A HYBRID NETWORK: SCATTERING AND CONVNET

Edouard Oyallon
Département Informatique
Ecole Normale Supérieure
Paris, France
edouard.oyallon@ens.fr

ABSTRACT

This paper shows how, by combining prior and supervised representations, one can create architectures that lead to nearly state-of-the-art results on standard benchmarks, which mean they perform as well as a deep network learned from scratch. We use scattering as a generic and fixed initialization of the first layers of a deep network, and learn the remaining layers in a supervised manner. We numerically demonstrate that deep hybrid scattering networks generalize better on small datasets than supervised deep networks. Scattering networks could help current systems to save computation time, while guaranteeing the stability to geometric transformations and noise of the first internal layers. We also show that the learned operators explicitly build invariances to geometrical variabilities, such as local rotation and translation, by analyzing the third layer of our architecture. We demonstrate that it is possible to replace the scattering transform by a standard deep network at the cost of having to learn more parameters and potentially adding instabilities. Finally, we release a new software, ScatWave, using GPUs for fast computations of a scattering network that is integrated in Torch. We evaluate our model on the CIFAR10, CIFAR100 and STL10 datasets.

1 INTRODUCTION

Deep architectures builds generic and low-dimensional representations that lead to state-of-the-art results on tasks such as classification (He et al., 2015), games (Silver et al., 2016), or generative models (Radford et al., 2015). These architectures are designed as cascades of non-linear modules that are fully learned. This paper addresses several questions: is it necessary to learn each module? Can a scattering networks replace the first layers? What are the potential benefits?

Hybrid architectures composed of a supervised representation learned on top of an unsupervised representation (Philbin et al., 2007) have been progressively abandoned for the end-to-end training approach (LeCun et al., 2010). Understanding the nature of the cascade of deep operators is difficult (Szegedy et al., 2013), since they are learned via back-propagation, and not layer-wise. However, the learned features appear to be transferable to other datasets and helpful for classification(Zeiler & Fergus, 2014), which implies that the learned representations have captured generic properties for image classification tasks.

Scattering representations (Mallat, 2012) are predefined and generic representations which only require the learning of a few hyper parameters. They consist of a cascade of wavelet transforms and modulus nonlinearities are have proven to be successful in classification tasks such as textures (Bruna & Mallat, 2013b; Sifre & Mallat, 2013), small digits (Bruna & Mallat, 2013b), sounds (Andén & Mallat, 2014) or complex image datasets with unsupervised representations (Oyallon & Mallat, 2015). Nevertheless, these representations do not adapt to the specific bias of each dataset and there is a huge performance gap between supervised and unsupervised representations(Oyallon & Mallat, 2015).

A convnet is typically a cascade of convolutional layers and nonlinearities, followed by a final average pooling or a sequence of fully connected layers. They lead to state of the art results on CIFAR10 and CIFAR100 (Zagoruyko & Komodakis, 2016). Some related work to ours (Perronnin & Larlus, 2015) proposed to replace the first layers of convolution by a combination of SIFT and Fisher vec-

tors, while learning on top of it a cascade of fully connected layers. This is a hybrid representation in the sense that it combines an unsupervised learned representation and a supervised learned MLP. The numerical results they obtained are competitive on ImageNet with the first AlexNet architecture (Krizhevsky et al., 2012), while saving computations.

Training a state of the art deep network requires a huge amount of labeled data. Several works tried to tackle this difficulty by developing unsupervised algorithm applied to deepnetwork: for instance evaluating on CIFAR10 an unsupervised generative adversarial method (GAN) pretrained on a subset of Imagenet-1K (Radford et al., 2015). In a setting where few annotated data are available, training a deep network is hard and requires a lot of regularization, yet a semisupervised learning algorithm applied to a GAN can improve even more the accuracy on CIFAR10, as in Salimans et al. (2016). Yet, if few data are available, such as in medical imaging, training a deep network from scratch is more complicated: one can only use imagenet pre-trained features (Carneiro et al., 2015).

Section 2 describes our model, which is a cascade of a scattering network and a convnet. We explain how we build our scattering network, describe its stability properties and exhibit our learning pipeline. Section 3 shows that our network provides competitive results on CIFAR10, CIFAR100 and STL10, while having theoretical guarantees for its representations, in both setting with limited data or not. The experiments can be reproduced using ScatWave [1], an implementation of our algorithm in Torch, which we make publicly available. More details about the software are available in the Appendix A.

2 TOWARDS A HYBRID ARCHITECTURE

We construct an architecture that consists of two blocks: the first is based on the scattering transform and involves no learning; the second is a classical convnet. In this section, we describe these architectures and their properties.

2.1 SCATTERING NETWORK

A scattering network belongs to the class of convolutional networks whose filters are predefined as wavelets (Oyallon & Mallat, 2015). The construction of this network has mathematical foundations (Mallat, 2012), meaning it is well understood, relies on few parameters and is stable, in contrast deep networks. Stability properties are discussed in Subsection 2.1.2 and Apprendix B. Besides, most of the parameters of this representation does not need to be adapted to the bias of the dataset (Oyallon & Mallat, 2015), making it a suitable generic representation.

2.1.1 A CASCADE OF WAVELETS AND MODULUS

In this section, we briefly recall the definition of the scattering transform. It is the cascade of wavelet transforms, and modulus nonlinearity which is finally spatially averaged. Since a modulus is non-expansive, and a wavelet transform is a linear isometry, a scattering transform is also non-expansive. The local averaging of this representation thus builds a local invariance to translation. In this paper, we only consider a second order scattering network, on the group of translations.

Consider a signal $x(u), u \in \mathbb{R}^2$ and an integer $J \in \mathbb{N}$, which is the spatial scale of our scattering transform. Let ϕ_J be an averaging with a spatial window of scale 2^J. (for example, a Gaussian averaging) Applying a subsampled averaging $A_J x(u) = x \star \phi_J(2^J u)$ builds an approximate invariant to translations smaller than 2^J, but it also results in a loss of high frequencies that are necessary to discriminate signals. We define $S_0 x = A_J x$ as the order 0 scattering.

A solution to avoid this loss is provided by wavelets. A wavelet is an integrable and localized function in the Fourier and space domain, with a 0 average. A family of wavelets is obtained by dilating a complex mother wavelet ψ (for example, 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. A given wavelet $\psi_{j,\theta}$ has thus its energy concentrated at a scale j, in the angular sector θ. Let $K \in \mathbb{N}$ be an integer representing the number of angles of our operator. A wavelet transform is the convolution of a signal with the family of wavelets introduced above, with an appropriate downsampling, i.e.

[1] Code can be found here: https://github.com/edouardoyallon/scatwave

$W_1 x(u, \theta_1, j_1) = \{x \star \psi_{j_1,\theta_1}(2^{j_1}u)\}_{j \leq J, \theta = 2\pi \frac{l}{L}, 1 \leq l \leq L}$. Observe that j and θ have been discretized: the wavelet is chosen to be selective in angle and localized in Fourier, thus the sampling is chosen such that $(\theta_1, j) \to W_1 x(u, \theta_1, j_1)$ is regular enough. Besides, the wavelet transform has been spatially oversampled by a factor 1. The wavelet parameters and this discretization were already chosen in (Oyallon & Mallat, 2015), where this representation is shown to be generic, so we have used the same hyper-parameters. In their case, $\{A_J x, W_1 x\}$ is approximatively an isometry on the set of signals with limited bandwidth, and this implies the energy of the signal is preserved. This operator belongs to the category of multi-resolution analysis operator, each filter being excited by a specific scale and angle, but the output coefficients are not invariant to translation. We can not apply A_J to $W_1 x$ since it gives a trivial invariant, namely 0.

We build the first order scattering coefficients. Applying a point-wise modulus to $W_1 x$, followed by an averaging A_J allows us to build an invariant. If the mother wavelet is analytic, then $|W_1 x|$ is more regular (Bernstein et al., 2013) which implies that the support in Fourier of $|W_1 x|$ is more likely to be contained in a lower frequency domain than $W_1 x$. Thus, A_J preserves the energy of $|W_1 x|$. In this case, it is possible to define $S_1 x = A_J |W_1| x$, which can also be written as: $S_1 x(u, \theta_1, j_1) = |x \star \psi_{j_1,\theta_1}| \star \phi_J(2^J u)$; this is the order 1 scattering. It is consequently invariant to translation up to 2^J.

Once more, applying a second wavelet transform $W_2 = W_1$ on each channels permits the recovery of the high-frequency loss due to the averaging applied to the first order, leading to $S_2 x = A_J |W_2||W_1|$, which can also be written as $S_2 x(u, \theta_1, \theta_2, j_1, j_2) = ||x \star \psi_{j_1,\theta_1}| \star \psi_{j_2,\theta_2}| \star \phi_J(2^J u)$. We only compute increasing paths, i.e. $j_1 < j_2$ because non-increasing paths bear no energy (Bruna & Mallat, 2013b). We do not compute higher order scatterings, because their energy has been shown experimentally not to be meaningful (Bruna & Mallat, 2013b).

2.1.2 Covariance and stability of the representation

In this section, we develop mathematical properties that are obtained by wavelets. Covariance with a group of variability permits building a localized invariant via local averaging. The degree of invariance will be decided by a supervised algorithm, in order to be adapted to the bias of the problem of classification. Here, the parameter J corresponds to a trade-off of invariance in translation and discrimination to adjust A_J, and it has to be learned from the data. By construction, as a cascade of convolutions, a scattering network is covariant with translations. Let $r_\theta.x \triangleq x(r_{-\theta}u)$ be a rotated signal by θ. The representation is still covariant with the rotation in the following sense:

$$S_1(r_\theta.x)(u, \theta_1) = S_1 x(r_{-\theta}u, \theta_1 - \theta) \triangleq r_\theta.(S_1 x)(u, \theta_1)$$

$$S_2(r_\theta.x)(u, \theta_1, \theta_2) = S_2 x(r_{-\theta}u, \theta_1 - \theta, \theta_2 - \theta) \triangleq r_\theta.(S_2 x)(u, \theta_1, \theta_2)$$

However, for $S_2 x$, the natural set of coordinates that gives a rotational invariant for angles is in fact given by:

$$\tilde{S}_2 x(u, \theta_1, \alpha) = S_2 x(u, \theta_1, \theta_1 + \alpha)$$

which naturally leads to:

$$\tilde{S}_2(r_\theta.x)(u, \theta_1, \alpha) = \tilde{S}_2 x(r_{-\theta}u, \theta_1 - \theta, \alpha) \triangleq r_\theta.(S_2 x)(u, \theta_1, \alpha)$$

In fact, this representation is covariant with the action of the roto-translation group, i.e. $\mathbb{R}^2 \ltimes SO_2$ (Sifre & Mallat, 2013). In addition, it can be proven that a scattering network linearizes small deformations (Mallat, 2012), which means that a linear operator can build invariants to a subset of deformations. It means also that a deep network could reduce the perturbations due to this variability. Furthermore, this representation is complete, in the sense that it is possible to reconstruct a signal from its scattering coefficients (Bruna & Mallat, 2013a).

It is also important to note that a scattering transform is non-expansive, as a cascade of non-expansive operators, e.g.: $\|Sx - Sy\| \leq \|x - y\|$. Thus, this representation is stable to additive noises, which correspond to perturbations studied in Goodfellow et al. (2016); Moosavi-Dezfooli et al. (2015); Szegedy et al. (2013); Goodfellow et al. (2014). By adding a very small quantity to an image, the classification performed by a deep network can be fooled, e.g. image x is well classified, but one can find $\|\epsilon\| \ll 1$ such that $x + \epsilon$ is not, with the two images being visually indistinguishable.

Apprendix B mathematically quantifies the stability of a hybrid network and proves the unstability potentially rises from the cascaded learned deep network. Indeed, Szegedy et al. (2013) reports that the operators of the different layers of a deep network are not contractive. Corrections were added to fix this by training a deep network on the fooling examples (Goodfellow et al., 2014), but this requires additional computations. With wavelets as an initialization, instabilities cannot occur in the first layers contrary to Szegedy et al. (2013), since the operator is non-expansive.

2.2 Adding supervision to our representation: cascading a convnet

This section introduces our hybrid representation, and explain its interest. We first justify why applying a deep network after scattering is natural. Scattering transforms have yielded excellent numerical results on datasets where the variabilities are completely known, such as MNIST or FERET. In these task, we only encounter the problem of sample variance and handling the variance leads to solving the problem. However, in classification tasks on more complex image datasets, such variabilities are only partially known. Applying the scattering transform on datasets like CIFAR or Caltech leads to nearly state-of-the-art results in the unsupervised case (Oyallon & Mallat, 2015). But there is a huge gap in performance when comparing to supervised representations, which deep networks can fill in.

We now explain why the scattering transform is an appropriate initialization. Recent works (Mallat, 2016; Bruna et al., 2013) have suggested that a deep network could build an approximation of the group of symmetries of a classification task and apply transformations along the orbits of this group. The objective of a supervised classifier is to reduce the variabilities due to those symmetries. To each layer corresponds an approximated Lie group of symmetry, and this approximation is progressive, in the sense that the dimension of this approximation is increasing with depth. For instance, the linear Lie group of symmetry of an image is the translation group, $\mathbb{R}^2$. If no non-linearity is applied, it is not possible to discover new linear groups of symmetry for natural images. In the case of a wavelet transform obtained by rotation of a mother wavelet, it is possible to recover a new subgroup of symmetry, the rotation SO_2, and the group of symmetry at this layer is the roto-translation group: $\mathbb{R}^2 \ltimes SO_2$. Discovering the next groups of symmetry is however a difficult task; nonetheless, the roto-translation group seems to be a good initialization for the first layers. In this work, we investigate this hypothesis.

We thus build a standard deep convolutional network on top of the scattering transform. Its architecture is represented in Figure 2.2. Our network consists of a cascade of $2C$ convolutions with spatial kernel size 3×3. The C first convolutions use K_0 input channels, except for the first layer, while the C next convolutions have K_1 output channels, except for the last layer. The number of inputs of the first layer is equal to the size of the scattering features, whereas the last layer consists in an average pooling followed by a linear projection. We used a ReLU non-linearity, and no non-linear pooling is involved in our architecture. Observe that if the scattering is applied up to a scale J, then the signal is at spatial resolution J, i.e. its sampling is 2^J, allowing faster computation. In the next section, we discuss the value of those parameters.

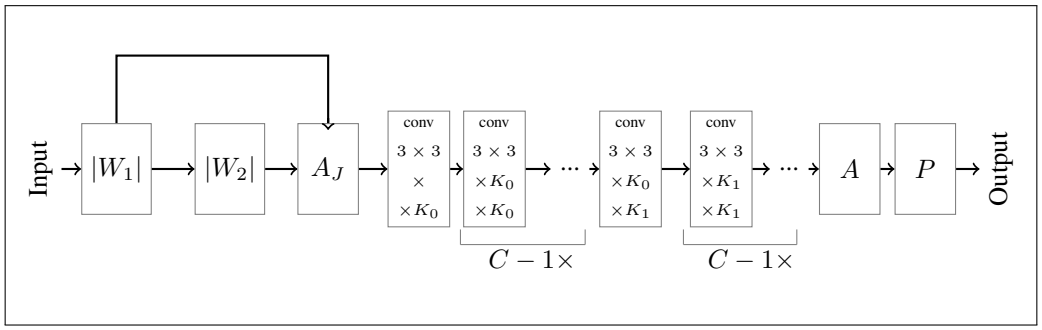

Figure 1: Architecture of our deep network. Observe that no downsampling is performed

Notably, the first layer F_1 of this deep convolutional network is structured by its input, the scattering representation. The nature of this operator and the features selected by this supervised algorithm will be discussed in the next sections.

3 EXPERIMENTS

We compare our algorithm to supervised, unsupervised and semi-supervised deep networks and evaluate them on the CIFAR10, CIFAR100 and STL10 datasets. CIFAR10 and CIFAR100 datasets consist of colored images of size 32×32, with 50000 images in the training set, and 10000 in the test set. CIFAR10 and CIFAR100 have respectively 10 and 100 classes. SLT10 dataset consists of colored 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 unlabeled images of this dataset were not used during our experiments. The three datasets are whitened as preprocessing, following the standard procedure. Our software, ScatWave, is implemented in Torch and is publicly available online [2].

3.1 EXPERIMENTAL RESULTS

3.1.1 METHODOLOGY

During all our experiments, we have trained our architecture with SGD with momentum 0.9 to minimize the standard negative cross-entropy. The batch size is 128. We have applied four types of regularization. First, 0.6 dropout is used after each consecutive two layers. Secondly, we have used 10^{-4} weight decay. Futhermore, we have used batch normalization techniques which are supposed to lead to a better conditioning of the optimization (Ioffe & Szegedy, 2015). Finally, we have augmented the dataset by using random cropping and flipping. The initial learning rate is 0.25 and we divide it by two after every 30 epochs. The networks are trained during 300 epochs.

We now depict the selected hyperparameters of our architecture for CIFAR and STL10 datasets respectively, which we have kept fixed for all experiments unless specifically stated otherwise. For the CIFAR datasets, using cross-validation, the parameters of invariance are set to $J = 2$ and the number of angles used is $L = 8$ for the scattering transform. In this case, the output of the scattering network is a tensor of size $\frac{N}{2^J} \times \frac{N}{2^J} \times 3(1 + LJ + \frac{1}{2}L^2J(J-1)) = 8 \times 8 \times 243$, after reshaping. For the deep net architecture, we chose to use $C = 10$ layers and $K_0 \in \{128, 512\}, K_1 = 128$ channels. The number of parameters for CIFAR10 is $9 \times (243 \times K_0 + (C-1)K_0^2 + K_0K_1 + (C-1)K_1^2) + 10 \times K_1)$, which is roughly equal to $1.6M$ and $12M$ parameters for $K_0 = 128$ and $K_0 = 512$ respectively. For the STL10 dataset, we chosed $K_0 = K_1 = 512$ and $C = 2$: the deep network is shallower to compensate the speed loss due to the use of larger images. In the following, the symbol $\%$ corresponds to an absolute percentage of accuracy.

3.1.2 NUMERICAL EXPERIMENTS ON THE ENTIRE DATASET

We report the classification accuracies in Table 3.1.2 and discuss them below. We compare our architecture with the unsupervised scattering architecture (Oyallon & Mallat, 2015). The roto-translation scattering is almost identical to scattering, except that it recombines the channels along the rotation axis by applying a wavelet transform along angles, building more complex geometrical invariants. The classifier of this paper is a RBF SVM kernel, which can be interpreted as a two-layer deep neural network. For the sake of simplicity, we have trained on top of our scattering network a 3 layer fully connected network with size 2048, similarly to (Perronnin & Larlus, 2015). Without data augmentation, the accuracy of the network is respectively 3.7% and 8.9% below the roto-translation scattering on CIFAR10 and CIFAR100. However, applying data augmentation with translation of length less than 2^2 permits recovering this loss in accuracy, resulting in accuracies of 83.0% and 56.7% (we let the network train for 400 epochs here) respectively on CIFAR10 and CIFAR100. One major difference to our approach is that the system in Oyallon & Mallat (2015) uses a large amount of oversampling. This suggests that in order to learn the appropriate features it is necessary to average the small translation displacement; even if the representation is built to be invariant to translation, its construction relies, for fast computation, on an approximative (but justified) downsampling that leads to non-linear aliasing.

Applying a supervised convnet significantly improves the accuracy of the scattering with 3 fully connected layers, and leads to comparable results with other supervised deep representations: 91.4% on CIFAR10 and 69.5% on CIFAR100. We compare our work with the Highway network (Srivastava et al., 2015), that consists in a deep cascade of 19 linear and non-linear operators, with an extensive

[2] https://github.com/edouardoyallon/scatwave

Table 1: Accuracy of scattering compared to similar architectures on CIFAR10

Architecture	Accuracy
Unsupervised architectures	
Roto-translation scattering	82.3
Scattering (ours) + 3 FC	
+no data augmentation	78.6
Scattering (ours) + 3 FC	**83.0**
Supervised hybrid architectures	
Scattering (ours) + CNN, $K = 128$	89.4
Scattering (ours) + CNN, $K = 512$	**91.4**
Supervised architectures	
Highway network	92.4
All-CNN	92.8
Wide ResNet	**96.2**

Table 2: Accuracy of scattering compared to similar architectures on CIFAR100

Architecture	Accuracy
Unsupervised architectures	
Roto-translation scattering	**56.8**
Scattering (ours) + 3 FC	
+ no data augmentation	47.9
Scattering (ours) + 3 FC	56.7
Supervised hybrid architectures	
Scattering (ours) +CNN, $K = 128$	64.4
Scattering (ours) +CNN, $K = 512$	**69.5**
Supervised architectures	
Highway network	67.8
All-CNN	66.3
Wide ResNet	**81.7**

data augmentation. Besides, since our convnet is kept as simple as possible, we also compare our architecture to the All-CNN (Springenberg et al., 2014) work. The latter performs slightly better on CIFAR10, but our hybrid network has a better generalization on CIFAR100. This indicates that supervision is essential, but that a geometric initialization of the first layers of a deep network leads to a discriminative representation as well. It is also interesting to observe that we could not find any architecture that was performing worse on CIFAR10 than ours, but better on CIFAR100. Since the number of samples available per class is lower, this could indicate that learning is easier in this case with a scattering initialization.

A Wide Resnet (Zagoruyko & Komodakis, 2016) outperforms our architecture by 4.8% on CIFAR10 and 12.2% on CIFAR100, but it requires more engineering process to be designed and is deeper. It is important to recall that, contrary to the wide Resnet, there are no instabilities due to geometric transformations (e.g. translations or deformations), and that the two first layers which are responsible for a large fraction of the computation are not learned, resulting in computational savings. Obviously, the scattering layers do not suffer from the vanishing or exploding gradient issues.

3.1.3 Numerical experiments on a small subset of the data

Supervised deep networks trained on small datasets easily overfit, for instance in the case of medical imaging where little data are available. Semisupervised algorithms exhibit good performances (Salimans et al., 2016), but it requires a large amount of unlabeled data to work. In this subsection, we show the benefit of using scattering in a framework where those data are not available. We demonstrate that scattering does prevent overfitting on small datasets, while keeping the same architecture and training methodology: this saves time to design an architecture.

For this experiment, we draw several random subsets of CIFAR10 for training our network, and used the same splits for each experiments to train a supervised deep network. Namely, we used a Network in Network (NiN) (Lin et al., 2013), VGG-like Simonyan & Zisserman (2014), Wide ResNet Zagoruyko & Komodakis (2016), which perform better than our network on the full dataset, and we use an implementation available online[3]. The VGG did not converge in this situation with the given hyperparameters (such as the depth) and for a simple and fair comparison we decided not to adapt them. We however applied a data augmentation to the inputs of the NiN, VGG and ResNet by translation and flipping. Table 3 corresponds to the averaged accuracy over 5 different subsets, with the corresponding standard deviation. We compare the two architectures with a semi-supervised model that consists in a GAN (Salimans et al., 2016), that is trained on all the data of CIFAR10 yet only a fraction is labeled. With 4000 and 8000 labeled samples, a wide ResNet with 40 layers outperforms by at least 3% the supervised and unsupervised methods which indicate this architecture suffer from less overfitting than the others. If less than 2000 samples are available, a hybrid network outperforms all the supervised architectures: the difference of accuracy between the hybrid architecture and the others is progressively favorable to the hybrid network when the number

[3] http://torch.ch/blog/2015/07/30/cifar.html

Table 3: Accuracy of a hybrid scattering in a limited sample situation on CIFAR10 dataset. N.A. and N.C. stands respectively for Not Available and Not Converged.

Architecture	1000	2000	4000	8000	50000
Supervised architecture					
Scattering+CNN, $K = 512$	**58.5±1.2**	**69.4±0.5**	76.2±0.4	81.8±0.7	91.4
NiN	54.8±1.0	65.1±0.7	71.2±3.8	79.0±3.8	91.9
VGG	N.C.	N.C.	N.C.	N.C.	92.5
Wide ResNet	58.0±1.2	68.9±1.4	**79.1±0.4**	**86.4±0.3**	**96.4**
Semi-supervised architecture					
GAN	79.2±2.0	80.4±2.1	82.4±2.3	83.3±1.8	N.A.

Table 4: Accuracy of a hybrid scattering on the STL-10 dataset

Architecture	Accuracy
Supervised architectures	
Scattering+CNN, $K = 512$	**77.4± 0.4**
CNN	70.1
Semi-supervised and unsupervised architecture	
Exemplar CNN	75.4±0.3
Unsup. Discr. CNN	76.8±0.3

of samples decreases. Nonetheless, a GAN performs better than a translation scattering with 3 fully connected layers; its performances is almost constant equal to 80%, which shows that this algorithm can adapt itself to the bias of the dataset.

In a second experiment, we apply our hybrid architecture on the STL10 dataset, which is a challenging dataset in the limited sample situation since only 500 samples per class are available. The averaged result is reported in Table 4. A supervised CNN (Swersky et al., 2013) whose hyper parameters have been automatically tuned achieves 70.1% accuracy. Using the unlabeled data improves by at least 5% the accuracy of a CNN. For an Exemplar CNN (Dosovitskiy et al., 2014) or an Unsupervised Discriminative CNN (Huang et al., 2016), the weights of the CNNs are unsupervisedly learned from patches of images. Those techniques add several hyper parameters and require an additional engineering process. Applying a hybrid network is straightforward and outpasses both the supervised and unsupervised state of the art by 7.3% and 0.6% respectively.

3.2 Retraining modules of the architecture

In this section, we show the benefit of using a structured representation. For instance, the first layer of a convnet cascaded on top of a scattering network inherits from the structure of the scattering coefficents. In all the following experiments, we use a converged hybrid network according to the procedure detailed in Subsection 3.1.1.

3.2.1 Simplifying the third layer of a deepnetwork

We numerically analyze the nature of the operations performed along angles by the first layer F_1 of our deep network on CIFAR10. Let us define as $F_1 x = \{F_1^0 x, F_1^1 x, F_1^2 x\}$ the components associated to the order 0,1,2 scattering coefficients respectively. Let $1 \leq k \leq K$. In this case, F_1^0 is a convolutional operator that depends on the variables (u, k), F_1^1 depends on (u, θ_1, k), and F_1^2 depends on (u, θ_1, α, k). We would like to characterize the smoothness of these operators with respect to the variables, because our representation is covariant to rotations.

To this end, we consider the Fourier transform, for each channels k. 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, α) respectively. In space, we applied a DCT transform restrained to its support which is similar to a Fourier transform, since the support

of the operator is small. Then the operator is expressed in the tensorial Frequency domain. Since those operations are involutive (up to constants and reflexions), one can easily recover the original operator.

Let us demonstrate how to sparsify this operator in its frequency basis. First, it is possible to threshold by ϵ the coefficients of the operators in the Fourier domain, i.e. we replaced the operators $\hat{F}_1^1$, $\hat{F}_1^2$ by $1_{|\hat{F}_1^1|>\epsilon}\hat{F}_1^1$ and $1_{|\hat{F}_1^2|>\epsilon}\hat{F}_1^2$. Before thresholding, all the frequencies were excited. After this operation, approximatively 10% of coefficients are non-zero. We have tested our network without any retraining and observed a negligible loss of accuracy. We proved that this basis permits a sparse approximation of our filters.

Furthermore, we decided to keep only the two first frequencies of F_1 along the variable θ_1, i.e. we replaced the operators $\hat{F}_1^1$, $\hat{F}_1^2$ by $1_{|\omega_{\theta_1}|\leq 1}\hat{F}_1^1$, $1_{|\omega_{\theta_1}|\leq 1}\hat{F}_1^2$. This results in a loss of accuracy of roughly 5%. We then fix the layer F_1, and decide to retrain the next layers, keeping the entire training procedure identical again with an initial learning rate of 1. We obtain a classification accuracy which is 3% below the original architecture. This indicates that the network can almost recover discriminative information from averaged coefficients in angle.

The first experiment shows that in a natural basis it is possible to sparsify the operator. The last experiment indicates that most of the operations performed by the first layer of our network are smooth since they are localized in the Fourier space. This is equivalent to first projecting via an angular averaging each scattering coefficient and it suggests that the system autonomously builds an invariant to geometrical variabilities. While this does not prove that nomore geometrical operators are applied in the next layers, but it does show it is possible to obtain a good accuracy (3% below the original result) with a representation after F_1 which is stable to local roto-translation and deformation variabilities, thanks to a roto-translation averaging.

3.2.2 Replacing the scattering network by a convnet

Many theoretical arguments of deep learning rely on the universal approximation theorem (Cybenko, 1989). 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 on Figure 3.2.2, 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. We keep our architecture identical, except that we replace the scattering part by this network. Then we retrain it, keeping the weights of all the other layers constant and equal to the optimal solution found with the scattering in the previous section. 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 is convincing. Using a shallower network seems to degrade the performances, but we did not investigate more this question. Besides, the learned network will not have any guarantee of stability properties.

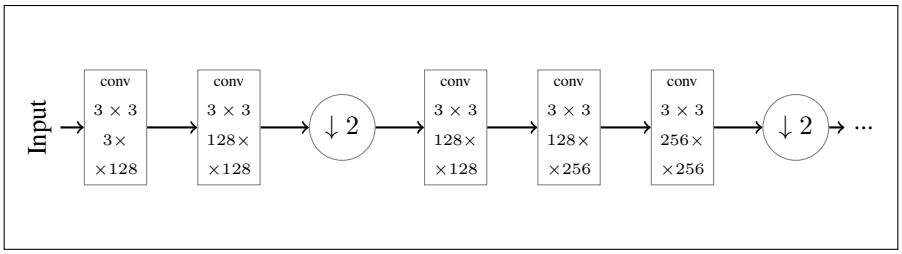

Figure 2: Architecture of our deep network that mimics a scattering network.

4 CONCLUSION

We proposed a new deep hybrid architecture that involves scattering and convnets and is competitive with existing approaches. We demonstrate its good generalization performances on CIFAR10, CIFAR100, STL10 and subsets of CIFAR10, CIFAR100. We showed that the cascaded convnet learns an invariant to roto-translation, and that it is possible to learn a deep network that mimics the scattering, at the cost to potentially create instabilities. We release also a fast software to compute a scattering transform on GPUs.

This is a preliminary work whose results must be extended to ImageNet. This paper was dedicated to incorporate geometry into deep networks, and we will show how they refine their construction of class invariants in a future work.

ACKNOWLEDGMENTS

The author would like to thank Mathieu Andreux, Eugene Belilovsky, Carmine Cella, Bogdan Cirstea, Michael Eickenberg, Stéphane Mallat and Sergey Zagoruyko, for helpful discussions and support. Also, the author especially thanks Carmine Cella for nice suggestions of experiments. This work is funded by the ERC grant InvariantClass 320959 and via a grant for PhD Students of the Conseil régional d'Ile-de-France (RDM-IdF).

Table 5: Computation time (in seconds) for different input size, one being with MATLAB on CPUs and the other with Torch on GPUs

Input size	**J**	**ScatNetLight (in s)**	**ScatWave (in s)**
$32 \times 32 \times 3 \times 128$	2	2.5	0.15
$32 \times 32 \times 3 \times 128$	4	13	0.49
$32 \times 32 \times 3 \times 128$	5	38	1.1
$128 \times 128 \times 3 \times 128$	2	16	1.0
$128 \times 128 \times 3 \times 128$	4	52	2.3
$128 \times 128 \times 3 \times 128$	5	120	3.7
$256 \times 256 \times 3 \times 128$	2	160	2.2

APPENDIX A: IMPLEMENTATION DETAILS OF SCATWAVE

Cascade of multi-resolution computations, such as perfomed in a scattering transform, are delicate. The bottleneck of the scattering on CPU was either speed and memory. It is thus necessary to quickly explain our algorithm: we show that by reorganizing the order of the computation of the algorithm, one can speed them up.

Computing a scattering transform at order 2 requires computing each path $||x \star \psi_{p_1}| \star \psi_{p_2}|$ where p_1, p_2 are the parameters with increasing scales of the filters. Computing each path can be viewed as a computational tree, where the coefficient of the scattering transform before an averaging are the leaf of the tree, and the internal nodes are the modulus of the intermediary wavelet transform. The way the tree is walked affects the computation time. In ScatNet (Andén et al., 2014), the traversal is done by first computing each internal node, storing the results of each internal node. In a second steps, the leafs are computed and stored. In terms of memory, this is not optimal since it requires storing the intermediate computations. Instead, we use an affix traversal of the tree. It reduces at its minimal the memory used, and allow the use of GPUs.

ScatWave is a GPU version of the scattering networks in Torch, that is based on our observation above. ScatNetLight is a MATLAB version on CPUs, which uses as much as possible multithreads. The Table 5 reports the difference in computation time, for identical parameters and output representations (e.g. same sampling, same hyper parameters). The input corresponds to batches of 128 tensor, the two first dimensions being the size of the image, and the third the number of elements in the tensor (e.g. the color in the case of images) . For comparisons, we used a machine with 24 cores and a TiTan GPU. The speed-up is at least of $\times 15$ in all cases, and up to $\times 70$: ScatWave uses the library cuFFT.

APPENDIX B: A NOTE ON THE STABILITY OF A HYBRID DEEP NETWORK

In this section, we recall the notion of additive stability of a deep network and derive some simple properties that shows that instabilities are due to the cascaded deepnetwork, and we demonstrate bounds to quantify them. Instabilities are due to perturbations of an initial sample that a deepnetwork does not reduce correctly: the modification is visually not significant or does not affect the label of the perturbated sample, yet the network incorrectly classifies the image. We consider a (trained) deep network f, with bounded outputs, that are for instance the probabilities output obtained after a sigmoid function. We write label(x) the labels computed by this deepnetwork for an input x. It is possible to define the contraction factor of a deep network f via:

$$\Delta(f) = \sup_{\text{label}(x)\neq\text{label}(y)} \frac{\|f(x)-f(y)\|}{\|x-y\|}$$

Observe that label$(x) \neq$ label$(y) \Rightarrow x \neq y$. A small value of $\Delta(f)$ indicates a better stability. It is possible to introduce a local version of this definition, that depends on the input sample (Moosavi-Dezfooli et al., 2015). It is consistant with the definitions of Goodfellow et al. (2016); Moosavi-Dezfooli et al. (2015); Szegedy et al. (2013); Goodfellow et al. (2014), in the sens that the numerator is bounded, but the denominator might become arbitrary small. Let us now consider a hybrid network, e.g. for an input x, the output is $f(Sx)$ where $Sx \in \mathbb{R}^n$ is its scattering transform. Let us write $\mathcal{S} = \{Sx, x\}$ the span of a scattering transform, which corresponds to a strict embedding in the standard euclidean space $\mathbb{R}^n$, e.g. $\mathcal{S} \subsetneq \mathbb{R}^n$.

Proposition 1. *The following bounds stand:*

$$\Delta(f \circ S) \leq \sup_{\substack{\textit{label}(\tilde{x})\neq\textit{label}(\tilde{y}) \\ (\tilde{x},\tilde{y})\in\mathcal{S}^2}} \frac{\|f(\tilde{x})-f(\tilde{y})\|}{\|\tilde{x}-\tilde{y}\|} \leq \Delta(f)$$

Proof. Let x, y two samples with different estimated labels, then $Sx \neq Sy$. In this case,

$$\frac{\|f(Sx)-f(Sy)\|}{\|x-y\|} = \frac{\|f(Sx)-f(Sy)\|}{\|Sx-Sy\|} . \frac{\|Sx-Sy\|}{\|x-y\|}$$

Thanks to the non-expansivity property of a Scattering transform,

$$\frac{\|Sx-Sy\|}{\|x-y\|} \leq 1$$

Setting $\tilde{x} = Sx, \tilde{y} = Sy$ and taking the supremum ends the demonstration. □

One sees that the amplitude of the resulting instabilities depends on the class of f. Furthermore, the inequalities might be strict and the bounds tighter, but there is no reason it does occur. However, it suggests that such instabilities could be removed if additional constraints were added during the training phase of f.

The deformation transformations are as well a class of instabilities. The cited works (Goodfellow et al., 2016; Moosavi-Dezfooli et al., 2015; Szegedy et al., 2013; Goodfellow et al., 2014) do not consider them, however, since the scattering transform linearizes them, one sees it is possible for a deepnetwork to explicitly build such invariance. Actually, the average along rotation and translation variables observed in Subsection 3.2.1 seems to indicate it is likely to occur.

REFERENCES

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]*, 2014.

Joakim Andén and Stéphane Mallat. Deep scattering spectrum. *IEEE Transactions on Signal Processing*, 62(16):4114–4128, 2014.

Swanhild Bernstein, Jean-Luc Bouchot, Martin Reinhardt, and Bettina Heise. Generalized analytic signals in image processing: comparison, theory and applications. In *Quaternion and Clifford Fourier Transforms and Wavelets*, pp. 221–246. Springer, 2013.

Joan Bruna and Stéphane Mallat. Audio texture synthesis with scattering moments. *arXiv preprint arXiv:1311.0407*, 2013a.

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

Joan Bruna, Arthur Szlam, and Yann LeCun. Learning stable group invariant representations with convolutional networks. *arXiv preprint arXiv:1301.3537*, 2013.

Gustavo Carneiro, Jacinto Nascimento, and Andrew P Bradley. Unregistered multiview mammogram analysis with pre-trained deep learning models. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pp. 652–660. Springer, 2015.

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

Alexey Dosovitskiy, Jost Tobias Springenberg, Martin Riedmiller, and Thomas Brox. Discriminative unsupervised feature learning with convolutional neural networks. In *Advances in Neural Information Processing Systems*, pp. 766–774, 2014.

Ian Goodfellow, Nicolas Papernot, and Patrick McDaniel. cleverhans v0. 1: an adversarial machine learning library. *arXiv preprint arXiv:1610.00768*, 2016.

Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *arXiv preprint arXiv:1512.03385*, 2015.

Chen Huang, Chen Change Loy, and Xiaoou Tang. Unsupervised learning of discriminative attributes and visual representations. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5175–5184, 2016.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pp. 1097–1105, 2012.

Yann LeCun, Koray Kavukcuoglu, Clément Farabet, et al. Convolutional networks and applications in vision. In *ISCAS*, pp. 253–256, 2010.

Min Lin, Qiang Chen, and Shuicheng Yan. Network in network. *arXiv preprint arXiv:1312.4400*, 2013.

Stéphane Mallat. Group invariant scattering. *Communications on Pure and Applied Mathematics*, 65(10):1331–1398, 2012.

Stéphane Mallat. Understanding deep convolutional networks. *Phil. Trans. R. Soc. A*, 374(2065): 20150203, 2016.

Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. Deepfool: a simple and accurate method to fool deep neural networks. *arXiv preprint arXiv:1511.04599*, 2015.

Edouard Oyallon and Stéphane Mallat. Deep roto-translation scattering for object classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2865–2873, 2015.

Florent Perronnin and Diane Larlus. Fisher vectors meet neural networks: A hybrid classification architecture. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3743–3752, 2015.

James Philbin, Ondrej Chum, Michael Isard, Josef Sivic, and Andrew Zisserman. Object retrieval with large vocabularies and fast spatial matching. In *2007 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1–8. IEEE, 2007.

Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. *arXiv preprint arXiv:1606.03498*, 2016.

Laurent Sifre and Stéphane Mallat. Rotation, scaling and deformation invariant scattering for texture discrimination. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1233–1240, 2013.

David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, and Martin Riedmiller. Striving for simplicity: The all convolutional net. *arXiv preprint arXiv:1412.6806*, 2014.

Rupesh K Srivastava, Klaus Greff, and Jürgen Schmidhuber. Training very deep networks. In *Advances in neural information processing systems*, pp. 2377–2385, 2015.

Kevin Swersky, Jasper Snoek, and Ryan P Adams. Multi-task bayesian optimization. In *Advances in neural information processing systems*, pp. 2004–2012, 2013.

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. *arXiv preprint arXiv:1605.07146*, 2016.

Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *European Conference on Computer Vision*, pp. 818–833. Springer, 2014.