Elsevier, 1990.
[53] K. Swersky, J. Snoek, and R. P. Adams. Multi-task Bayesian optimization. In *Advances in neural information processing systems*, pages 2004–2012, 2013.
[54] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus. Intriguing properties of neural networks. In *International Conference on Learning Representations*, 2014.
[55] E. Tola, V. Lepetit, and P. Fua. Daisy: An efficient dense descriptor applied to wide-baseline stereo. *IEEE transactions on pattern analysis and machine intelligence*, 32(5):815–830, 2010.
[56] I. Waldspurger. *These de doctorat de l'Ecole normale supérieure*. PhD thesis, École normale supérieure, 2015.
[57] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson. How transferable are features in deep neural networks? In *Advances in neural information processing systems*, pages 3320–3328, 2014.
[58] S. Zagoruyko. 92.45% on cifar-10 in torch. *Torch Blog*, 2015.
[59] S. Zagoruyko and N. Komodakis. Wide residual networks. In *BMVC*, 2016.
[60] M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. In *European conference on computer vision*, pages 818–833. Springer, 2014.
[61] J. Zhao, M. Mathieu, R. Goroshin, and Y. LeCun. Stacked what-where auto-encoders. *arXiv preprint arXiv:1506.02351*, 2016.

Edouard Oyallon is a faculty member at CentraleSupelec, University of Paris-Saclay. He completed his PhD at the Ecole Normale Supérieure (ENS) in 2017.

Sergey Zagoruyko is a PhD student at Ecole des Ponts ParisTech.

Gabriel Huang is a PhD student at MILA and DIRO at the University of Montreal.

Nikos Komodakis is a faculty member at Ecole des Ponts ParisTech.

Simon Lacoste-Julien is a faculty member at MILA and DIRO at the University of Montreal, and a CIFAR fellow.

Matthew Blaschko is a faculty member in the Electrical Engineering department at KU Leuven.

Eugene Belilovsky is a postdoctoral fellow at MILA at the University of Montreal. He completed a joint PhD at CentraleSupelec and at KU Leuven.