THÈSE DE DOCTORAT

de l'Université de recherche Paris Sciences et Lettres
PSL Research University

Préparée à l'Ecole Normale Supérieure

Analyzing and Introducing Structures in Deep Convolutional Neural Networks

Ecole doctorale n°386
SCIENCES MATHEMATIQUES DE PARIS CENTRE

Spécialité INFORMATIQUE

COMPOSITION DU JURY :

M. PARAGIOS Nikos
CentraleSupélec, Rapporteur

M. PERRONIN Florent
Naver Labs, Rapporteur

M. CORD Matthieu
LIP6 / UPMC, Président du jury

M. LAPTEV Ivan
INRIA / ENS, Membre du jury

M. PEREZ Patrick
Technicolor, Membre du jury

Soutenue par Edouard **Oyallon**
le 6 Octobre 2017

Dirigée par **Stéphane Mallat**

ENS PSL RESEARCH UNIVERSITY PARIS

Analyzing and Introducing Structures in Deep Convolutional Neural Networks

Edouard Oyallon

Résumé

Cette thèse étudie des propriétés empiriques des réseaux de neurones convolutionnels profonds, et en particulier de la transformée en Scattering. En effet, l'analyse théorique de ces derniers est difficile et représente jusqu'à ce jour un défi: les couches successives de neurones ont la capacité de réaliser des opérations complexes, dont la nature est encore inconnue, via des algorithmes d'apprentissages dont les garanties de convergences ne sont pas bien comprises. Pourtant, ces réseaux de neurones sont de formidables outils pour s'attaquer à une grande variété de tâches difficiles telles la classification d'images, ou plus simplement effectuer des prédictions. La transformée de Scattering est un opérateur mathématique, non-linéaire dont les spécifications sont inspirées par les réseaux convolutionnels. Dans ce travail, elle est appliquée sur des images naturelles et obtient des résultats compétitifs avec les architectures non-supervisées. En placant un réseaux de neurones convolutifs supervisés à la suite du Scattering, on obtient des performances compétitives sur ImageNet2012, qui est le plus grand jeu de données d'images étiquetées accessible aux chercheurs. Cela nécessite d'implémenter un algorithme efficace sur carte graphique. Dans un second temps, cette thèse s'intéresse aux propriétés des couches à différentes profondeurs. On montre qu'un phénomène de réduction de dimensionalité progressif a lieu et on s'intéresse aux propriétés de classifications supervisées lorsqu'on varie des hyper paramètres de ces réseaux. Finalement, on introduit une nouvelle classe de réseaux convolutifs, dont les opérateurs sont structurés par des groupes de symétries du problème de classification.

Abstract

This thesis studies empirical properties of deep convolutional neural networks, and in particular the Scattering Transform. Indeed, the theoretical analysis of the latter is hard and until now remains a challenge: successive layers of neurons have the ability to produce complex computations, whose nature is still unknown, thanks to learning algorithms whose convergence guarantees are not well understood. However, those neural networks are outstanding tools to tackle a wide variety of difficult tasks, like image classification or more formally statistical prediction. The Scattering Transform is a non-linear mathematical operator whose properties are inspired by convolutional networks. In this work, we apply it to natural images, and obtain competitive accuracies with unsupervised architectures. Cascading a supervised neural networks after the Scattering permits to compete on ImageNet2012, which is the largest dataset of labeled images available. An efficient GPU implementation is provided. Then, this thesis focuses on the properties of layers of neurons at various depths. We show that a progressive dimensionality reduction occurs and we study the numerical properties of the supervised classification when we vary the hyper parameters of the network. Finally, we introduce a new class of convolutional networks, whose linear operators are structured by the symmetry groups of the classification task.

Acknowledgement

This work is supported by the ERC InvariantClass grant 320959.

Remerciements

Tout d'abord merci à Stéphane pour m'avoir beaucoup appris et consacré beaucoup de temps. J'ai passé d'intenses moments de réflexions, et ton regard sur tous les sujets que nous avons étudiés est exceptionnellement profond, et m'inspirera longtemps.

Je tiens à remercier tous les brillants (actuels, anciens) membres de l'équipe de Stéphane avec qui j'ai pu parfois discuté, comme Vincent, Joakim, Joan, Laurent, Irène, Tomàs, Matthew, Guy, Tomàs, Louis, John, Sira, Carmine, Sixhin, Gilles, Alberto, Ravi, Ivan, Michaël, Mathieu. Je vous souhaite le meilleur parcours possible. Merci à Damien, Bogdan, Gabriel, Frederick, Maxime, Sergey, Jörn pour les collaborations ! Enfin merci à Joëlle, Lise-Marie, Valérie et Sophie, ainsi que le SPI qui sont d'une efficacité redoutable.

Puis de remercier mes camarades de Rennes, avec qui j'ai passé de bons moments en licence. Puis les amis parisiens. Comme Pauline, Jonathan, Thomas, Gisela, Agnès, Adrien. Merci aux slovaques. (Michal^2 & Anna) Merci à vous d'être des personnes formidables.

Je tiens à remercier aussi chaleureusement Maxime & Daria (les meilleurs voisins), Paul-Darius, Rafael, Grégoire & Misko, Fred, Eugene, Mikael, Bourreau pour avoir contribué très négativement à l'écriture de ce manuscrit.

Je sais que j'en oublie, et je m'excuse :-)

Merci à ma famille (ma sœur, cousines, cousins, tantes, oncles, mes grandparents, mes parents) pour son soutien, qui est très important pour moi, et de ne pas avoir trop râlé pendant ces 4 ans de ne pas trop me voir.

Et puis merci à toi, Hélène.

Contents

List of Figures

Chapter 1

Introduction

Designing a robot that can talk, hear, see or understand is a big sci-fi dream. Indeed, there are no clear instructions or source codes to create it: this requires to process efficiently a large flow of data that are often complex, and the methods to process this information are as well unclear because there exists no clear mathematical framework that adresses these tasks. However, recent engineering efforts attack the question of artifical intelligence and obtain excellent results, with techniques like Convolutional Neural Networks. This thesis studies the latters.

1.1 Introduction to Convolutional Neural Networks

1.1.1 Neural networks: supervised and generic algorithms

Neural Networks exist since at least 1958 [92, 93], yet only recently have they achieved outstanding performances in a wide range of large-scale and difficult

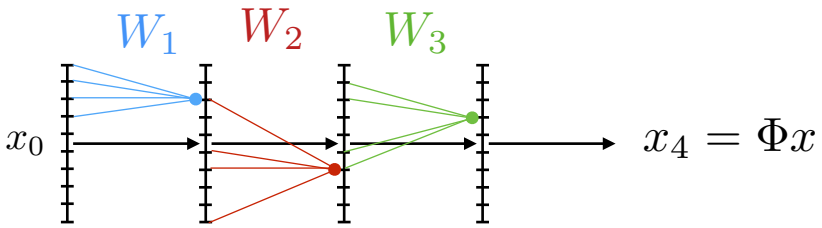


Figure 1.0.1: A schematic representation of a Neural Network with $J = 4$. Each layer is computed via the linear operators $W_j, j \leq J$, the non-linearity is omitted.

tasks for machines, like image classification [56] of huge datasets. In a machine learning context, a neuron is an elementary unit [62], which can perform basic operations and is connected to other neurons. Typical artifical neurons activations result from linear combination of input activations, followed by a non-linearity. Then, a Neural Network is a stack of successive layers of neurons that are connected to each others, in order to produce an output activation. An initial signal is propagated through the network, which leads to successive activation of neurons, until this final layer. Often, the neurons are optimized to solve a specific task. The input signal can be for example an audio wave, a text, an image, a chess-board picture, ... or even unstructured data.

Among the tasks that are solved by Neural Networks, there are classifying signals [56, 46], transfering artistic style [37], generating images [41], detecting signals [81] or playing a strategy game [76, 103]. The neural method is also referred as "deep learning" [58] because it often consists in learning a deep cascade of neurons. The above-mentionned tasks are difficult tasks because for example some of them are not well defined; to illustrate this, let us take two applications that are very different. The style transfer application [37] is a first example: one must transfer the artistic style from an initial image to a target image. Here, the fitting measure is purely perceptual. In other cases, the exact goal of the task is computationally intractable, for the game of Go. Indeed, the number of possible outcomes of the game is about 10^{170} which results in higher complexity by a googol than the game of chess [103]. Until recently, it was thought that designing a winning strategy for this type of very complex game would be exclusively a free-of-machine task, because humans can exploit very complex regularities that they learn through the practice of playing, and refine their strategies until they win. In 2015, the AlphaGo [103] of Google has contradicted this affirmation by beating a master of the Game of Go. Larger datasets and computational power are partially an explanation of this success [54].

Classifying images or playing the game of Go are also different tasks by nature, which until recently required a lot of knowledge specific to those different domains. On the contrary, Neural Network methods tend to be generic [35] and can be deployed in very different contexts: obtaining a sucessful Neural Network Φ requires simply to accumulate a large collection of data, with a limited preprocessing procedure.

For $d \in \mathbb{N}$, let us consider a supervised problem, where $x^1, ..., x^N \in \mathbb{R}^d$, are some training samples, such that for each x^n the desired output or label is given by $f(x^n)$. Often, $(x^n, f(x^n))$ follow an unknown distribution of probability $\mathbb{P}$. For example, x could be an image with a label $f(x)$ like "cat". In our case, we learn a parametric representation $\hat{f}$ from the data $(x^n, f(x^n))_{n \leq N}$, and the objective is to build $\hat{f}$ which is an estimator (i.e. a statistic that depends on the data) that best fits $f(x)$ from x.

We assume that $\hat{f}(x)$ can be factorized as $f_0(\Phi x)$, which is detailed in [71], where Φ is a deep neural network and f_0 a parameter-free function that assigns a class. For example, in the case of a binary classification, f_0 can be a sign

function, which assigns one of the elements of $\{-1, 1\}$ to the different classes.

Our objective is the following supervised task [11]: estimating Φ which minimizes the risk

$$\mathbb{P}(f(x) \neq \hat{f}(x)). \tag{1.1.1}$$

Unfortunately, this quantity can only be approximated through a validation set, via an empirical estimator of the risk, which is given by:

$$\frac{1}{N} \sum_{i=1}^{N} 1_{f(x^i) \neq \hat{f}(x^i)} \tag{1.1.2}$$

We consider the representations that are constituted by Neural Networks [58]. Figure 1.0.1 represents a typical neural network architecture. Each activation x_j of successive neurons is obtained by a linear combination W_j of past activations x_{j-1} followed by a non-linear operator ρ:

$$x_j = \rho W_j x_{j-1} \tag{1.1.3}$$

If the cascade is of depth J, then the final layer x_J corresponds to the representation Φx. Training a neural network is equivalent to finding the weights W_j, for all $j \in \{1, ..., J\}$, via estimation or optimization procedures based on a training dataset, and the outcome of it is to minimize the risk of Eq. (1.1.2) because the risk of Eq. (1.1.1) is an intractable value. In particular, the supervision permits to adapt Φ to the specific biases of the data $(x^n)_{n \leq N}$.

However the rules and mechanisms that permit to obtain the weights W_j, for all $j \in \{1, ..., J\}$, are not specified explicitly during the training procedure. It means there is no indication at training time on the nature of the operations that are performed in the inner layers to build a representation Φ that leads to a good performance, and thus that some of the properties exploited by neural networks remain an ubiquitous enigma to researchers [112, 123].

1.1.2 Supervised image classification: a high-dimensional task

In this subsection, we explain why classifying high dimensional signals is difficult. The high-dimensional nature of x implies that f is hard to estimate. Indeed, typical images are represented by a vector of length $d = 10^5$. For illustration purpose, let us consider a regression problem on $[0, 1]^d$, where f is real valued. A reasonable regularity constraint in main machine learning tasks is that f is L-lipschitz. It implies that for two x, x' satisfying $\|x - x'\| \leq \frac{\epsilon}{L}$, one has:

$$|f(x) - f(x')| \leq \epsilon \tag{1.1.4}$$

Approximating f is equivalent to generate a partition of $[0, 1]^d$, with balls of radius $\frac{\epsilon}{L}$. However, in this case the partition has a covering number of about

$(\frac{\epsilon}{L})^d$ ball centers, which grows exponentially, and typical datasets do not have enough samples available. For example, with only $d = 30, \epsilon = 0.5$, at least 10^9 samples are necessary which is a larger amount than the size of typical datasets. There is a *curse of dimensionality*.

In this thesis, we generally study natural image classification tasks, which implies that the signals of the dataset have a lot of (complex) structures, for which mathematical models are often limited to geometric consideration. As the output of a neural network is a class, Φ performs a necessary dimensionality reduction, and it must contract the space along non-informative direction for the classification task. It means that for a given $t = f(x')$, the representation Φ must contract the space along the level sets of f, which are:

$$\Omega_t = \{x, f(x) = t\}. \tag{1.1.5}$$

The non-informative directions correspond to variabilities $\mathcal{L}$ that preserve the class of an image,

$$\forall x, f(\mathcal{L}x) = f(x). \tag{1.1.6}$$

We refer to $\mathcal{L}$ as a *symmetry of the classification task* [71, 16]. In particular Ω_t is preserved by $\mathcal{L}$ if and only if it is a symmetry. An example is given by translations by a

$$\forall x, \forall u, \mathcal{L}_a x(u) = x(u - a). \tag{1.1.7}$$

Here, translating an image labeled with "cat" preserves its class (as the cat is still visible), which means that:

$$f(\mathcal{L}_a x) = f(x) \tag{1.1.8}$$

Other examples of Euclidean transformations that preserve the class are given by small rotations or small deformations of an image. Similarly, the change of colors of a background is as well non informative in a face recognition task yet this is not in general a transformation of the Euclidean group. Thus, $f_0 \circ \Phi$ must necessary eliminate those variabilities to classify images which mathematically means that

$$f_0(\Phi\mathcal{L}x) = f_0(\Phi x) \tag{1.1.9}$$

A way to build an invariant to translation is to apply an averaging, indeed if $x' = \mathcal{L}_a x$, then

$$\int \Phi\mathcal{L}_b x' db = \int \Phi\mathcal{L}_{a+b} x db = \int \Phi\mathcal{L}_b x db \tag{1.1.10}$$

An invariant is thus created. Unfortunately, this loses important informations of an image: two very different signals can have the same averaging, because for example, in the linear case, an averaging corresponds to a projection with codimension 1.

1.1.3 Breaking the curse of dimensionality with CNNs

Convolutional Neural Networks (CNNs) in vision produce low-dimensional representations that give outstanding numerical results in classification [46, 56], detection [80], scene understanding [114], etc. Yet they are also successfully used in strategy games [76, 103], audio classification [90], text generation [110], which implies this technique is generic, and thus of extreme interest. Not only the technique is generic, yet it is possible to apply successfully an architecture on a dataset different from the training set [122]. It implies that such networks have captured generic properties of images that are not linked to the specific bias of the data. The most famous Deep Convolutional Network is the AlexNet [56], introduced by Alex Krizhevsky, which has set up the trend on deep learning, and permitted the revival of neural networks in 2012.

We explain why Convolutional Neural Network architectures introduce necessary structure in the context of image classification. For monotically-increasing continuous non-linearities, the universal approximation theorem [29] states that Neural Networks are universal approximators of continuous function, with only two layers. However, the number of neurons that is necessary grows exponentially with the dimension [3]: there is again a curse of dimensionality. Besides, the use of "deep" networks with increasingly deeper [46] layers suggests that more than 2 layers are necessary to learn from the data. Incorporating structure into deep network in order to shed light on these phenomena is thus necessary to understand how to design their architecture.

We now discuss the notion of Convolutional Neural Networks that were introduced by LeCun [59], and explain why they outperform Neural Networks that do not have constraints on linear operators. For example, in the case of images a huge engineering effort has been done to design appropriate neural network architectures. The spatial index plays an important role in images. Sharing the weights between different spatial positions permits to reduce drastically the number of parameters of a CNN, thus it reduces the learning complexity of the supervised tasks.

However the number of parameters remains large. Estimating a Φ via the training set, which generalizes, is hard as well. For example, for typical neural networks like the famed AlexNet [56], the number of parameters p of Φ is about $p \approx 60M$ when the number of samples is about $N \approx 1M$, thus the model is prone to overfitting, which means that the network might not generalize well and that the risk of Eq. (1.1.1) has not been minimized because of a large estimation error. The reason why they tackle this difficulty is still unclear [123].

In signal processing, the corresponding linear operator that shares weight location is given by convolutions from a filter k. This latter represents the weights of the neurons. Convolutional operators commute with translation, and are mathematically defined as:

$$x \star k(u) = \int x(t)k(u-t)dt \tag{1.1.11}$$

and thus for a spatial translation $\mathcal{L}_a$:

$$(\mathcal{L}_a x) \star k = \mathcal{L}_a (x \star k) \tag{1.1.12}$$

In the case of CNNs [59, 60], the linear operator W takes as an input and output some feature layers $x(u, \lambda)$, where λ is a channel index:

$$Wx(u, \lambda') = \sum_{\lambda} x(., \lambda) \star k_{\lambda, \lambda'} \tag{1.1.13}$$

and $k_{\lambda,\lambda'}$ is the kernel associated to W. The non-linearity ρ is often [56] chosen as ReLU, e.g. $\rho(x) = \max(0, x)$. Typical hyper parameters have been widely optimized. For example, a filter has a small support for two reasons: first, it reduces the number of parameters to learn, secondly, any filters of arbitrary size can be decomposed in cascade of small filters [68]. Indeed, many architectures incorporate an affine bias combined with a ReLU, which means that one can write:

$$\rho(x(u, \lambda)) = \max(0, x(u, \lambda) - \rho_\lambda) \tag{1.1.14}$$

where ρ_λ is a weight vector learned. By choosing large values for ρ_λ the CNNs could literally learn to drop the effects of the non-linearity: thus a CNN can potentially implement any convolution with arbitrary size if this is necessary, which has the benefit to slightly reduce the number of parameters necessary to implement this operation. While convolutions incorporate structures, they do not tackle the question of the necessary depth, the size of the layers or other architecture considerations.

Cascades of convolutional operators permit to build representations that are covariant with the action of translation. It means that:

$$\Phi \mathcal{L}_a = \mathcal{L}_a \Phi \tag{1.1.15}$$

and the averaging of the Eq. (1.1.10) can be written as a linear averaging A via the spatial index u:

$$\int \Phi \mathcal{L}_b x db = \int \mathcal{L}_b \Phi x db \triangleq A \Phi x \tag{1.1.16}$$

One important question is to understand the nature of the invariants that are built. Translation invariance is a necessary property that is achieved by state-of-the-art neural networks, and the previous equation shows that the structure of the CNN plays an important role.

Data augmentation via translations can be added in order to achieve a better invariance to translation. Yet translations are not the only source of variability. This raises the question to understand whether the non geometric transformations invariants can be specified by similar geometrical considerations and methods.

In summary, neural networks are generic techniques that can drastically reduce the dimensionality of a representation, in order to solve challenging high-dimensional and large scale tasks. However the mathematical foundations of the strategy they use to do so remains absolutely unclear.

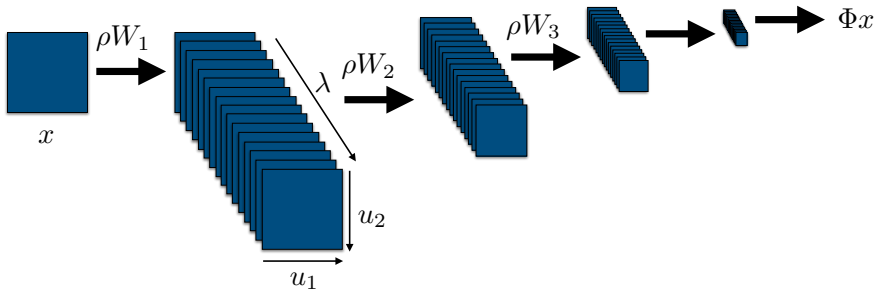


Figure 1.1.1: A typical CNN architecture. The signal is propagated through cascades of convolutions W_j, a point-wise non-linearity ρ and progressive down-sampling. Each layer is indexed by the spatial positions (u_1, u_2) and the channel index λ.

1.1.4 Introducing and discovering structure in CNNs

In this thesis, we introduce and analyze structures in CNNs. It means that each chapter aims at either incorporating structures, or understanding the self organization of the representations that are built by the CNNs. We try to understand the design of the architecture of the CNNs: for example, among other questions: which depth is required? How should the non-linearity be chosen?

While the objective at the training time is clear, which is to separate the samples of different classes, what are the inner mechanisms that permit it to generalize? To do so, we propose methods that permit to discover empirical properties of CNNs.

We apply the scattering transform introduced in [69, 70] which is a deep non-blackbox representation that leads to promising numerical results.

In a nutshell, among comprehensive software implementations that we provide online [1], some of our main contributions are the following:

1. Studying the invariants learned in CNNs and in particular, geometric transformations [86, 85, 84, 82],
2. Performing empirical analysis of CNNs trained on complex images datasets [83],
3. Specifying CNN operators with symmetries of the classification task [50].

1.2 Nature of the invariances learned

As said above, there are two kinds of variabilities: euclidean transformations and others. We raise the question to understand to which extent invariants

[1] https://github.com/edouardoyallon/

to geometric transformations are incorporated in the deep pipelines. To this end, we study the Scattering Transform which is introduced in [69, 70], and corresponds to a representation fully specified by geometric considerations. We apply it to complex image classification benchmarks, in order to understand to which extent the knowledge of Euclidean groups is enough to discriminate complex images.

1.2.1 Priors compete with unsupervised representations

The Scattering Transform is introduced by [70]. It is the only neural network whose design and weights are fully guided by mathematical foundations, and which has been shown useful when the classification task only aims to reduce several geometric variabilities like translations. It is a predefined representation that involves limited learning. In vision, its architecture consists in a cascade of two 2D complex wavelet transforms [68, 13], that perform convolutions along euclidean groups. The ultimate goal of a Scattering Transform is to build some invariance w.r.t. those groups, and in particular translation: this latter is a natural candidate of variability to remove for image classification.

Contrary to fully supervisedly learned CNNs, the operators of a Scattering Network can be fully analyzed. In particular, the connection, non linearity and linear combinations between the neurons of the Scattering layers are specified by the wavelets that are used: it implies that the indexes of the layer of a Scattering Network are organized. We review its properties in Chapter 2.

A natural question that arises in the context of classification with a Scattering representation is to understand to which extent the invariants from non-Euclidean transformations are necessary for image classification. Indeed, in an ideal case, an appropriate geometrical representation of an image would reduce the variance of the classification task to a pure sample variance problem.

This is the case for example with classification of small handwritten digits [15]: the main variabilities are due to small deformations and small translations. As they are reduced by the Scattering Transform, a classifier such as a Gaussian SVM [113] completely solves this digit recognition task.

In Chapter 3, we show that with a purely roto-translation representation that relies on a limited amount of learning, it is possible to obtain significative numerical performances on natural image classification. Furthermore, the Scattering representation is generic (i.e. is not adapted to very different datasets) and the numerical accuracies that we obtain are competitive with unsupervised algorithms for building a representation for classification. It indicates that current unsupervised representations do not capture more complex invariants than Euclidean ones.

This is a problem since many efforts are done to use the large unlabeled datasets that are available. Indeed, annotating data is expensive. Thus, we suggest that in the case of images that are signals with a lot of geometry, incorporating this geometry might lead to similar performances to representations that are built upon a very large unlabeled dataset. However, there exists still an important gap between unsupervised and supervised representations that

we discuss in the next Subsection: how can we integrate supervision in the Scattering, in order to fill in this gap?

1.2.2 Filling the gap by adding supervision

Using a discriminative descriptor in a supervised pipeline permits to incorporate more predefined complex structure than with the raw signal. We could incorporate some learning that will be adapted to the wavelets in the Scattering Transform. Although this direction has not been explored in this thesis, this property would mean that the operators of the Scattering Transform must be adapted to a specific dataset.

Instead, our contribution is to show a stronger property: we demonstrate that the Scattering Transform is an appropriate predefined and fixed initialization of the first layers, in Chapter 4, for natural images. This is coherent with the transfer learning properties of CNNs as well [122], that shows that the earlier layers of CNNs are related to geometric considerations. We refer to those networks as Hybrid Networks. It implies that the generic geometrical invariants that are encoded by the Scattering Transform can be exploited by a supervised CNNs cascaded on top of it and propagate other sources of regularity which are important for the classification.

Thus the scattering does not lose discriminative information that is used by the supervised CNNs. In fact, we demonstrate that incorporating the geometrical prior improves the classification algorithms in the case where a limited amount of data is available. This is coherent, because it permits to avoid learning those invariants and to reduce the geometric variance of the supervised task.

A contribution of Chapter 4 is to show that since the input of the Hybrid CNN is covariant with rotation, the CNN learns to build a linear invariant to the rotation: an averaging along the rotation group. While this is not surprising, it was not specified during the optimization. Obtaining insights in the inner layers is complicated to do, but thanks to the Scattering Transform, it is possible to interpret a few of them: the depth of the network is smaller, then we understand the first learned layers and the non learned layers: this is a promising direction to understand the remaining layers.

1.3 Empirical analysis of Neural Networks

Here, we report the empirical properties that we obtained in this thesis. Several works suggest that a progressive specialization of the layers occurs. It means for example that the different atoms of the operators of a CNN answer to different stimuli. First they are more sensitive to geometrical patterns, then in the next layers, they are more task-specific.

However, more analysis remains to be done to understand more precisely the nature of the operation performed at each layer of a CNN, and identifying or estimating the variabilities that are not related to geometrical transformations in a classification problem is hard.

Several models rely on a manifold hypothesis, which is a simplification that can not permit to modelize the whole diversity and the hardness of the classification task. Indeed, it could imply that the variability of images are related to a manifold structure. However, no manifold learning techniques have been shown to be applicable (which suggests the dimension of the data is locally low) in the case of image classification because this task is high-dimensional. Let us give an example, the small deformations by $\tau \in \mathcal{C}^\infty, \|\nabla\tau\| < 1$, which act via

$$\mathcal{L}_\tau x \triangleq x(u - \tau(u)) \tag{1.3.1}$$

They are high-dimensional objects that are not parametric, and thus they can be potentially hard to estimate. Besides, there is a large number of variabilities that can potentially have a manifold structure in several cases (occlusions, light change, color changes, euclidean and projective variabilities, ...) and they fail to lay in a low dimensional structure thus their estimation is delicate.

For example, the work of [2] suggests that a deep network is able to build a tangent space of variabilities. This is also proved for the Scattering Transform [70]. It implies the variability has been locally linearized, which means that:

$$\Phi\mathcal{L}x \approx \Phi x + \partial\Phi(x).\mathcal{L} \tag{1.3.2}$$

Here, a linear projector Π can build an invariant to this variability by removing the dependency in $\mathcal{L}$, if:

$$\text{span}(\partial\Phi(x)) \subset \ker\Pi. \tag{1.3.3}$$

1.3.1 Designing generic and simplified architectures

We explain how to simplify CNN architectures. They use a wide number of different modules [126, 119, 116, 107, 63, 60, 49, 10] that are cascaded through the architecture. Yet, many of them have a limited impact on the classification while adding a lot of non-linear effects, and often improving only marginally a benchmark. They are often hard to analyze because many variations of their combinations are possible and the optimal setting is never clear.

Very deep networks are also harder to analyze than shallow ones. Indeed, each extra-layer which is learned through back-propagation and that can be removed is a layer less to analyze. They are also harder to train and need some specific hints to be optimized, like the "shortcut connections" with the ResNet [46] that permit to train very deep and state-of-the-art networks.

In Chapter 5, we propose a generic class of state-of-the-art deep networks that depends only on two hyper parameters: the non-linearity and the width of the network. We study its classification performances w.r.t. those parameters.

1.3.2 Progressive properties of CNNs

The work of [122] suggests that CNN representations are progressively linearly separable. In Chapter 5, our contribution is to refine the description of this

progressive separation.

Since the cascade of CNNs operators is non-linear, we must refine the usual linear tools that are used. While the final representation of a CNN is built by the cascade, we propose to study the representations layer wise, for each x_j. We derive some basic properties that indicate a progressive reduction of dimensionality: as j increases, the representations have a lower variance along the principal axis and the intra-class distances are reduced.

Let us recall that a nearest neighbor is a classifier that assigns the class of its nearest neighbors. Its success relies on the choice of the metric that is used. The classification boundary corresponds to the sets that delimit the different classes of a classification task.

We refine the previous analysis by introducing a notion of *local support vectors*, in order to describe the classification boundary of CNNs at different depths. They correspond to sample representations of the training set that are mis-classified by a nearest neighbor algorithm, and thus locally support the boundary of classification. We show that their number is progressively reduced, which indicates a phenomenon of contraction. Besides, we introduce a notion of complexity of the classification boundary at these points: it shows that the classification boundary is getting lower dimensional with depth, which indicates the representations of the different classes are more easily to be separated with depth.

This explains why a nearest neighbor classifier accuracy progressively improves with depth: the linear metric is progressively more meaningful.

1.4 Imposing symmetries of Neural Networks

We describe a novel class of deep neural networks inspired by the hypothesis of the existence of a Lie group which aims at approximating the symmetry group of the classification task [71, 16, 38, 25].

1.4.1 Parallel transport along symmetry groups

We explain the notion of parallel transport along a symmetry group, which is also described by [71]. As said before, a symmetry in a supervised classification task is an invertible operator $\mathcal{L}$ that preserves the class:

$$\forall x, f(\mathcal{L}x) = f(x) \tag{1.4.1}$$

We recall that the symmetries of f are a group. Indeed, if $\mathcal{L}, \mathcal{L}'$ are two invertible operators, $\mathcal{L}^{-1}\mathcal{L}'$ is invertible as well and:

$$\forall x, f(\mathcal{L}^{-1}\mathcal{L}'x) = f(\mathcal{L}'x) = f(x) \tag{1.4.2}$$

However, this group can be infinitely dimensional, and by definition non-parametric, thus difficult to estimate. [71] proposes to progressively approximate

it with finite dimensional groups of symmetries that correspond to Lie groups. To this end, we consider each layer at depth j and write:

$$\Phi(x) = \hat{\Phi}_j(x_j), \text{ where } \hat{\Phi}_j = \rho W_J \rho ... \rho W_j \tag{1.4.3}$$

A parallel transport is defined [71] as an action g_j along the coordinate of x_j:

$$\forall v, g_j.x_j(v) \triangleq x_j(g_j.v) \tag{1.4.4}$$

Translations are an example of such actions. For each $\hat{\Phi}_j$, [71] specifies its linear group of symmetry, which is given by the linear and invertible operators g_j such that:

$$\forall x, \hat{\Phi}_j(g_j.x) = \hat{\Phi}_j(x) \tag{1.4.5}$$

where the inputs x to $\hat{\Phi}_j$ do not necessarily correspond to the representation at depth j. Each g_j belongs to a Lie group G_j and this group of symmetry is increasing, in the sense that $G_j \subset G_{j+1}$. How can we exploit those groups of symmetries? An answer is proposed in [71].

1.4.2 Multiscale Hierarchical CNNs

The objective of a good representation is to build invariances w.r.t. symmetry groups while preserving discriminating informations. At each depth j, local invariants can be obtained by performing operations along the orbit $\{g_j.x_j\}_{g_j \in G_j}$. To do so, [71, 16, 38] propose to build operators ρW_j such that applying $g_j \in G_j$ to x_j corresponds to the action of a symmetry $g_{j+1} \in G_{j+1}$ on x_{j+1}, e.g.:

$$\rho W_j(g_j.x_j) = g_{j+1}.(\rho W_j x_j) \tag{1.4.6}$$

It implies that if $\hat{\Phi}_j(g_j.x_j) = \hat{\Phi}_j(x_j)$ then $\hat{\Phi}_{j+1}(g_{j+1}.x_{j+1}) = \hat{\Phi}_{j+1}(x_{j+1})$: invariants are propagated through this operator. A cascade of those operators is referred as Multiscale Hierarchical CNNs.

However, a naive implementation of the W_j requires a number of parameters that grows exponentially with depth, which is thus hard to engineer. Hierarchical Attribute CNNs provides a solution to this issue.

1.4.3 Hierarchical Attribute CNNs

A contribution of the Chapter 6 is to prove that the operators W_j can be implemented and trained with symmetry groups that contain Euclidean translations $G_j = \mathbb{R}^{d_j}, d_j \in \mathbb{N}^*$, and thus are linear convolutions. We thus have $d_j \leq d_{j+1}$. It is less general than the Multiscale Hierarchical CNNs because non-commutative groups convolutions can not be embedded so, as in the case of the non-separable Roto-translation Scattering [101].

We construct each layer x_j such that they depend upon a set of indexes structured by translations along G_j to which we refer as *attributes*. It defines

Hierarchical Attribute CNNs which are a subclass of the Multiscale Hierarchical CNNs introduced in [71]. The operator W_j performs convolutions along the attributes of G_j and build invariants to these via linear averaging. The final output Φx is linearly averaged along all those directions of variabilitiy, and thus completely invariant to the translations G_J. It requires to use high-dimensional convolutions and we thus propose an efficient implementation of it. The averaging is also important because it permits to control the dimensionality of the network.

Our work aims at mapping non-linear symmetries of Φ into the symmetries of Φ_j, that are linear translations. It implies that the former have been linearized. Without structuring the network, it is difficult to observe symmetries because they can potentially correspond to high-dimensional operators. For instance, in the case of a linear translation of signals of length N, the corresponding matrix has N^2 coefficients. Directly specifying the translations permit to tackle the necessity to estimate the symetries because they are explicitly incorporated. Can we analyze those symmetries? When training a neural network with these constraints, we drastically reduce the number of parameters necessary to parametrize the W_j.

We study the properties of those attributes and translations on natural images, and explain the limitation of our method.

Chapter 2

Background

This chapter gives a short introduction to Convolutional Neural Networks (CNNs) [59] and Scattering Transform [70] that we use systematically in each chapter of this thesis. We first describe them in term of architectures and mathematical properties. CNNs and in particular Scattering Transforms can both be used as a strong baseline for image classification, albeit their fundations might seem opposed. For example, a CNN is fully learned via supervision when a Scattering Transform depends upon a limited number of hyper parameters. Few mathematical results are available for CNNs, contrary to a Scattering Transform. However, a Scattering Transform can be interpreted as a specific class of CNN, and it was introduced so in [69].

2.1 Convolutional Neural Networks Review

convolutional neural networks (CNNs) were introduced at the end of the 80s by LeCun [59]. Now, they are standard to *solve* vision tasks [58, 56, 60]. They are a specific class of neural networks, where almost each operation performed by the network is covariant with translations. A neural network is the cascade of linear and non-linear operators, or more informally, a hiearchical stack of neurons, fully learned via supervision. They are inspired from neuroscience, and how the brain works [4]. However, these interpretations remain unclear in many cases [36].

From a technical point of view, Neural Networks had a revival of interest since the arrival of GPUs and large datasets. Indeed, those models have a large amount of parameters, and are extremely prone to overfitting. Their training is computationally heavy, as they rely on a large number of trainable parameters. GPUs permit to speed-up linear algebra routines and are a core element of CNNs. Their use in machine learning is recent, and almost corresponds to the release of the large image dataset, ImageNet. It consists in one million large natural colored images divided into one thousand classes, that were annotated by humans. Such a dataset is necessary to train those large models. The

combinations of those two techniques is thus attractive in the industry for where a lot of data is available with high-dimensional problems that were unsolved until now, and these possibilities show that deep networks are a large-scale architecture.

While the flexibility and the possibility to train larger models is appealing to improve current benchmarks, it also makes their analysis harder. Few mathematical results explain why those networks generalize well from a training set to a validation set.

First we describe the "neurons" that arise in CNNs via Subsection 2.1.1, how to optimize the weights of this architecture in Subsection 2.1.2, and we explain the theoretical challenges that arise in Section 2.1.3.

2.1.1 Standard architectures

We summarize some pivotal definitions about standard CNNs architecture that we will widely use in the next sections. We also recall simple properties that hold for CNNs, for example they can build invariance to translation and are Lipschitz w.r.t. their input. A Neural Network is the cascade of linear operators and non-linearity. It propagates an input signal $x_0 = x \in L^2$ through the last layer x_J, where J is referred as the depth of the network. Let us consider a spatial index p. We refer to $x_j[p, \lambda]$ as the propagated layer at depth $0 \leq j \leq J$, where $\lambda \leq \Gamma_j$ is the index of the feature map. Each layer is computed from the previous layer via:

$$x_{j+1} = \mathcal{P}_j \rho_j W_j x_j \tag{2.1.1}$$

where W_j is a CNN linear operator, ρ_j a point-wise non-linearity and $\mathcal{P}_j$ is an extra operator, like a pooling operator or the identity. Let us discuss briefly the different operators of each layers.

CNN linear operator W_j

A CNN linear operator is defined as an operator of l^2 between two layers , $\forall p, \forall \lambda \leq \Lambda$:

$$W x[p, \lambda] = \sum_{\tilde{\lambda}} x \star k_{\lambda, \tilde{\lambda}}[p] \tag{2.1.2}$$

where p is a discrete spatial position index, λ the index of a feature map and $k_{\lambda, \tilde{\lambda}}$ is convolutional kernel. Usually, the convolutions are performed with a zero-padding. In this case, one can show that a linear operator is a CNN linear operator if and only if it is covariant with the action of spatial translations. Purely cascading linear operators without intermediary non-linearity would be equivalent applying one linear operator. To avoid extra non-necessary operations and to obtain a non-trivial CNN, pointwise non linearities are applied. Each of the weights $k_{\lambda, \tilde{\lambda}}$ will be learned via a gradient descent procedure described in Subsection 2.1.2.

Pointwise non-linearity

Let be $\rho^\lambda : \mathbb{R} \to \mathbb{R}$ a non-linearity. We call $\rho = (\rho^\lambda)_\lambda$ a non-uniform pointwise non-linearity, the operator given by:

$$\rho(x)[p, \lambda] = \rho^\lambda(x[p, \lambda]) \tag{2.1.3}$$

Observe that its action commutes with translation as well. If $\rho^\lambda = \rho^{\lambda'}, \forall \lambda, \lambda'$, we say that it is a pointwise non-linearity. In particular, this means that each layer is covariant with the action of translation, if the module $\mathcal{P}_j$ satisfies $\mathcal{P}_j = I$ for example.

Extra-modules $\mathcal{P}_j$

For example, the operator $\mathcal{P}_j$ can be a pooling operator. Pooling operators are given by the $\sum$ or the max pooling, that are applied on a predefined window size. They are expected to build invariance [10] w.r.t. translations. One of the issues with the latter is that it is unstable w.r.t. to translations. We write:

$$Ax[\lambda] = \sum_p x[p, \lambda] \tag{2.1.4}$$

a global average pooling operator.

Besides, $\mathcal{P}_j$ could also be a batch normalization module, which consists in an affine operator that helps to improve the conditioning of the layer at training time, by standardizing the representations at each iterations of an optimization algorithm. As this operator is completely affine at test time, we however omit it in our notations and simply mention it as an optimization trick.

As all the operators that are involved are covariant with translation, applying a global averaging A after the last convolution imposes a translation invariance on the output of the CNN and removes this variability. Similarly, assuming that each operator is non-expansive implies that the cascade will be as well non expansive.

2.1.2 Training procedure

We describe the common training procedure of CNNs, which is at the fundations of almost all the experiments performed during this thesis. Usually, CNN operators depend on a large amount of trainable parameters, that we denote as θ and non-trainable parameters, η that are computed through fixed policy rules $\mathcal{P}$. A policy rule can be for example an iterative update of the affine weights of a regularization procedure (such a dropout or a batch normalization), via a Markov Chain. For an input x, $f(x, \theta, \eta)$ is the outpout of the CNN with parameters $\theta \in \Theta$. All the operators in the cascade of the CNN are chosen weakly differentiable, with respect to their input, but as well to their parameters. In particular, it is possible to train them via a gradient descent algorithm, to minimize a loss function $\mathcal{E}$. This latter aims to measure the distance between the

label of the datasets and the output of the CNN. Typically, a loss can be the l^2 norm of the difference between an output and its desired value:

$$\mathcal{E}(x,y) = \|f(x) - y\|^2 \tag{2.1.5}$$

or the neg entropy, i.e.:

$$\mathcal{E}(x,y) = -\sum y_i \log(\frac{y_i}{f(x)_i}) \tag{2.1.6}$$

which represents the distance between the distribution of coefficients of $f(x)$ and y. Those losses being also differentiable, training a CNN on a dataset $\mathcal{X} = \{(x_1, y_1), ..., (x_K, y_K)\}$ aims to minimize:

$$\inf_{\theta\in\Theta,\eta} \frac{1}{K}\sum_{k=1}^{K} \mathcal{E}(f(x_k,\theta,\eta), y_k) = \inf_{\theta\in\Theta,\eta,\mathcal{X}} \bar{\mathcal{E}}(\theta,\eta,\mathcal{X}) \tag{2.1.7}$$

One could use a gradient descent scheme combined with the policy rules $\mathcal{P}$ at step n and with gradient step α:

$$\begin{cases} \theta^{n+1} = \theta^n - \alpha \frac{\partial \bar{\mathcal{E}}}{\partial \theta}(\theta^n, \eta^n, \mathcal{X}) \\ \eta^{n+1} = \mathcal{P}\eta^n \end{cases} \tag{2.1.8}$$

However, typically, $K \gg 1$ and using Stochastic Gradient Descent (SGD) batches of size $\mathcal{B}$ often demonstrates better performances, which leads to using the following SGD:

$$\begin{cases} \theta^{n+1} = \theta^n - \alpha \frac{1}{\mathcal{B}} \sum_{k=1}^{\mathcal{B}} \frac{\partial \mathcal{E}}{\partial \theta}(f(x_{\sigma(k)}, \theta^n, \eta^n), y_{\sigma(k)}) \\ \eta^{n+1} = \mathcal{P}\eta^n \end{cases} \tag{2.1.9}$$

$$\tag{2.1.10}$$

A batch normalization procedure is often used at the intermediary layers of f, since deep learning training operators often suffer from ill-conditionning.

2.1.3 Theoretical challenges

There is a lack of mathematical understanding of deep learning techniques. In particular, Subsection 2.1.3.1 explains the obstacles to obtain good generalization properties. Then, Subsection 2.1.3.2 develops existing results on interpreting the models. Finally, we recall they might have some unstabilities in Subsection 2.1.3.3.

2.1.3.1 Generalization properties

In this subsection, we discuss the generalization properties of a CNN, and to this end, we first recall several statistics results and explains results that suggest an appropriate framework to understand their performances is missing.

First, a CNN has usually more parameters than commons models, such as SVMs or Random Forests. Those classifiers are analogous to the fully connected layers of the AlexNet, that concentrate the mass of the parameters. It is well-known, and it has been stated recently in [123], that despite their size, they do not overfit, even without l^2 regularization or data augmentation. It means that the optimization procedure captures some source of regularities via the gradient descent that we do not have access to.

Secondly, [123] shows that the Rademacher [11] complexity results do not stand. The Rademacher complexity permits to relate a notion of expressiveness with a functional subspace. The capacity of an algorithm can then be bounded, w.r.t. to its parameters and the number of training samples, using the Rademacher complexity bounds. [123] shows that the Rademacher complexity does not decrease in the case of CNNs since they have the ability to learn perfectly the boundary classification of random labeled data.

A solution is suggested by [71, 16], which uses the concept of symmetry groups of a classification problem. It is a high-dimensional object that corresponds to all the undesirable variability that one must reduce in a classification task. A model is developped, in which a deep network would progressively build a finite dimensional approximation of this group of symmetries, and build progressive invariants. The last Chapter 6 of this thesis tries to use this concept.

2.1.3.2 Interpretability

Interpretating a CNN which obtains good performance is necessary, for example, in the case of medicine problems to understand which features permit the building of a decision and to refine the analysis performed by a doctor.

The back-propagation of the gradient through several layers makes the analysis hard, because the optimization is not constrained. It means that the choice of a weight for a given layer is fully guided by the supervision, and there is currently limited hypothesis to explain their mechanisms. Some works [71, 16] suggest that each deep operators could correspond to parallel transport along group of symmetries, yet some other works suggest CNNs detect "patterns", [118].

In order to understand the operations implemented by a "neuron" (i.e. a filter of the linear operator), [122] selects images that maximizes the activation of the output of a CNN at a given depth. However, this is extremely empirical, and [112] suggests that any linear combinations of neurons is an interesting direction as well. Yet, the nature of each operator is still unclear.

2.1.3.3 Stability

Among theoretical properties, the stability of CNNs [19, 112] is essential to apply these methods in practical cases. Stability means that a small perturbation should not have large effects on the representation built via a CNN. Therefore, for training purpose, small modifications of a dataset should not drastically alterate the quality of the learned representation. It also means that a small

perturbation of the input of a CNN which is almost not perceptible should not affect the corresponding output of the CNN.

Several works indicate that CNNs do not have this property. For example, [77, 112] show that it is possible to find an additive perturbation of an image which is visually imperceptible but drastically changes the estimated class from the CNN representation. In fact, [77] shows such a vector consistently exists and is the same across all images, which makes it universal in some sense. These effects can be avoided to a certain degree [126, 42], but no theoritical results to quantify its existance [112]. Besides additive transformations, few results exist to measure the invariance that is built w.r.t. rotations or scaling effects for example [40, 87].

2.2 Scattering Transform Review

In this section, we review the scattering transform of order 2. We will show that it is a cascade of two wavelets transforms followed by a modulus non-linearity, which is spatially averaged. The final averaging permits to build invariance to translation and to build stable invariants w.r.t. small deformations. Those two properties are of interest for image classification because they represent some of the main variabilities of natural images.

The Scattering Transform has been first defined and studied mathematically in [70, 69]. It has been successfully used in classification task of stationary textures and small digits [13, 15]: its success relies in the fact that in those problems, samples variance is small relatively to the geometric variabilities. Indeed, deformations and rigid deformations, in particular translations, are mainly responsible for the variance in those problems. In fact, it is possible to build a scattering transform along more complex variabilities on the non commutative roto-translation group [100, 101], that permits obtaining a significant improvement.

This representation has wide connection with deep learning techniques that lead to state-of-the-art results on all standard vision benchmarks. First, one can show that a scattering of scale J is actually a predefined deep network with depth J, with appropriate down-samplings. Secondly, wavelets are often observed in the first layer of a deep network [56]: in the scattering, they are systematically present. This raises the question of understanding the nature of the next layer of a deep network, in order to verify if the connection with a Scattering Transform concerns deeper layers.

The Subsection 2.2.1 starts with an input signal, and explains how to build an invariant to translation representation that still discriminates important spatial information. Next Subsection 2.2.2 recalls the properties of the Scattering Transform that permit to obtain the state of the art on the MNIST dataset and textures datasets. The last Subsection 2.2.3 argues that a Scattering Transform is actually a deep network, which leads to the name of "Scattering Network".

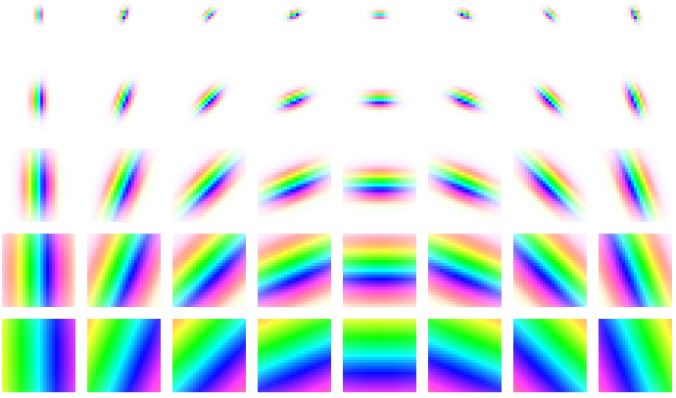


Figure 2.2.1: The 2D morlet wavelets. The amplitude is given by the contrast, and the phase by the the color. Best viewed in color.

2.2.1 Construction of the 2nd order Scattering Transform

Consider a signal $x(u)$ with finite energy, i.e. $x \in L^2(\mathbb{R}^2)$, with u the spatial position index and an integer $J \in \mathbb{N}$, which is the spatial scale of our scattering transform. We propose to explain how to build invariant representations up to translations of 2^J , but which remains discriminative. Let ϕ_J be a local averaging filter with a spatial window of scale 2^J . This filter is chosen to be dilated from a low-pass filter that builds invariance up to 2:

$$\phi_J(u) = \frac{1}{2^{2J}}\phi(\frac{u}{2^J}) \tag{2.2.1}$$

For example, ϕ can be set as:

$$\phi(u) = \frac{1}{2\pi\sigma^2}e^{-\frac{\|u\|^2}{2\sigma^2}}$$

where $\sigma = 0.85$ is a parameter that controls the bandwidth of the averaging according to Nyquist sampling [31, 68]. Interestingly, gaussian envelops permit to obtain an optimal trade-off between spatial and frequency localizations, w.r.t. the Heisenberg inequality [78, 68]. It defines a local averaging operator via:

$$A_J x(u) = x \star \phi_J(2^J u)$$

Applying the latter on the signal x we obtain the zeroth order scattering coefficient:

$$S_J^0 x(u) \triangleq A_J x(u) \tag{2.2.2}$$

This operation 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. A solution to avoid the loss of high frequency information is

provided by the use of wavelets. A wavelet [74] is an integrable and localized function in the Fourier and space domain, with zero mean. Let us consider for example a Morlet wavelet, which is given by:

$$\psi(u) = C(e^{iu^T\xi} - \kappa)e^{-\frac{u^T B u}{2\sigma^2}} \tag{2.2.3}$$

where C is a normalization factor, κ is chosen such that the wavelet has 0 averaging, $\xi = \frac{3}{4}\pi$ corresponds to the central frequency of the wavelet and B is a symmetric matrix that permits to adapt the selectivity angular frequencies of the wavelet [68]. For instance, $B = \begin{bmatrix} 1 & \\ & \xi^2 \end{bmatrix}$, $\xi = \frac{1}{2} < 1$ is used in current implementations and it implies that the level set of the amplitude of a Morlet wavelet corresponds to some ellipsoids. A family of wavelets is obtained by dilating and rotating the complex mother wavelet ψ such that:

$$\psi_{j,\theta}(u) = \frac{1}{2^{2j}}\psi(r_{-\theta}\frac{u}{2^j}) \tag{2.2.4}$$

, where $r_{-\theta}$ is the rotation by $-\theta$, and j is the scale of the wavelet. An example of this family is given by Figure 2.2.1. A given wavelet $\psi_{j,\theta}$ has thus 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, with an appropriate downsampling:

$$W_1 x(j_1, \theta_1, u) = \{x \star \psi_{j_1,\theta_1}(2^{j_1}u)\}_{j_1 \leq J, \theta_1 = 2\pi\frac{l}{L}, 1 \leq l \leq L} \tag{2.2.5}$$

Observe that j_1 and θ_1 have been discretized: the wavelet is chosen to be selective in angle and localized in Fourier. With appropriate discretization of θ_1, j_1 , $\{A_J x, W_1 x\}$ is approximatively an isometry on the set of signals with limited bandwidth, and this implies the energy of the signal is preserved: there exists $0 \leq \epsilon < 1$, for such signals x,

$$(1-\epsilon)\|x\|^2 \leq \|A_J x\|^2 + \|W_1 x\|^2 \leq \|x\|^2 \tag{2.2.6}$$

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. To achieve invariance we can not apply A_J to $W_1 x$ since it gives almost a trivial invariant, namely zero.

To tackle this issue, we apply a non-linear point-wise complex modulus to $W_1 x$, followed by an averaging A_J, which builds a non trivial invariant. Here, the mother wavelet is analytic, thus $|W_1 x|$ is regular [6] (and we prove it in Subsection 5.1.2.1) which implies that the energy in Fourier of $|W_1 x|$ is more likely to be contained in a lower frequency domain than $W_1 x$ [70].

Thus, A_J preserves more energy of $|W_1 x|$. It is possible to define $S_J^1 x \triangleq A_J|W_1|x$, which can also be written as:

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

which are the first order scattering coefficients. Again, the use of the averaging builds an invariant to translation up to 2^J. Once more, we apply a second wavelet transform W_2, with the same filters as W_1, on each channel.

This permits the recovery of the high-frequency loss due to the averaging applied to the first order, leading to:

$$S_J^2 x \triangleq A_J|W_2||W_1| \tag{2.2.7}$$

which can also be written as:

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

We only compute increasing paths, e.g. $j_1 < j_2$ because non-increasing paths have been shown to bear no energy [13]. We do not compute higher order scatterings, because their energy is negligible [13]. We call $S_J x(u)$ 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:

$$\begin{aligned} S_J x(u) &\triangleq \{S_J^0 x(u), S_J^1 x(., u), S_J^2 x(., u)\} && (2.2.8) \\ &= \{A_J x(u), A_J|W_1|x(u), A_J|W_2||W_1|x(u)\} && (2.2.9) \end{aligned}$$

With discrete signals, at computation time, intermediary downsamplings are involved to speedup the algorithm used to a compute a Scattering Transform[15, 13], and are discussed in Subsection 2.2.3, when combined with an "algorithme à trou", with a critical sampling.

In most of the application cases, an oversampling factor α is necessary to avoid aliasing, which is equivalent to increasing by a factor of α the resolution of the input signal. The filters with the smallest scales of an (approximate) analytic wavelet transform are likely to suffer from a bad localization in space [68, 97]. Consequently, we systematically apply at least an oversampling of $\alpha = 1$ [97], which means that the final spatial size of each side of the signal will be multiplied by $2^\alpha = 2$. For the sake of simplicity, we do not mention it later however we refer the reader to the related softwares to check these subtle implementation details.

Representation dimensionality

In the case of colored images, we apply independently a scattering transform to each of the 3 RGB channels of an image of size N^2, which means Sx has a size equal to $\frac{N^2}{2^{2J}} \times 3 \times \left(1 + JL + \frac{1}{2}J(J-1)L^2\right)$, and the original image is down-sampled by a factor 2^J [15].

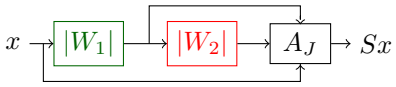


Figure 2.2.2: A scattering network. A_J concatenates the averaged signals.

2.2.2 Successful applications of the Scattering Transform

In this section, we discuss the reason why the Scattering Transform outperforms standard methods on several classic benchmarks. More precisely, we will first state several properties that the Translation Scattering Transform satisfies which are proved in [70], and then we will briefly introduce the Roto-translation Scattering Transform.

2.2.2.1 Translation Scattering

We review four theoretical properties of the Scattering Transform that are geometrical and concern stabilities of the representation w.r.t. subsets of geometrical transformations. Then, we explain how they permit to reach state-of-the-art results on the MNIST dataset.

First, the Scattering transform of order 2 is a non-expansive operator because all the operators of which it is composed, are non-expansive [70]. It means that:

$$\|S_J x - S_J y\| \leq \|x - y\| \tag{2.2.10}$$

In particular, it implies this representation is stable to additive noise, like white noise, which is common and due for example to digital camera processing.

Secondly, a Scattering Transform is almost translation invariant w.r.t. small translations $\forall a, \mathcal{L}_a.x(u) \triangleq x(u-a)$, which means:

$$\forall |a| \ll 2^J \Rightarrow S_J(\mathcal{L}_a.x) \approx S_J x \tag{2.2.11}$$

Consequently, the variability along the translation group is reduced.

Thirdly, a Scattering Transform is stable to small deformations. A small deformations $I - \tau$ is modelized as a $\mathcal{C}^\infty$ application, such that the suprema norms of τ and its Jacobian $\nabla\tau$ are small, eg:

$$\begin{cases} \|\tau\|_\infty < 1 \\ \|\nabla\tau\|_\infty < 1 \end{cases} \tag{2.2.12}$$

In this case, $I - \tau$ is a diffeomorphism, and for a signal $x(u)$, the action of a deformation τ is defined as:

$$\mathcal{L}_\tau x(u) \triangleq x(u - \tau(u)) \tag{2.2.13}$$

In [70], it is proved that, without any band-limited assumption on a signal $x \in L^2(\mathbb{R}^2)$, under mild and realistic assumption on τ, for a given J, there exists a constant $C_J > 0$ that does not depends on x or τ, such that:

$$\|S_J(\mathcal{L}_\tau x) - S_J x\| \leq C\|\nabla\tau\|_\infty\|x\| \tag{2.2.14}$$

The purpose of this theorem is to show that S_J is locally Lipschitz, w.r.t. small deformations. In particular, it implies that S_J is almost everywhere Gateaux differentiable w.r.t. the deformations, and thus it does linearize small deformations. An invariant to a subset of deformations can thus be obtained via a linear projection.

Finally, it is also often possible to reconstruct a signal from its scattering coefficients which indicates it is an almost complete representation [33]. All those properties imply that with a linear classifier, one should be able to build relevant invariants for classifications, that reduce variabilities along the translation group and deformations.

Let us give an example of an application of the Scattering Transform, in order to stress its strength. The MNIST dataset is comprised of 610^4 digits for training and $1\ 10^4$ digits for testing. On this specific dataset, most of the intraclass variability is due to small deformations and small translations. The previous properties imply that a linear projection applied on the Scattering Transform can remove those variabilities: it will reduce this problem with a-priori large variance, to a problem of sample complexity. Then a localized classifier such as a Gaussian SVM permits to discriminate the remaining variabilities, and obtain a nearly state-of-the-art result [13]. However, a supervised CNN improves even more the numerical performances [116], because it can capture more specific invariants by adapting its representation to the specific bias of the dataset.

2.2.2.2 Roto-translation Scattering

A roto-translation scattering [101, 100] improves the numerical classification performances on datasets where rigid deformations are the main variabilities. [101] introduces a wavelet transform along the roto-translation group $G = \mathbb{R}^2 \ltimes SO_2$, that is applied on the first modulus wavelet coefficients obtained in Subsection 2.2.1. Here, we write $g = (v, \theta) \in G$. Observe that in this case, $|W_1 x|(u, \theta_1, j_1) = |W_1 x|(g_1, j_1)$, with $g_1 = (u, \theta_1) \in G$. The non-commutative roto-translation group G acts on $L^2(\mathbb{R}^2)$ via:

$$\mathcal{L}_g x(u) \triangleq x(r_{-\theta}(u - v))$$

Moreover, its action is covariant with $|W_1|$, in the sense that its action does correspond to a permutation of the indexes of this modulus wavelet transform:

$$|W_1|\mathcal{L}_g x(g_1, j_1) = |W_1 x(g^{-1}g_1, j_1)| \triangleq \mathcal{L}_g|W_1|x(g_1, j_1) \tag{2.2.15}$$

As a product of $\mathbb{R}^2$ and a compact group SO_2, it is possible to define a canonical and unique (up to a multiplicative constant) notion of measure [48], which the product of the Euclidean measure and a Haar measure. In particular,

if $\forall j_1, \theta_1, |W_1 x(., \theta_1, j_1)| \in L^2(\mathbb{R}^2)$, then, since the measure along θ_1 is finite, because the rotation group is compact, and thus the modulus wavelet transform is integrable w.r.t. θ_1, hence $|W_1 x(., j_1)| \in L^2(G)$.

A wavelet transform $\tilde{W}$ can be built along G [101], and in a very similar manner to our introduction below to the Scattering Transform, permits to build a Scattering Transform which is globally (or even locally) invariant to the roto-translation group and stable to deformations. We do not study the details of its construction, as invariants to roto-translation representations are not the purpose of this thesis.

This technique was applied on textures datasets that have often a small limited number of samples per class. It implies that deep fully supervised techniques can not be applied because not enough data are available. In his thesis [102], Laurent Sifre proves that again, this representation lienarizes small deformation, is invariant to the roto-translation group and is stable to noise. Combined with a data-augmentation technique in scales and ad-hoc renormalization, [101] builds an invariant to the Euclidean group and obtain state-of-the-art results on standard benchmarks such as UIUC or UMD.

2.2.3 Scattering Transform as a Scattering Network

This section demonstrates that a Scattering Transform can be viewed as a particular type of CNN, with small kernel filters [69]. In particular, we show that the number of non-linearities (e.g. the order, which is 2 in our case) does not correspond necessarily to the effective depth of the network. We recall a demonstration similar to the "algorithme à trou" [68] to illustrate our purpose, and state this properly in our case.

2.2.3.1 Wavelet implementation

We first give sufficient conditions to build a wavelet transform via finite impulse filters. Let us set $\phi_0 = \delta_0$ and let us consider $h \in l^2(\mathbb{Z}^2)$. We also consider $\{g_\theta\}_{\theta \in \Theta} \subset l^2(\mathbb{Z}^2)$ such that for all j, we define recursively:

$$\begin{cases} \hat{\phi}_{j+1}(\omega) = \frac{1}{\sqrt{2}} \hat{h}(2^j \omega) \hat{\phi}_j(\omega) \\ \hat{\psi}_{j,\theta}(\omega) = \frac{1}{\sqrt{2}} \hat{g}_\theta(2^j \omega) \hat{\phi}_j(\omega) \quad \forall \theta \end{cases} \tag{2.2.16}$$

where $\{h, g_\theta\}_\theta$ are the finite impulse filters associated to $\{\phi, \psi\}$. In a perfect situation, no aliasing would occur and then it would be enough to apply cascade of convolutions with filters $\hat{h}_j(\omega) = \hat{h}(2^j \omega)$. Those filters are obtain by adding $2^j - 1$ zeros between each sample of $h[n]$. However, here we consider a set of filters $\{\phi_J, \psi_{j,\theta}\}_{0 \leq j < J}$, which are approximatively localized in the frequency domain. It means that they energy in Fourier of each $\psi_{j,\theta}$ is concentrated in ball of radii proportional 2^{-j}. Thus we can subsampled at intervall 2^j the result of the filtering, e.g. we consider:

$$x \star \psi_{j,\theta}(2^j p), 0 \leq j < J \tag{2.2.17}$$

and, we assume similarly that ϕ_J permits a subsampling every 2^J:

$$x \star \phi_J(2^J p) \tag{2.2.18}$$

No downsampling is applied after a filtering by g_θ to avoid aliasing effects under critic subsamplings. In this case, one has the following theorem that permits to compute a wavelet transform with appropriate downsampling, which can be found in [68]:

Proposition 1. *For $p \in \mathbb{Z}^2$, let $x_{j,-1}[p] = x \star \phi_j(2^j p)$ and $x_{j,\theta}[p] = x \star \psi_{j,\theta}(2^j p)$. Then, given $x_{j,-1}[p]$, we have: $x_{j+1,-1}[p] = x_{j,-1} \star h[2p], x_{j+1,\theta}[p] = x_{j+1,-1} \star g_\theta[p]$.*

Proof. First observe that (2.2.16) means, by taking the inverse Fourier transform:

$$\phi_{j+1}(u) = \frac{1}{\sqrt{2}} \sum_n \phi_j(u - 2^j n) h[n] \tag{2.2.19}$$

Then, we observe that:

$$\begin{aligned}
x_{j-1,-1} \star h[2p] &= \sum_{n \in \mathbb{Z}^2} h[2p-n] x_{j-1,-1}[n] \\
&= \sum_{n \in \mathbb{Z}^2} h[2p-n] \int x(u) \phi_j(2^j n - u) du \\
&= \int x(u) \sum_{n \in \mathbb{Z}^2} h[2p-n] \phi_j(2^j n - u) du \\
&= \int x(u) \sum_{n \in \mathbb{Z}^2} h[n] \phi_j(2^{j+1} p - 2^j n - u) du \\
&= \int x(u) \phi_{j+1}(2^{j+1} p - u) du \text{ by inserting (2.2.19)} \\
&= x_{j,-1}[p]
\end{aligned}$$

Similar computations stand for $x_{j,\theta}$. □

If $\phi(u) \in \mathbb{R}, \forall u \in \mathbb{R}^2$, then it is clear that $h[n] \in \mathbb{R}$ from (2.2.16). Also, if the low-pass filter is positive, h will be positive, as proved by the following lemma:

Lemma 2. *j if $\forall x, \forall u, x(u) \geq 0 \Rightarrow \forall u, x \star \phi(u) \geq 0$, then $h[n] \geq 0$.*

Proof. By linearity, it also implies $\forall u, x(u) \leq 0 \Rightarrow u, x \star \phi(u) \leq 0$, then $h[n] \geq 0$. This is in particular true for dilated wavelets. Assume $\exists n_0, h[n_0] < 0$ and let be $x = \delta_{n_0} \in l^2(\mathbb{Z})$. Then,

$$x \star \phi_{j+1}(0) = x \star h_j \star \phi_j(0) = h_{n_0} \phi_j(n_0) \leq 0 \tag{2.2.20}$$

which is absurd because $x[p] \geq 0, \forall p$ □

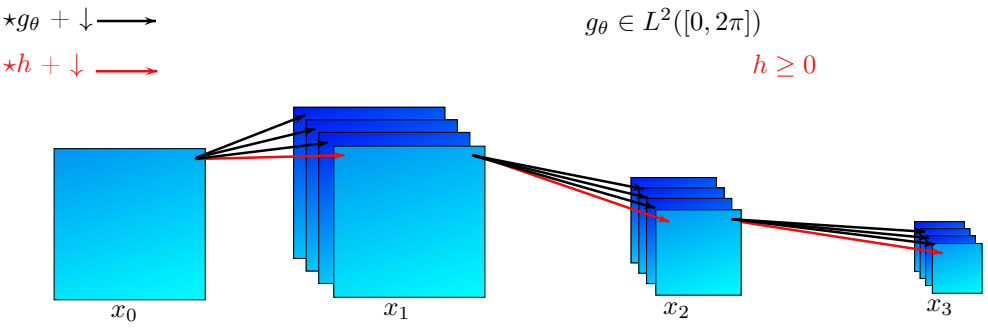


Figure 2.2.3: Wavelet transform as a cascade of small finite impulse response filters

In the following, let us instead consider $\tilde{\Theta} = \Theta \cup \{-1\}$. Again in this case, down-sampled coefficients can be obtained via:

$$\{x_{j+1,\theta}[p]\}_{\theta \in \tilde{\Theta}} = \{x_{j,-1} \star h[2p], x_{j+1,-1} \star g_\theta[p]\}_{\theta \in \Theta} \tag{2.2.21}$$

This recursive relation is expressed via Figure 2.2.3 (with an oversampling at the first scale). Next subsection shows that a Scattering Transform is a cascade of finite implulse response filters and modulus non-linearities.

Usually, the filters $\{\psi_{j,\theta}, \phi_J\}_{j<J}$ are chosen such that one obtains an energy preservation property. We assume that the input signals x are sampled at the Nyquist rate [68], e.g. they have a compact spectrum in Fourier. In this case, if the support in Fourier of the $\{\psi_{j,\theta}, \phi_J\}_{j<J}$ covers a disk with radius π say $\mathcal{B}$, and for $x \in L^2 \cap \mathcal{B}$, i.e. with support in Fourier included in this disk, we require:

$$\int |x \star \phi_J|^2(u)du + \sum_{j<J,\theta} \int |x \star \psi_{j,\theta}|^2(u)du = \int |x|^2(u)du \tag{2.2.22}$$

which implies for example that ($J = 1$ inserted in (2.2.16)) for $\omega \in \mathcal{B}$:

$$|\hat{h}(\omega)|^2 + \sum_\theta |\hat{g}_\theta(\omega)|^2 = 2. \tag{2.2.23}$$

2.2.3.2 Scattering Network

We use the properties from the previous section to show that a Scattering Transform can be interpreted as a deep cascade [69], in the case of wavelets obtained by filter bank algorithms. As reviewed in Section 2.2, the Scattering Transform is a cascade of modulus-non linearities and wavelets. Let us focus on order 0, 1 and 2 Scattering coefficients with increasing paths and scales between 0 and

J because they are the only ones used in an effective implementation. We will show that it is equivalent to applying the filters $\{h, g_\theta\}_\theta$ with modulus non-linearity $|.|$ to an input image x J times, and then selecting the paths for which at most 2 non-linearities have been applied.

First, as explained in the previous section, the low-pass filter is a cascade of J filters h with down-samplings. We will then make one assumption, which simplifies the next formula and is fundamental about the regularity of non-linearity:

$$\forall x, |x \star \psi_{j,\theta}| \star \phi_j(2^j u) \quad = \quad |x \star \psi_{j,\theta}|(2^j u) \tag{2.2.24}$$

It means that the envelope of the signal is smoother than the original signal. Let us consider $\mathcal{P}_j = \{(\theta_1, ..., \theta_j), \theta_i \in \tilde{\Theta}, \forall i \leq j\}$, and for $j \in \mathbb{N}$, we define recursively:

$$x^{j+1}_{(\theta_1,...,\theta_j,\theta_{j+1})}[p] = \begin{cases} |x^j_{(\theta_1,...,\theta_j)} \star g_\theta|[2p] & \text{if } \theta_{j+1} \in \Theta \\ |x^j_{(\theta_1,...,\theta_j)} \star h|[2p] & \text{if } \theta_{j+1} = -1 \end{cases} \tag{2.2.25}$$

Here, we subsample the results of the filtering by g_θ as the envelop of the signal is assumed to be smooth. Observe that if $h \geq 0$, then:

$$|x_{(\theta_1,...,\theta_j)} \star h|[2p] = x_{(\theta_1,...,\theta_j)} \star h[2p] \tag{2.2.26}$$

We further consider the indexes $\mathcal{P}^2_j = \{(\theta_1, ..., \theta_j) \in \mathcal{P}_j, |\{\theta_i \in \Theta\}| \leq 2\}$. In this case:

Proposition 3. *Thanks to (2.2.24), $\{x^J_\theta, \theta \in \mathcal{P}^2_J\}$ corresponds exactly to the second order coefficients at scale J.*

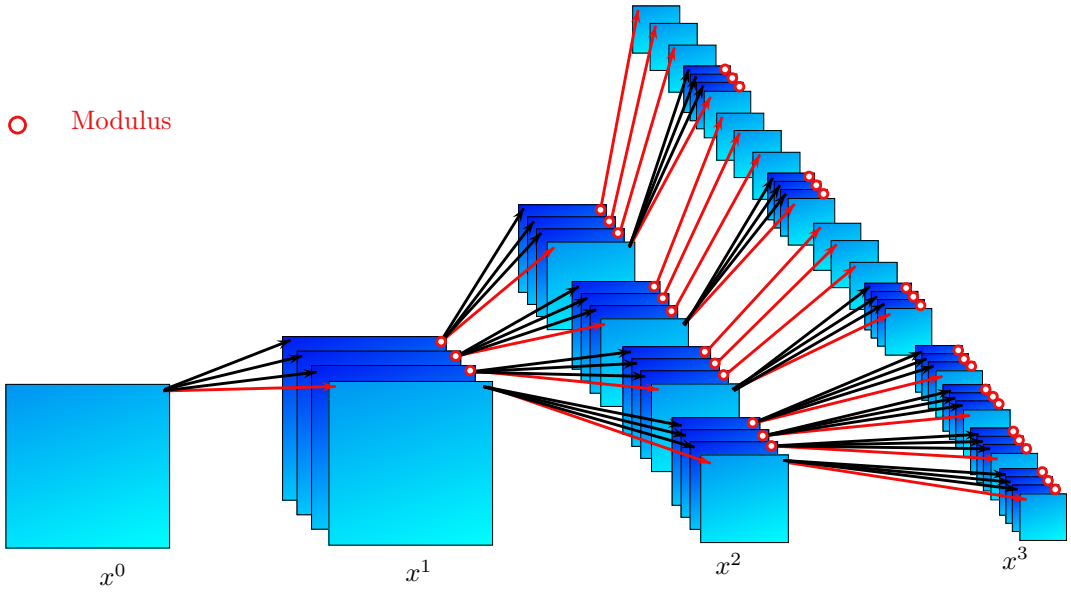
This recursive relation is again shown in Figure 2.2.4 (with an oversampling at the first scale).

Removing aliasing in the case of modulated Gaussian filters

We now discuss the aliasing issues due to a filter bank implementation. For wavelets such as Haar wavelets, h has a finite support. In this case, we are precisely in the setting of a CNN. [101] showed that it permits to obtain a fast implementation of the Scattering Transform. Typical support lenght of the filters should be about 3, except for the first layer where they have a larger support to avoid aliasing.

First, the input signals verify the Nyquist condition, thus the filters (which are in finite number) must be adapted such that they cover a domain in Fourier large enough.

Then, the previous writing can only approximately hold for Morlet wavelets because they have an infinite support, and the ratio between the dilated Fourier transforms is not periodic: the filters $\{h, g_\theta\}$ do not exist. In an implementation



$$S_3 x = \{x^3\}$$

Figure 2.2.4: Scattering Network

that tries to mimic the Morlet wavelets with cascade of small filters, the high-frequency wavelets of the transform suffer the most from the aliasing. Indeed, a high frequency Morlet has a large support in Fourier and can not be implemented because numerically the support in Fourier must be finite. There is a trade-off between the localization in Fourier and space. Besides, for Morlet wavelets, it is required to oversample the wavelet convolutions, to avoid losing high-frequencies that would be lost because of the down-sampling and the lack of localization in Fourier, and thus could not be recovered via a linear interpolation of the signal.

To have more accurate computations, each filters of the cascade could be optimized [101] to improve the approximation of (2.2.24), instead of being chosen fixed and constant equal to $\{h, g_\theta\}$. For example, without the hypothesis (2.2.24), it is necessary to chose a different set of finite impulse response filters $\{h, \tilde{g}_\theta\}$ for the first order wavelets to handle the aliasing, and to adapt the next filters to these.

Those considerations require a lot of design that do not speed-up significantly the computations, without an extra-loss in precision, and are not the purpose of this thesis: we will thus not implement the Scattering via filter banks to avoid these issues, yet rather with cascades of Fourier transform with plain filters. Oversampling in space the wavelet transform is however still necessary as we use Morlet wavelets.

Chapter 3

Scattering Networks for Complex Image Classification

This chapter shows that a Scattering Network cascaded with a simple classifier obtains results that are competitive with unsupervised algorithms on complex images datasets. We extend the experiments performed on textures or small handwritten digit datasets to complex images classification [100, 101, 102, 13, 15]. A major difference is that, in the case of natural images, the variabilities of the classification are partially known. Thus, we here restrict ourselves to representations that are built only via the roto-translation group $SO_2 \times \mathbb{R}^2$, which acts naturally on images: a Translation Scattering Transform and a Separable Rototranslation Scattering. We address the following question: are predefined discriminative representations built via convolutions along the orbits of this group enough to classify natural images?

Among images representations, unsupervised representations [9, 10] tend to perform better than predefined representations [99]. Typically, they consist in the aggregation of a local descriptor, such as SIFT, that has been encoded by an unsupervised algorithm, such a k-means. It can be interpreted as a two layers architectures, since two non-linear modules have been cascaded. These pipelines hold a lot of hyper-parameters, that must be carefully selected. We will compare our work to these architectures.

The Scattering Transform of order 2 is as well a two layers architectures. Indeed, while the first order coefficients are analog to a SIFT descriptor and have only one non-linearity, the second order coefficients allow to recover discriminative information that were lost by the averaging of the first order coefficients and are built upon two non-linearity. Contrary to bag-of-words approaches, no-extra learning is performed on the Scattering Coefficients, which imply that this representation can be considered as an unsupervised representation.

The representation we used is generic, in the sense that for very different datasets, the representation was fixed and predefined; no hyper parameter were adapted, except the window length of invariance, as the size of the images

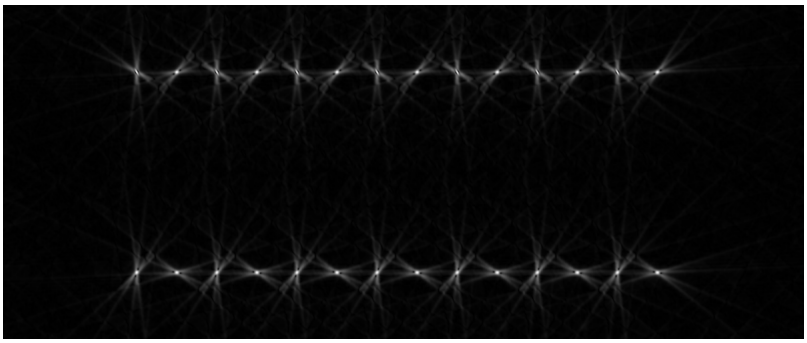


Figure 3.1.1: Amplitude of two complex signals that have identical Translation Scattering, and for which the variabilitiy is dominated by local rotations centered at integer points.

were different. Indeed, we used the exact same wavelets, without adapting the representation to a specific bias of the dataset. We obtain some results that are competitive with unsupervised algorithms, and tend to suggest that unsupervised algorithms do not capture invariants that are more discriminative than euclidean invariants. Yet, we show also that there exists a large gap with pre-trained and trained from scratch CNNs.

This chapter is divided in three sections. The first Section 3.1 introduces a variant of the scattering transform on the rotation group introduced by [101]. Next Section 3.2 reviews an algorithm to decorrelate scattering coefficients, and finally Section 3.3 reports accuracies results on CIFAR and Caltech standard datasets.

3.1 Separable roto-scattering

In this section, we build a variant of the Roto-translation Scattering that consists in a simple modification of the second wavelet transform of the order 2 Translation Scattering. Contrary to the case of textures to which Roto-translation Scattering was applied, invariance to rotation is not desired in the case of natural images. However, with ad-hoc wavelets, it is possible to build a representation that discriminate important information relative to rotations. Let us first give an example of patterns that are locally rotated, and can not be discriminated by a translation scattering, with a very similar proof to [102]:

Proposition 4. *If* $\forall \theta \neq \theta' \in \{0, \frac{\pi}{4}, \frac{\pi}{2}, \frac{3\pi}{4}\}, \psi_\theta \star \psi_{\theta'} = 0$, *then:*

$$x = \sum_{n \in \mathbb{Z}^2} \mathcal{L}_{2n}.\psi_{j,0}(u) + \mathcal{L}_{2n}.\psi_{j,\frac{\pi}{2}}(u) + \sum_{n \in \mathbb{Z}^2} \mathcal{L}_{2n+(1,1)}.\psi_{j,\frac{\pi}{4}}(u) + \mathcal{L}_{2n+(1,1)}.\psi_{j,\frac{3\pi}{4}}(u)$$

and

$$y = \sum_{n\in\mathbb{Z}^2} \mathcal{L}_{2n}.\psi_{j,0}(u) + \mathcal{L}_{2n}.\psi_{j,\frac{3\pi}{4}}(u) + \sum_{n\in\mathbb{Z}^2} \mathcal{L}_{2n+(1,1)}.\psi_{j,\frac{\pi}{4}}(u) + \mathcal{L}_{2n+(1,1)}.\psi_{j,\frac{\pi}{2}}(u)$$

have the same Translation Scattering coefficients, whereas the integer coordinates of x can be locally deduced from y by a rotation of amplitude $\frac{\pi}{4}$, as shown on Figure 3.1.1.

In the previous example, the energy of different patterns is distributed along scattering paths that are independent. To discriminate these, it is necessary to apply an operator along the rotation variable, before the spatial averaging that will discard most of the relative spatial informations. Moreover, such discriminative coefficients might be obtained by a linear operator that would recombine the wavelet transform coefficients, but this would be equivalent to change the filterbank. Indeed, let $C : \Theta \to \mathbb{R}^K, K \in \mathbb{N}$ be a linear operator along the angles and let us consider:

$$[CW_1x(u, j_1, k)] = \sum_{\theta_1} C_{k,\theta_1} x \star \psi_{\theta_1,j_1}(u) \tag{3.1.1}$$

Notice that then:

$$[CW_1x(u, j_1, k)] = x \star \sum_{\theta_1} C_{k,\theta_1} \psi_{\theta_1,j_1}(u) \tag{3.1.2}$$

and it was equivalent to first perform a convolution with a different filter. instead of designing a different filter bank, we take advantage of an operator applied on the modulus of the wavelet transform.

The Roto-translation Scattering consists in a non-separable wavelet transform on the semi-direct rototranslation group as explained in Subsection 2.2.2.2, which is applied on the modulus of the wavelet coefficients. Here, we replace it by a separable wavelet transform along rotation and spatial variables. This Scattering Transform can still be reformulated as a Scattering Network, as it can be implemented via an "algorithme à trou" as in Subsection 2.2.3.2. We will consider 3 dimensional morlet wavelets, for which the the filter bank algorithm is true up to approximations terms.

Section 4 will show that a deep network recombines efficiently the Scattering paths, in order to build locally invariants to rotation representation. Following this idea, we propose to discriminate angular high-frequencies by applying a wavelet transform along θ_1 on the first modulus wavelets coefficients, that will recombine angles along rotation. Contrary to deep networks, no extra-learning parameters are involved, except the maximum scales of the respective wavelet transforms along space and angles.

As in Section 2.2.1, let us consider the modulus wavelets coefficients given by:

$$|W_1x|(u, \theta_1, j_1) = |x \star \psi_{j_1,\theta_1}|(u)$$

In the case of natural images, $\theta_1 \to |W_1 x|(u, \theta_1, j_1)$ is a smooth application if the selectivity in angle is not too abrupt. For example, a worst case in space is given by $x = \delta_0$. In the case of Gabor filters, for example:

$$|W_1 \delta_0|(u, \theta_1, 0) \propto e^{-\frac{u^T [r_{-\theta_1}] B [r_{\theta_1}] u}{2\sigma^2}} \tag{3.1.3}$$

where $[r_{\theta_1}]$ is the matrix of the rotation by θ_1. Indeed, for $u_0 = (0, \sqrt{2}\frac{\sigma}{\xi})$, a limited development of θ_1 gives around 0:

$$\log(|W_1 \delta_0|(u_0, \theta_1, 0)) = -1 + \theta_1^2 (1 - \frac{1}{\xi}) + \mathcal{O}(\theta_1^3)$$

It shows that the smaller is ξ, the larger will be the derivative which is related to the local regularity of the signal. Thus, the sampling in angle must be adapted to the selectivity in angles. It is important, in order to further define a convolution along this axis.

Let us now build the separable roto-translation scattering. Similarly to deep convolution network architectures, we also recombine the information in these image frames indexed by the angle θ_1. To understand how to do so, let us compute the wavelet coefficients of a rotated image $\mathcal{L}_{r_\alpha}.x(u) \triangleq x(r_{-\alpha} u)$ by angle α:

$$|(\mathcal{L}_{r_\alpha}.x) \star \psi_{j_1,\theta_1}|(u) = |x \star \psi_{j_1,\theta_1 - \alpha}|(r_{-\alpha} u) \triangleq \mathcal{L}_{r_\alpha} |x \star \psi_{j_1,\theta_1}|(u) \tag{3.1.4}$$

It rotates the spatial coordinates u but also "translates" by α the angle parameter θ_1. It means that the representation is covariant with the action of rotation.

As explained at the beginning of this section, our goal is not to build a rotation invariant representation but a representation which linearizes variabilities along rotation angles. These rotation variabilities can thus be discriminated or removed by a linear classifiers at the output. We thus do not use a rotation invariant scattering representation as in [101]. To build a representation which is stable to rotations, and to deformations, we compute a wavelet transform along the angle parameter θ_1. A proof of the stability can be adapted from [101].

We perform convolutions along θ_1, with angular one-dimensional wavelets:

$$\breve{\psi}_k(\theta_1) = 2^{-k} \breve{\psi}(\frac{\theta_1}{2^k})$$

and

$$\breve{\psi}_{-k}(\theta_1) = 2^{-k} \breve{\psi}^*(\frac{\theta_1}{2^k})$$

where $\breve{\psi}$ is a one dimensional Morlet wavelets, which is given by:

$$\breve{\psi}(\theta_1) = \breve{C}(e^{i\breve{\xi}\theta_1} - \breve{\kappa}) e^{-\frac{\theta_1^2}{2\breve{\sigma}}} \tag{3.1.5}$$

where the parameters that are chosen are completely analogous from Equation (2.2.3). Considering the conjugate of the wavelet is absolutely mandatory, because as we will see below, we will consider a complex wavelet transform and not a real transform, because the input will be in the complex domain. We also consider $\breve{\phi}_K$ being a 1 dimensional real and positive averaging such that the little hood paley is approximately verified, for complex valued signals, eg there exists $0 \leq \epsilon < 1$, such that for any angular frequency ω:

$$1 - \epsilon \leq \sum_{|k| \leq K} |\widehat{\breve{\psi}_k}(\omega)|^2 + |\widehat{\breve{\phi}_K}(\omega)|^2 \leq 1 \tag{3.1.6}$$

The resulting wavelet transform $\breve{W}y = \{\breve{\psi}_k \star y, \breve{\phi}_K \star y\}_{|k| \leq K}$ allows to compute separable convolutions along both the 2d spatial variable u and the angle variable θ_1, with a 3d separable complex wavelet defined by two wavelets:

$$\breve{\psi}_{0,j_2,\theta_2,k}(u, \theta_1) = \psi_{j_2,\theta_2}(u)\breve{\psi}_k(\theta_1) \tag{3.1.7}$$

and:

$$\breve{\psi}_{1,j_2,\theta_2}(u, \theta_1) = \psi_{j_2,\theta_2}(u)\breve{\phi}_K(\theta_1) \tag{3.1.8}$$

where (j_2, θ_2) corresponds to the spatial frequencies, whereas k corresponds to the scale of the wavelets along θ_1. It is a separable product of a spatial wavelet $\psi_{j_1,\theta_2}(u)$ of scale j_1 and an angular wavelet $\psi_k(\theta)$ of scale k for $1 \leq |k| \leq K < \log_2 L$. If $\breve{\psi}_k(\theta_1)$ are one-dimensional Morlet wavelets, then the resulting separable wavelet transform $\breve{W}_2 = W_2\breve{W}$ is a stable and invertible operator [68], which nearly preserves the signal norm, bounded by (3.1.6). The latter can be proved by inserting Equation (2.2.6) and Equation (3.1.6) because the Fourier transform of a separable operator is the product of each of the Fourier transform. As this wavelet transform is separable,

$$\breve{W}_2 = W_2\breve{W} = \breve{W}W_2.$$

Here again, the wavelet transform modulus for $j_2 > j_1$ is computed with a three-dimensional separable convolution along the spatial and angular variables (u, θ), and it performs a sub-sampling along both variables. The 3D separable wavelet transform $\breve{W}_2$ could be either computed with a cascade of filtering across the deep network layers introduced in 2.2.3.2, or directly with 3 dimensional convolutions calculated with FFT's.

The deep roto-scattering is computed by cascading the modulus $|W_1|$ of a first 2 dimensional spatial wavelet transform, followed by the modulus $|\breve{W}_2|$ of a second 3 dimensional separable wavelet transform along space and angles, followed by the averaging A_J introduced in

$$\breve{S}_J x = \{A_J x, A_J|W_1|x, A_J|\breve{W}_2||W_1|x\} \tag{3.1.9}$$

Since W_1, $\breve{W}_2$ and A_J are contractive operators it guarantees that S_J is also contractive and hence stable to additive perturbations, as in the case of translation scattering, which is proven in [101]. It means that:

$$\|\breve{S}_J x - \breve{S}_J y\| \leq \|x - y\| \tag{3.1.10}$$

Moreover, since the wavelet transforms W_1, $\breve{W}_2$ and A_J are Lipschitz-stable relatively to deformations [70], $\breve{S}_J$ is also Lipschitz-stable and hence linearizes small deformations. This guaranties to avoid the instabilities observed on deep networks such as Alex-net [77] where a small image perturbation can considerably modify the network output and hence the classification.

Dimensionality of the representation

The wavelet transform $\tilde{W}$ preserves the number of coefficients, thus the dimensionality of the Translation and the Separable Roto-translation Scattering are the same. Good quality images can be reconstructed from scattering coefficients as long as the number of scattering coefficients is larger than the number of image pixels [14].

3.2 Decorrelating Scattering coefficients with Orthogonal Least Square

In this section, we review a dictionary learning algorithm to decorrelate Scattering coefficients which are introduced in the previous section, called Orthogonal Least Squares (OLS), described in [21]. We first motivate the use of this algorithm with scattering coefficients, and we explains its steps. For example, a singularity (or a pattern) can excite a large panel of scattering path across scales, which introduce correlations between those coefficients. Decorrelating the representation permits to reduce the variance of a representation and to separate independent coefficients, however it also introduces a bias (because it is a linear projection operator) that can be significant during experiments, as explained in the numerical experiments, Section 3.3.

The number of scattering coefficients is of the same order as the original image. It provides a nearly complete signal representation, which allows one to build a very rich set of geometric invariants with linear projection operators. The choice of these linear projection operators is done at the supervised classification stage with an SVM. As said below, scattering coefficients are strongly correlated.

In the case of non-linear classifiers, results are improved by reducing the variance of the representation, with a supervised feature selection, which considerably reduces the number of scattering coefficients before computing an SVM classifier. This is implemented with a supervised orthogonal least square regression [21, 8], which greedily selects coefficients with a regression algorithm.

A logarithm non-linearity is applied to scattering coefficients in order to separate low frequency multiplicative components due to the variations of illuminations. These low-frequency modulations add a constant to the logarithm of scattering coefficients which can then be removed with an appropriate lin-

ear projector by the final classifier. Also, it linearizes exponential decay of the scattering coefficients across scales.

In the following, we denote by $\Phi x(u)$ the logarithm of scattering coefficients at a scale 2^J, that can be either the roto-scattering or the translation Scattering. We are given a set of training images $\{x_i\}_i$ with their class label. The orthogonal least square selects a set of features adapted to each class C with a linear regression of the one-versus-all indicator function

$$f_C(x) = \begin{cases} 1 & \text{if } x \text{ belongs to class } C \\ 0 & \text{otherwise} \end{cases} . \tag{3.2.1}$$

It iteratively selects a feature in the dictionary and updates the dictionary, which implies that the size of the dictionary is reduced at each iteration. Let $\Phi^k x = \{\phi_p^k x\}_p$ be the dictionary at the k^{th} iteration. We select a feature $\phi_{p_k}^k x$, and we update the dictionary by decorrelating all dictionary vectors, relatively to this selected vector, over the training set $\{x_i\}_i$:

$$\tilde{\phi}_p^{k+1} = \phi_p^k - \Big(\sum_i \phi_{p_k}^k(x_i)\, \phi_p^k(x_i) \Big) \phi_{p_k}^k . \tag{3.2.2}$$

Each vector is then normalized

$$\phi_p^{k+1} = \tilde{\phi}_p^{k+1} \Big(\sum_i |\tilde{\phi}_p^{k+1}(x_i)|^2 \Big)^{-1} . \tag{3.2.3}$$

The k^{th} feature $\phi_{p_k}^k x$ is selected so that the linear regression of $f_C(x)$ on $\{\phi_{p_r}^r x\}_{1 \le r \le k}$ has a minimum mean-square error, computed on the training set, e.g.:

$$\phi_{p_k}^k = \arg\min_{\phi_p^{k-1}} \|f_C(x) - \phi_p^{k-1} x\|^2 \tag{3.2.4}$$

This is equivalent to finding $\phi_{p_k}^k$ in Φ^k which maximizes the correlation:

$$\phi_{p_k}^k = \arg\max \sum_i f_C(x_i)\, \phi_p^k(x_i).$$

The orthogonal least square regression thus selects and computes K scattering features $\{\phi_{p_k} x\}_{k<K}$ for each class C, which are linearly transformed into K decorrelated and normalized features $\{\phi_{p_k}^k x\}_{k<K}$. For a total of n_C classes, the union of all these feature defines a dictionary of size $M = K\, n_C$. They are linear combinations of the original log scattering coefficients $\{\phi_p x\}_p$. This dimension reduction can thus be interpreted as a last fully connected network layer, which outputs a vector of size M. The parameter M governs the bias versus variance trade-off. It can be adjusted from the decay of the regression error of each f_C or fixed a priori. In classification experiments, M is about 30 times smaller than the size of the original scattering dictionary. We discuss in the next Section 3.3 applications to standard datasets.

3.3 Image classification results on standard benchmarks

The unsupervised representation that we built can be fed to a supervised classifier, in order to measure its discriminability performances. To this end, we apply SVMs algorithms that are discriminative models and a perceptron on the scattering coefficients, which is a neural network method. First we describe the datasets and the hyper parameters of the Scattering Transforms, and then numerical accuracies will be discussed in Subsection 3.3.1 and 3.3.2.

We compare the performance of a scattering network with state-of-the-art algorithms on CIFAR and Caltech datasets, which include complex object classes, at different or fixed resolutions. In particular, unsupervised algorithm will be compared because Scattering Transform lands in this type of algorithm. Images of each databases are rescaled to become square images of $N^2 = 2^{2d}$ pixels. The scattering transform depends upon few parameters which are fixed a priori. The maximum scale of the scattering transform is set to $2^J = 2^{d-2}$. Scattering coefficients are thus averaged over spatial domains covering $1/4$ of the image width, and coefficients sampled over a spatial grid of 4×4 points, a final down-sampling being performed without degrading classification accuracies. This preserves some coarse localization information.

Coefficients are computed with Morlet wavelets having $L = 8$ orientations. The wavelet transform along these $L = 8$ angles are computed at a maximum scale $2^K = L/2$, which corresponds to a maximum angular variation of $\frac{\pi}{2}$. Indeed these object recognition problems do not involve larger rotation variability. The resulting scattering representation is nearly complete as previously explained. It is computed independently along the 3 color channels YUV. We apply a logarithm to separate illumination components. The classifier is implemented by first reducing the dimensionality, to $M = 2000$ feature vectors on CIFAR-10 for instance, with the orthogonal least square regression previously introduced, and applying a SVM.

To stress the genericity of our representation, we use the same architecture and same hyper parameters for each datasets, apart from the number M of selected coefficients, which increases proportionally to the size of the scattering representation, which depends upon the image size.

Caltech-101 and Caltech-256 are two color image databases, with respectively 101 and 256 classes. They have 30 images per class for training and the rest is used for testing. Caltech images are rescaled to square images of $N = 2^{2d} = 256^2$ pixels. Average per class classification results are reported with an averaging over 5 random splits. We removed the clutter class both from our training and testing set.

CIFAR are more challenging color image databases due to its high class variabilities, with $6\,10^4$ tiny colors images of $N = 2^{2d} = 32^2$ pixels. CIFAR-10 has 10 classes with $5\,10^3$ training images per class, whereas CIFAR-100 has 100 classes with $5\,10^2$ training images per class.

3.3.1 Linear and Gaussian SVMs

We first classify our scattering coefficients using kernel methods. Linear SVMs are discriminant classifiers that select few samples to build a classification boundary. It uses the linear metric define by $k(x,y) = \langle x, y\rangle$. They have proved generalization bound properties in the case many unlabeled data are available. Contrary to a Linear SVM, a Gaussian SVM classifier uses a non-linear kernel, $k_\sigma(x,y) = e^{-\frac{\|x-y\|^2}{\sigma^2}}$, where σ is the bandwidth of the kernel. It can be viewed as a localization of the linear metric, as when $\sigma \to \infty$, the metric which is defined approaches the linear SVM. The selected features of the previous section are then provided to SVM classifiers.

A supervised SVM classifier is applied to the logarithm of the scattering transform coefficients $\Phi x = \log S_J x$, as in texture classification [101]. In both case, a large oversampling is used (e.g. $\alpha = 2$, Subsection 2.2.1) to compensate aliasing effects and linearizing small translations. As the scattering representation are precomputed once, this is not a bottleneck in this pipeline. The logarithm enables the linear SVM to be invariant to variations of illuminations. Indeed, these variations introduce low-frequency multiplicative factors in scattering coefficients which become additive with the logarithm.

The linear SVM is invariant to these components if the generated weight vector is orthogonal to the low-dimensional space generated by these variabilities. The logarithm of scattering coefficients are standardized by subtracting their mean and normalizing their variance, before applying the linear SVM using a "one versus all" strategy, with cross-validated parameters.

A Gaussian SVM can not build exact invariance to linear variabilities, yet it is a locally linear classifier. In particular, this classifier has the ability to build a linear decision boundary if necessary. The variance of the Gaussian kernel is set to the average norm of the scattering vectors, calculated from the training set. This large variance performs a relatively small localization in the feature space, but it reduces classification errors, as we will see below.

Table 3.1 reports the classification accuracy of a second order roto-translation scattering algorithm in various setting, for CIFAR and Caltech databases. It is compared to state of the art algorithms, divided in four categories. "Prior" feature algorithms apply a linear or an RBF type classifier to a predefined set of features, which are not computed from training data. Scattering, SIFT and HOG vectors, or deep networks with random weights belong to this Prior class. "U. Deep" algorithms correspond to unsupervised convolutional deep learning algorithms, whose filters are optimized with non-labeled training data, before applying a linear classifier or a Gaussian kernel SVM. "Unsup." algorithms transform SIFT type feature vectors or normalized pixel patches, with one, two or three successive sparse dictionaries computed by unsupervised learning. It may then be followed by a max-pooling operator over a pyramid structure [57]. "Sup." algorithms compute feature or kernel representations, which are optimized with supervised learning over labeled training data. In this case, the training may be performed on a different databases such as ImageNet, or may include a data augmentation by increasing the dataset with affine transforma-

Method	Acc.	Type
Translation Scattering + Gaussian SVM	70.0	Prior
Separable Roto-translation Scattering + Linear SVM	75.6	Prior
Separable Roto-translation Scattering + Gaussian SVM	74.5	Prior
Separable Roto-translation Scattering + OLS + Gaussian SVM	79.9	Prior
Random features [96]	50.3	Prior
CDBN [61]	65.4	U. Deep
M-HMP [9]	82.5	Unsup.
SLC [73]	81.0	Unsup.
Ask the locals [10]	77.3	Unsup.
RFL [53]	75.3	Unsup.
CNN [125]	92.3	Sup.

(a) Results for different types of representations, on Caltech101

Method	Acc.	Type
Translation Scattering + Gaussian SVM	80.3	Prior
Translation Scattering + Perceptron	84.7	Prior
Separable Roto-translation Scattering + Linear SVM	78.4	Prior
Separable Roto-translation Scattering + Gaussian SVM	81.5	Prior
Separable Roto-translation Scattering + OLS + Gaussian SVM	82.3	Prior
RFL [53]	83.1	Unsup.
NOMP [64]	82.9	Unsup.
LIFT [104]	82.2	U. Deep
Exemplar CNN [34]	84.3	U. Deep
CNN [120]	96.1	Sup.

(b) Results for different types of representations, on CIFAR10

Method	Acc.	Type
Separable Roto-translation Scattering + Linear SVM	50.5	Prior
Separable Roto-translation Scattering + OLS + Gaussian SVM	56.8	Prior
RFL [53]	54.2	Unsup.
NOMP [64]	60.8	Unsup.
CNN [120]	81.2	Sup.

(c) Results for different types of representations, on CIFAR100

Method	Acc.	Type
Separable Roto-translation Scattering + Linear SVM	46.2	Prior
Separable Roto-translation Scattering + OLS + Gaussian SVM	43.6	Prior
M-HMP [9]	50.7	Unsup.
SLC [73]	46.6	Unsup.
Ask the locals [10]	41.7	Unsup.
CNN [125]	86.0	Sup.

(d) Results for different types of representations, on Caltech256

Table 3.1: Classification rates for Caltech and CIFAR data bases, with the same Scattering network, compared to state of the art unsupervised learning algorithms and supervised convolution networks, with algorithms provided in reference.

tions and deformations. Supervised deep convolution networks or supervised kernel learning are examples of such algorithms.

Let us emphasize again that we are using the same scattering representation, besides image size adaptation, for Caltech and CIFAR databases. RFL (Receptive Field Learning) [53] is the only unsupervised learning algorithm which reports close to state of the art results, both on Caltech and CIFAR data bases. RFL does not perform as well as a scattering on Caltech and CIFAR-100, and slightly better on CIFAR-10. This illustrates the difficulty to have a single algorithm which works efficiently on very different databases. We reported the result on CIFAR-100 from [53] via [67]. We first compare the numerical results obtained with a Gaussian SVM, and then discuss the accuracy, depending on the classifier on top of the Scattering Network.

3.3.2 Comparison with other methods

3.3.2.1 Comparison with unsupervised methods

Scattering gives better classification results than all Prior features classification on Caltech-101, as shown by Table 3.1. Convolutional network with random filters on mono-CIFAR-10 (gray level CIFAR-10) have an accuracy of 53.2% in [96]. Color information improves classification results by at most 10% on all algorithms, so it remains well below scattering accuracy. No result is reported on CIFAR-100 using predefined "prior" feature classifiers.

The unsupervised classification algorithm reporting state of the art classification results on CIFAR is different than the one of Caltech, but still based on patch of pixels encoded with an unsupervised dictionary, max pooled [24]. For CIFAR, there are enough images to obtain state of the art results with a supervised training of convolution networks, without data augmentation. For CIFAR-100, previous state of the art with unsupervised learning lead to 51.7% error [24, 67], which is comparable with the accuracy of scattering networks. The accuracy of recent results improves by 9.7% this state of the art by using a variant of OMP during the encoding step [64]. The results of scattering networks are comparable, achieving 56.8% on CIFAR-100, without feature selection, since results on this databases are known to scale well towards much larger data bases. For CIFAR as well as Caltech data bases, scattering network results are at the level of state of the art algorithms in 2012 [9, 10, 56], both for unsupervised and supervised learning, and is thus two years late. Further refinements need to be added to this representation, and in particular reduce the output dimensionality of the scattering network, to reduce intra-class variability, which could be potentially done with a cascaded CNN.

3.3.2.2 Comparison with supervised methods

The best classification results are obtained by supervised deep convolutional networks [120, 45, 125]. They improve non-supervised accuracy by about 15% on CIFAR and Caltech datasets. The improvement on CIFAR-100 is smaller

than on CIFAR-10 because there is only 500 samples per classes for supervised training, as opposed to 5000. The Caltech databases does not have enough training sample to train a supervised deep network: there are not enough labeled images to train algorithms that outperform unsupervised techniques [122]. This is particularly important in Caltech-256 because each class has a considerable variability and relatively few training images per class. We thus report classification results obtained by the supervised Alex-network trained on ImageNet, to which is applied a linear SVM classifier which is trained on Caltech [125]. Although this deep network was not trained on Caltech, it still achieves the state of the art on this databases. Experiments show that if the training and testing image datasets are different, a supervised deep network provides a feature vector having a lower accuracy for classification, but this accuracy is not dramatically reduced. It indicates that supervised deep classifiers are learning generic image representations which are likely to capture more complex geometric properties than unsupervised algorithms or a roto-translation scattering transform.

A scattering network with a Linear or Gaussian SVM yields classification results which are about the best unsupervised learning algorithms for Caltech-101 and Caltech 256.

3.3.2.3 Scattering combined with different classifier

Table 3.2 gives the classification accuracy for different scattering projections, followed by a Gaussian SVM, on the datasets CIFAR-10 and Caltech-101. We consider predefined projection that selects first or second order, and then supervised projection learned via the OLS algorithm. First order scattering coefficients are comparable to SIFT [15], but are calculated over larger neighborhoods. Second order scattering coefficients computed with translated wavelets (no filtering along rotations) reduces the error by 10%, which shows the importance of this complementary information. Incorporating a wavelet filtering along rotations, leads to a further improvement of 4.5% on Caltech-101 and 1.2% on CIFAR-10. Rotations produce larger pixel displacements on higher resolution images. It may explain why improving sensitivity to rotations plays a more important role on Caltech images, which are larger. Adding a feature reduction by orthogonal least square reduces the error by 5.4% on Caltech-101 and 0.7% on CIFAR-10. The orthogonal least square has a bigger impact on Caltech-101 because there are less training examples per class, so reducing the variance of the estimation has a bigger effect.

Table 3.1 shows that applying a Gaussian SVM improves by a considerable margin the classification accuracy on most of the datasets. It indicates that all the important variabilities of classification have not been linearized, and thus, a non-linear classifier can handle them better. The linear SVM performs better on Caltech-256, yet this might be due to ad-hoc normalization phenomenon and specific bias of this dataset. No results of the OLS combined with a linear SVM are displayed, as it systematically reduces the classification accuracies. It shows that this algorithm introduces a bias (due to the projection) that can not help a linear classifier. An OLS algorithm followed by a Gaussian classifier can be

Method	**Caltech101**	**CIFAR10**
Translation, order 1	59.8	72.6
Translation, order 2	70.0	80.3
Translation, order 2 + OLS	75.4	81.6
Roto-translation, order 2	74.5	81.5
Roto-translation, order 2 + OLS	79.9	82.3

Table 3.2: Classification accuracy with 5 scattering configurations. First a translation scattering up to order 1, then up to order 2, then with an Orthogonal Least Square (OLS) feature reduction. Then a roto-translation scattering up to order 2, then with an OLS feature reduction.

interpreted as a two layers classifier, and permits to fill by a limited margin this gap. Training a neural network that consists in three fully connected layers permits to substantially reduce this gap, and we exhibit state-of-the-art results in the unsupervised case of CIFAR10, and will be studied in the next section.

Chapter 4

Improving Scattering with Hybrid Networks

In this chapter, we combine a Scattering Network with a fully supervised learned CNN and show accuracies that are competitive with the state of the art on the dataset ImageNet2012. As explained in the previous chapter, Scattering Networks are predefined deep networks that use geometrical priors and which are competitive with unsupervised algorithm. The first layer of [56] suggests that initial layers of a CNN are composed of atoms similar to wavelets [115]. In order to extend this idea and avoid learning those initial filters, we consider Hybrid Networks that consists in the cascade of a Scattering Network, and a supervised CNN. Our work aims to show that it is as well a competitive and generic initialization of the first layer of a CNN that does not reduce discriminative informations that are exploited by those algorithms. In particular, we show the Scattering Networks do improve small data regimes and permit the building of a competitive local descriptor, the Shared Local Encoder. We also release a very fast GPU implementation of the Scattering Network.

Scaling the Scattering Transform for computations on ImageNet requires to implement a Scattering Network on GPUs. While no extra computation time for back-propagating a gradient is required, FFT based convolutions make this implementation challenging. Indeed, GPUs have a very limited memory, which requires rethinking the order of the computations performed in [1].

We demonstrate that a Scattering Network permits preserving discriminative and relevant information for CNNs classification. Indeed, cascading the scattering module with a ResNet leads to state-of-the-art results on ImageNet. We use again the generic representation of Chapter 2.2, without extra-adaptation, except the maximum scales of the averaging.

We also show that in the case of the ResNet, at a given number of parameters, the Hybrid and non Hybrid networks performs similarly while the Hybrid Network is significantly more shallow.

The geometric priors that are incorporated are also useful in the case of

limited samples, which is a a problem of interest. Indeed, those situations are really common in real world dataset, as supervised data is either costly either complex to produce, as in the case of medical imaging for example. We show state-of-the-art results in the limited data setting on CIFAR10 and STL10 are obtained via a Hybrid Network and the accuracies are above the fully supervisedly learned counterpart (e.g. which does not include the Scattering Transform). This indicates that incorporating geometric invariants regularizes the process of learning of the cascaded CNN, as they are not required to be learned anymore.

We introduce a novel local descriptor, the Shared Local Encoder (SLE), which is fully learned through supervision. It consists in a cascade of 3 fully connected layers applied on scattering representation. Aggregating non-overlapping patches of this descriptor leads to AlexNet-like performance. Finally, we prove that an explicit invariant to rotations is learned by the SLE.

This chapter is divided into three sections. First, Section 4.1 explains how to implement a fast Scattering Transform on GPUs. Then, we develop the training procedure and numerical results obtained by a Hybrid architecture, in Section 4.2. The last Section 4.3 introduces the SLE and its properties.

4.1 Fast implementation of Scattering Networks on GPUs

The implementation of a Scattering Network must be re-thought to benefit from the 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 intermediary computations. In this section, we do not review a faster algorithm, yet how to solve this problem of memory.

4.1.1 Tree implementation of computations

We recall the algorithm to compute a Scattering Transform and its implementation as done in [15], for order 2 Scattering with a scale of J and L angles. 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 4.1.1 Let us consider an input signal $x[p]$ of size N^2 which is a power of 2, which has 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 boundaries effects that are inherent to periodic convolutions. The filter bank corresponds, as described in Section 2.2.1, to $JL+1$ filters, e.g. : $\{\psi_{\theta,j}, \phi_J\}_{\theta,j\leq J}$. We only consider periodized filters, e.g.:

$$\tilde{\psi}_{\theta,j}(u) = \sum_{k_1,k_2} \psi_{\theta,j}(u + (Nk_1, Nk_2)) \tag{4.1.1}$$

A first wavelet transform must be applied on the input signal of the Scattering Transform. To this end, a FFT (a) of size N is applied. Then, JL dot-wise

multiplications (b) 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 respectively by a factor 2^j and 2^J, in order to reduce the computational complexity of the next operations. This is performed by a periodization (c) of the signal in the Fourier domain, which is equivalent to a spatial down-sampling in the spatial domain, e.g. 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, because they introduce an aliasing. An iFFT (a') is then applied on each of the resulting filtered signals, that are of size $\frac{N^2}{2^j}, j \leq J$. A modulus operator (d) is applied on each of the signals, except on the real filters given. It means there are $\{|x \star \tilde{\psi}_{j_1,\theta_1}[2^{j_1} p]|\}_{\theta_1 \leq L, j_1 < J, p_1 \leq \frac{N}{2^{j_1}}, p_2 \leq \frac{N}{2^{j_1}}}$ that will be reused at the next layer, and a low pass filter. It requires to store $\mathcal{O}_1$ intermediary coefficients, where:

$$\begin{aligned} \mathcal{O}^1 &= L \sum_{j_1=0}^{J-1} \frac{N^2}{2^{2j_1}} + \frac{N^2}{2^{2J}} \\ &= N^2[L(4\frac{1-4^{-J}}{3}) + 4^{-J}] \end{aligned}$$

This exact step is iterated one more time, on each of the JL wavelet modulus signals, yet only considering increasing paths. It means that a wavelet transform and a modulus applied on a signal $|x \star \tilde{\psi}_{j_1,\theta_1}[2^j p]|$ leads to:

$$\mathcal{O}^2_{j_1} = L \sum_{j_2=j_1+1}^{J-1} \frac{N^2}{2^{2j_2}} + \frac{N^2}{2^{2J}} \tag{4.1.2}$$

$$= N^2[L(4\frac{4^{-j_1-1} - 4^{-J}}{3}) + 4^{-J}] \tag{4.1.3}$$

Consequently, the total number of coefficients relatively to the second order, that is stored will be:

$$\mathcal{O}^2 = \sum_{j_1=0}^{J-1} L\mathcal{O}^2_{j_1} \tag{4.1.4}$$

$$= \sum_{j_1=0}^{J-1} LN^2[L(4\frac{4^{-j_1-1} - 4^{-J}}{3}) + 4^{-J}] \tag{4.1.5}$$

$$= LN^2[-JL\frac{4^{-J+1}}{3} + J4^{-J} + \frac{L}{3}(4\frac{1-4^{-J}}{3})] \tag{4.1.6}$$

Finally, an averaging is applied on the second order wavelet modulus coefficients, which leads to:

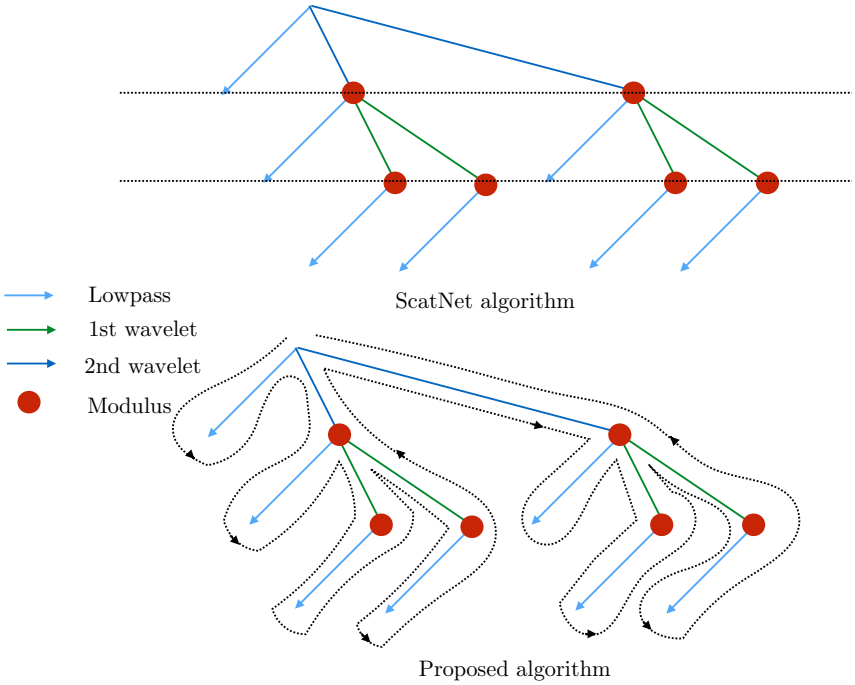


Figure 4.1.1: Tree of computations for a Scattering Transform implemented via FFT's

$$\mathcal{O}^3 = \frac{J(J-1)}{2} L^2 \frac{N^2}{2^{2J}} \tag{4.1.7}$$

At the end, the total number of coefficients stored is:

$$\mathcal{O} = \mathcal{O}^1 + \mathcal{O}^2 + \mathcal{O}^3 \tag{4.1.8}$$

For example, for $J = 2, 3, 4, L = 8$, and $N = 256$, which corresponds to the setting used on ImageNet, we numerically have $\mathcal{O} \approx 2M, 2.5M, 2.6M$ 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 colored images, we thus need at least 6GB of memory, simply to store the intermediary tensors used by the scattering, which does not take in account extra-memory used by libraries such as cuFFT for example. In particular, it shows that a typical GPU with 12GB of memory can not handle simultaneously that much images.

4.1.2 Memory efficient implementation on GPUs

We now describe a GPU implementation which tries to minimize the memory usage during the computations. The procedure (a/a'), (b), (c) and (d) of the

Params	pyScatWave	Speed up w.r.t. ScatNetLight
$32 \times 32 \times 3 \times 128$ $(J = 2)$	0.03s	8×
$256 \times 256 \times 3 \times 128$ $(J = 2)$	0.71s	225×

Table 4.1: Different computation times between CPU and GPU

previous section can be fully implemented on GPUs, in a GPU friendly way. They are fast, and can be implemented in batches, which permits to perform parallel computations of the scattering representation. This is necessary for the deep learning pipeline, that commonly uses batches of data augmented samples. However, a GPU is a device with a limited amount of memory.

To this end, we propose to perform an "infix parcours" of the tree of computations of the scattering. To this end, we introduce $\{\tilde{U}_j^1, \tilde{U}_j^2\}_{j \leq J}$ that are two sequences of temporary stamp variables of length $\{\frac{N}{2^j}\}_{j \leq J}$ and a $\tilde{U}_0^0$ of length N. It means that the total amount of memory that will be used is at most $5N^2$. Here, a colored image of $N = 256$ coefficients correspond to about $0.98M$ coefficients, at most. It divides by 2 at least the memory sized used and permits to scale on ImageNet. The Algorithm 4.1 presents the algorithm we used in PyScatWave [84], which gives an identical output relatively to the one described previously. The Table 4.1 introduces the speed-up for different values of tensors on a TitanX, compared with ScatNetLight [85].

We shall mention that it is possible to store the intermediary computed scattering coefficients, via a system of cache. In this case, it is possible to obtain a speed up by a large factor since no extra computations are required to compute the earlier layer that are the computationally extensive in comparison with deeper layers, that have been downsampled.

4.2 Cascading a deep CNN: the ResNet

We now demonstrate cascading deep CNN architectures on top of the scattering network can produce high performance classification systems. First, we justify that a Scattering Network is an ideal initialization of a CNN in Subsection 4.2.1. We then apply hybrid convolutional networks on the Imagenet ILSVRC2012 dataset (Subsection 4.2.2) as well as the CIFAR-10 dataset (Subsection 4.2.3) and show that they can achieve performance comparable to deep end-to-end learned approaches. In the last Subsection 4.2.4, we evaluate the hybrid networks in the setting of limited data by utilizing a subset of CIFAR10 as well as the STL10 dataset and show that we can obtain substantial improvement in performance over analogous end-to-end learned CNNs.

4.2.1 Scattering as an ideal initialization

We now motivate the use of a supervised architecture on top of a Scattering Network. Indeed, as said in the previous section, Scattering transforms have

Algorithm 4.1 Algorithm used for GPU fast implementation

input: x

1. $\tilde{U}_0^0 = FFT(x)$
2. $\tilde{U}_0^1 = \hat{\tilde{\phi}}_J \odot \tilde{U}_0^0$
3. $\tilde{U}_J^1 = \text{periodize}(\tilde{U}_0^1, J)$
4. $S_J^0 x = iFFT(\tilde{U}_J^1)$
5. **for** $\lambda_1 = (j_1, \theta_1)$
   (a) $\tilde{U}_0^1 = \hat{\tilde{\psi}}_{\lambda_1} \odot \tilde{U}_0^0$
   (b) $\tilde{U}_{j_1}^1 = \text{periodize}(\tilde{U}_0^1, j_1)$
   (c) $\tilde{U}_{j_1}^1 = iFFT(\tilde{U}_{j_1}^1)$
   (d) $\tilde{U}_{j_1}^1 = |\tilde{U}_{j_1}^1|$
   (e) $\tilde{U}_{j_1}^1 = FFT(\tilde{U}_{j_1}^1)$
   (f) $\tilde{U}_{j_1}^2 = \hat{\tilde{\phi}}_J \odot \tilde{U}_{j_1}^1$
   (g) $\tilde{U}_J^2 = \text{periodize}(\tilde{U}_{j_1}^2, J)$
   (h) $S_J^1 x[\lambda_1] = iFFT(\tilde{U}_J^2)$
   (i) **for** $\lambda_2 = (j_2, \theta_2)$
      i. $\tilde{U}_{j_2}^2 = \hat{\tilde{\psi}}_{\lambda_2} \odot \tilde{U}_{j_1}^1$
      ii. $\tilde{U}_{j_2}^2 = \text{periodize}(\tilde{U}_{j_2}^2, j_2)$
      iii. $\tilde{U}_{j_2}^2 = iFFT(\tilde{U}_{j_2}^2)$
      iv. $\tilde{U}_{j_2}^2 = |\tilde{U}_{j_2}^2|$
      v. $\tilde{U}_{j_2}^2 = FFT(\tilde{U}_{j_2}^2)$
      vi. $\tilde{U}_{j_2}^2 = \hat{\tilde{\phi}}_J \odot \tilde{U}_{j_2}^1$
      vii. $\tilde{U}_J^2 = \text{periodize}(\tilde{U}_{j_2}^2, J)$
      viii. $S_J^2 x[\lambda_1, \lambda_2] = iFFT(\tilde{U}_J^2)$

output: $Sx = \{S_J^0 x, S_J^1 x, S_J^2 x\}$

yielded excellent numerical results [15] 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 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, as we showed in Section 3.3. CNNs fill in this gap, thus we consider the use of deep neural networks utilizing generic scattering representations in order to reduce more complex variabilities than geometric ones.

Recent works [16, 71] 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. [71] motivates that to each layer corresponds 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 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: $\mathbb{R}^2 \ltimes SO_2$. As stated in Subsection 3.1, 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 because they are highly dimensional and there is no known algorithm to obtain them ; 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.

A major strength of using a fixed and predefined Scattering Network with a CNN is that the operators of the first layers perform combinations of the Scattering coefficients. It implies that the operations performed by the filtering of the first layer of the CNN can be explicitly written with a meaning of the weights, and thus mathematically analysed to a certain extent.

4.2.2 Deep Hybrid CNNs on ILSVRC2012

In this section, we consider cascading the scattering transform with a deep CNN architecture, such as Resnet [120, 46]. 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

Method	Top 1	Top 5	#params
AlexNet	56.9	80.1	60M
VGG-16	68.5	88.7	138M
Scat+ResNet-10	68.7	88.6	12.8M
ResNet-18	68.9	88.8	11.7M
ResNet-200	78.3	94.2	64.7M

Table 4.2: 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 analogous amounts of parameters while learning parameters at a spatial resolution of 28×28

Stage	Output size	Stage details
scattering	28×28	$J = 3, 651$ channels
conv1	28×28	[256]
conv2	28×28	$\begin{bmatrix} 256 \\ 256 \end{bmatrix} \times 2$
conv3	14×14	$\begin{bmatrix} 512 \\ 512 \end{bmatrix} \times 2$
avg-pool	1×1	$[14 \times 14]$

Table 4.3: Structure of Scattering and Resnet-10 used in Imagenet experiments. Taking the convention of [120] we describe the convolution size and channels in the "Stage details".

5 for our model versus 11.1%), while utilizing fewer layers. We take the Resnet-18 [120, 46] 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). The ResNet-18 typically has 4 residual stages of 2 blocks each which gradually decrease the spatial resolution [120]. Since we utilize the scattering as a first stage we remove two blocks from our model. The network is described in Table 4.3.

The accuracy is reported in Table 4.2 and compared to other CNNs. 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.

Training procedure

We describe our training pipeline, which is similar to [120]. We trained our network for 90 epochs to minimize the standard cross entropy loss, using SGD

Method	Accuracy	Type
Separable Roto-translation Scattering + OLS + Gaussian SVM	82.3	Unsup.
ExemplarCNN [34]	84.3	Unsup.
DCGAN [89]	82.8	Unsup.
Scat + FCs	84.7	Unsup.
Scat + ResNet	93.1	Sup.
Highway networks [108]	92.4	Sup.
All-CNN [106]	92.8	Sup.
WRN 16-8 [120]	95.7	Sup.
WRN 28-10 [120]	96.0	Sup.

Table 4.4: 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.

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 dropped off by 0.1 at epochs 30, 60, 80. During the training process, each image is randomly rescaled, cropped, and flipped as in [46]. The final crop size is 224×224. At testing, we rescale the image to a size of 256, and extract a center crop of size 224×224.

4.2.3 Hybrid Representations on CIFAR-10

We again consider the popular CIFAR-10 dataset consisting of colored images composed of $5\ 10^4$ images for training, and $1\ 10^4$ images for testing divided into 10 classes. We perform two experiments, the first with a cascade of fully connected layers, that allows us to evaluate the scattering transform as an unsupervised representation. In a second experiment, we again 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.

In the unsupervised comparison we consider the task of classification using only unsupervised features. 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 features, i.e. 84.7% accuracy on CIFAR-10. This approach outperforms all methods utilizing learned and not learned unsupervised features further demonstrating the discriminative power of the scattering network representation.

In the case of the supervised task 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) [120]. Specifically we modify the WRN of 16 layers which consists of 4 convolutional stages. De-

noting the widening factor, k, after the scattering output we use a first stage of $32 \times k$. We add intermediate 1×1 layers to increase the effective depth, without increasing too much the number of parameters. Finally we apply a dropout of 0.2 as specified in [120]. 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 [46] and the wide resnet [120]. 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: we do believe there are regularization techniques, normalization techniques, and data augmentation techniques which can be designed specifically for the scattering networks.

Training procedure

We follow the training procedure prescribed in [120] 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 case. We utilize batch normalization techniques at all layers which leads to a better conditioning of the optimization [49]. Table 4.4 reports the accuracies in the unsupervised and supervised settings and compares them to other approaches.

4.2.4 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 CIFAR10 and the STL10 dataset.

4.2.4.1 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, that are split uniformly amongst the 10 classes.

We use as a baseline the Wide ResNet [120] 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 Table 4.5 of [120]. 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 4.5. Unlike the baseline, referred from here-on as WRN 16-8, our archi-

Stage	Output size	Stage details
scattering	8×8, 24×24	$J = 2$
conv1	8×8, 24×24	$16 \times k, 32 \times k$
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 4.5: Structure of Scattering and Wide ResNet hybrid used in small sample experiments. Network width is determined by factor k. For sizes and stage details if settings vary we list CIFAR10 and then the STL10 network information. All convolutions are of size 3×3 and the channel width is shown in brackets for both the network applied to STL10 and CIFAR10. For CIFAR10 we use $n = 2$ and for the larger STL10 we use $n = 4$.

Method	100	500	1000
WRN 16-8	34.7±0.8	46.5±1.4	60.0±1.8
Scat + WRN 12-8	38.9±1.2	54.7±0.6	62.0±1.1

Table 4.6: 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.

tecture has 12 layers and equivalent width, while keeping the spatial resolution constant through all stages prior to the final average pooling.

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, 100 for the 1000, 500, and 100 sample case respectively. For example the default schedule of 60, 120, 160 becomes 600, 1200, 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 4.6 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 baseline, particularly when the sample size is smaller. This is not surprising as we incorporate a geometric prior in the representation.

4.2.4.2 STL-10

The SLT-10 dataset consists of colored images of size 96×96, with only $5\,10^3$ labeled images in the training set divided equally in 10 classes and $8\,10^3$ images

Method	Accuracy	Type
Scat + WRN 19-8	76.0 ± 0.6	Supervised
CNN [111]	70.1 ± 0.6	Supervised
Exemplar CNN [34]	75.4±0.3	Unsupervised
Hierarchical Matching Pursuit (HMP) [9]	64.5±0.1	Unsupervised
Convolutional K-means Network	60.1 ± 0.1	Unsupervised

Table 4.7: 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.

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 100 thousand 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 4.5 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 standard 10 folds procedure which takes 1000 training images. The averaged result is reported in Table 4.7.

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 [34, 111] 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 [47] who reported 81.3%, 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 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, for downstream discriminative tasks, may still not outperform supervised methods and pre-defined representations. One possible explanation is that in the case of natural images, learning in an unsupervised way more complex variabilities than geometric ones (e.g the roto-translation group), might be ill-posed.

4.3 Shared Local Encoder

In the previous section, we introduced a Hybrid Network based on the Scattering Networks. This section contends the notion of global representation for image classification, e.g. representations that potentially combine any neighborhoods of an image in order to build a discriminative representation. We first 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 mix global information. For example, at depth 2 of [56], the effective spatial support of the corresponding filter is already 32 pixels (out of 224). The specific representations derived from CNNs trained on large scale image recognition are often used as representations in other computer vision tasks or datasets [117, 122, 118].

On the other hand prior to 2012 local encoding methods led to state of the art performance on large scale visual recognition tasks [94]. In these approaches local neighborhoods of an image were encoded using methods such as SIFT descriptors [65], HOG [30], and wavelet transforms [98]. They were also often combined with an unsupervised encoding, such as sparse coding [10] or Fisher Vectors (FVs)[94]. Indeed, many works in classical image processing or classification [55, 10, 95, 88] suggest that the local encoding of an image permit to describe efficiently an image. Additionally for some algorithms that rely on local neighbourhoods, the use of local descriptors is essential [65]. 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, on large scale classification, this approach was surpassed by fully supervised learned methods [56].

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 [88]. 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 2.2.1, 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. Then, 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.

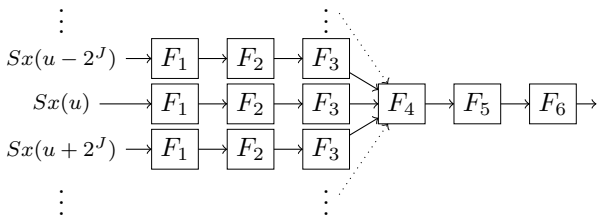


Figure 4.3.1: Architecture of the SLE, which is a cascade of 3 1×1 convolutions followed by 3 fully connected layers. The ReLU non-linearities are included inside the F_i blocks for clarity.

Method	Top 1	Top 5
FV + FC	55.6	78.4
FV + SVM	54.3	74.3
AlexNet	56.9	80.1
Scat + SLE	57.0	79.6

Table 4.8: Top 1 and Top 5 percentage accuracy reported from one single crop on ILSVRC2012. We compare to other local encoding methods, and SLE outperforms them. [88] 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 as well structured. We further explain and investigate this first layer in Subsection 4.3.2. Unlike standard CNNs, there is no linear combinations 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. We note it is not possible to use an average pooling as it will remove any spatial localization information for the supervised task: extra-convolutions could have permitted to discriminate spatial information (like wavelets), but it is impossible to implement them with 1×1 convolutions.

In the next Subsection 4.3.1, we investigate empirically this supervised SLE trained on the ILSVRC2012 dataset.

4.3.1 Encoding scattering coefficients

We use an architecture which consists of a cascade of a scattering network, a SLE f, followed by fully connected layers. Figure 4.3.1 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 in the channel dimension. f is implemented as 3 layers of 1×1 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 [49] followed by a ReLU [56] non-linearity. We compute the mean and variance of the scattering coefficients on the whole Imagenet, and standardized each spatial scattering coefficients with it.

Table 4.8 reports our numerical accuracies obtained with a single crop at testing, compared with local encoding methods, and the AlexNet that 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 AlexNet [56]. In term of architecture, our hybrid model is analogous, and comparable to that of [94, 95, 88], for which SIFT features are extracted followed by FV [95] encoding. Observe the FV is an unsupervised encoding compared to our supervised encoding. Two approaches are then used: either the spatial localization is handled either by a Spatial Pyramid Pooling [57], which is then fed to a linear SVM, either 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 quantified. 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 the scattering transform is not oversampled) via random cropping during training, allowing to build an invariant to small translations. In Top 1, [94] and [88] obtain respectively 44.4% and 45.7%. Our method brings a substantial improvement of 1.4% and 2.7% respectively.

The BVLC AlexNet [1] 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 context information is only necessary at the very last layers, to reach AlexNet accuracy.

Transfer learning ability

We study briefly the local SLE, which has only a spatial extent of 16×16, as a generic local image descriptor. We use the Caltech-101 benchmark which is a dataset of 9144 image and 102 classes. We followed the standard protocol for evaluation [10] 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, F_3 brings respectively an absolute improvement of 13.7, 17.3, 20.1. The accuracy of the final SLE descriptor is thus 82.9 ± 0.4, similar to that reported for the final AlexNet final layer in [122] and sparse coding with SIFT [10]. However in both cases spatial variability is removed, either by

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

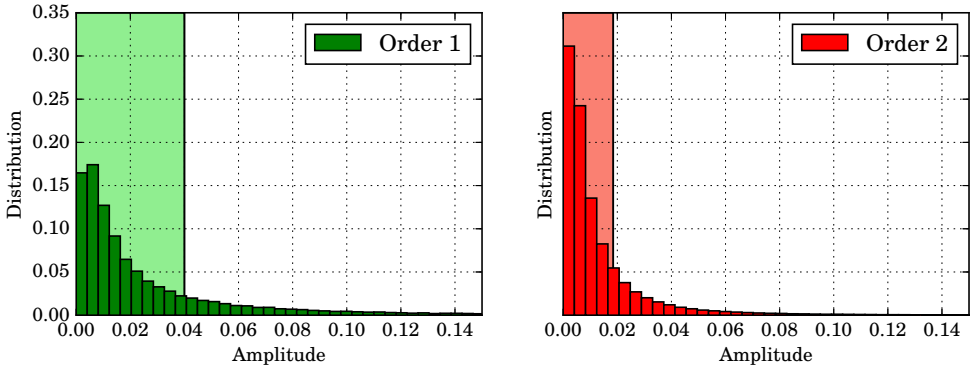


Figure 4.3.2: Histogram of $\hat{F}_1$ amplitude for first and second order coefficients. The vertical lines indicate a threshold that is used to sparsify $\hat{F}_1$. Best viewed in color.

Spatial Pyramid Pooling [57], or the cascade of large filters. By contrasts the concatenation of SLE descriptors are completely local.

Training procedure

We describe our training pipeline, which is similar to [120]. 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 dropped off by 0.1 at epochs 30, 50, 70, 80. During the training process, each image is randomly rescaled, cropped, and flipped as in [46]. The final crop size is 224×224. At testing, we rescale the image to a size of 256, and extract a center crop of size 224×224.

4.3.2 Interpreting SLE's first layer

Finding structure in the kernel of the layers of depth less than 2 [115, 122] is a complex task, and few empirical analyses exist that shed light on the structure of deeper layers, for example the work we did in Chapter 5 or Chapter 6. A scattering transform with scale J can be interpreted as a CNN with depth J, as we see in Subsection 2.2.3, 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 analyze F_1 and show that it builds explicitly invariance to local rotations, yet also that the Fourier bases associated to rotation are a natural bases of our operator. It is a promising direction to understand the nature of the two next layers.

We first establish some mathematical notions linked to the roto-translation group that we use in our analysis. For a given input image $x \in L^2(\mathbb{R}^2)$, let us recall we denote in Subsection 3.1:

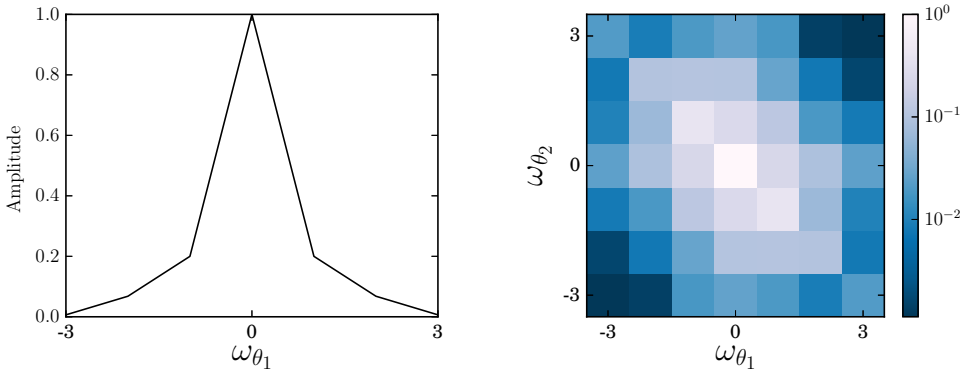


Figure 4.3.3: Energy $\Omega_1\{F\}$ (left) and $\Omega_2\{F\}$ (right) from Eq. (4.3.11) and Eq. (4.3.12) for given angular frequencies. Best viewed in color.

$$\mathcal{L}_{r_\theta}.x(u) \triangleq x(r_{-\theta}(u)) \tag{4.3.1}$$

which is 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:

$$\begin{aligned} S_1(\mathcal{L}_{r_\theta}.x)(\theta_1, u) &= S_1 x(\theta_1 - \theta, r_{-\theta}u) && (4.3.2)\\ &\triangleq \mathcal{L}_{r_\theta}(S_1 x)(\theta_1, u) && (4.3.3)\end{aligned}$$

and

$$\begin{aligned} S_2(\mathcal{L}_{r_\theta}x)(\theta_1, \theta_2, u) &= S_2 x(\theta_1 - \theta, \theta_2 - \theta, r_{-\theta}u)\\ &\triangleq \mathcal{L}_{r_\theta}(S_2 x)(\theta_1, \theta_2, u)\end{aligned}$$

The unitary representation framework [109] permits the building of a Fourier transform on compact group, like rotations. It is even possible to build a scattering transform on the roto-translation group [101]. Fourier analysis permits the measurement of the smoothness of the operator and, in the case of CNN operators, it is a natural basis.

Besides, in the case of the second order coefficients, (θ_1, θ_2) is covariant with rotations, but $\alpha = \theta_2 - \theta_1$ is an invariant to rotation that correspond to a relative rotation. Thus, for $S_2 x$, the natural set of coordinates that gives a rotational invariant for angles is in fact given by:

$$\tilde{S}_2 x(u, \theta_1, \alpha) \triangleq S_2 x(u, \theta_1, \theta_1 + \alpha) \tag{4.3.4}$$

where α corresponds to a relative angle, which naturally leads to:

$$\tilde{S}_2(\mathcal{L}_{r_\theta}x)(u,\theta_1,\alpha) = \tilde{S}_2x(r_{-\theta}u,\theta_1-\theta,\alpha) \quad (4.3.5)$$
$$\triangleq \mathcal{L}_{r_\theta}(\tilde{S}_2x)(u,\theta_1,\alpha) \quad (4.3.6)$$

We can now numerically analyze 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 $\{F_1^0S_0x, F_1^1S_1x, F_1^2S_2x\}$ the restrictions of F_1Sx to the order 0,1,2 scattering coefficients respectively. In other words,

$$F_1Sx = F_1^0S_0x + F_1^1S_1x + F_1^2S_2x \quad (4.3.7)$$

Let $1 \leq k \leq K$ an index of the feature channel and $1 \leq c \leq 3$ the color index. In this case, $F_1^0S_0x$ is the weights associated to the smoothing S_0x. $F_1^1S_1x$ depends only (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, e.g.:

$$\hat{F}_1^1(k,c,j_1,\omega_{\theta_1}) = \int F_1^1(k,c,j_1,\theta_1)e^{-i\omega_{\theta_1}\theta_1}d\theta_1 \quad (4.3.8)$$

and

$$\hat{F}_1^2(k,c,j_1,j_2,\omega_{\theta_1},\omega_{\theta_2}) = \int F_1^2(k,c,j_1,j_2,\theta_1,\theta_2)e^{-i\omega_{\theta_1}\theta_1-i\omega_{\theta_2}\theta_2}d\theta_1d\theta_2 \quad (4.3.9)$$

These operators 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 the normalization of the Scattering Transform is chosen such that the transform is unitary as well, which means:

$$\forall k, F_1^0(k)^2 + \sum_{c,j_1,\theta_1} F_1^1(k,c,j_1,\theta_1)^2 + \sum_{c,j_1,j_2,\theta_1,\theta_2} F_1^2(k,c,j_1,j_2,\theta_1,\theta_2)^2 = 1 \quad (4.3.10)$$

Figure 4.3.2 shows the distribution of the amplitude of $\hat{F}_1^1$,$\hat{F}_1^2$. We observe that the distribution is shaped as a Laplace distribution, which is an indicator of sparsity, which gives an insight for the following experiment.

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 $\hat{F}_1^1$ by:

$$1_{|\hat{F}_1^1|>\epsilon}\hat{F}_1^1$$

and $\hat{F}_1^2$ by:

$$1_{|\hat{F}_1^2|>\epsilon}\hat{F}_1^2$$

We select an ϵ that sets 80% of the coefficients to 0, which is indicated in Figure 4.3.2. *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:

$$\Omega_1\{F\}(\omega_1) \triangleq \sum_{k,j_1,c} |\hat{F}_1^1(k,c,j_1,\omega_{\theta_1})|^2 \tag{4.3.11}$$

and

$$\Omega_2\{F\}(\omega_{\theta_1},\omega_{\theta_2}) \triangleq \sum_{k,c,j_1,j_2} |\hat{F}_1^2(k,c,j_1,j_2,\omega_{\theta_1},\omega_{\theta_2})|^2 \tag{4.3.12}$$

They correspond to the energy propagated by F_1 for a given frequency, and permit to quantify the smoothness of our first layer operator w.r.t. the angular variables. Figure 4.3.3 shows variation of $\Omega_1\{F\}$ and $\Omega_2\{F\}$ along 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. Observe also that the energy $\Omega_2\{F\}$ seems concentrated in an ellipsoid with main axis $\omega_{\theta_1} = -\omega_{\theta_2}$, which indicates that in the network builds a stronger invariance along relative local rotations given by an angle $\alpha = \theta_2 - \theta_1$.

Chapter 5

Empirical Analysis of CNN Properties

This chapter is dedicated to report experimental results on CNNs that are not predicted by the theory, or intentionally due to the training scheme. The previous section introduces a class of deep networks for which the first layers are fully initialized via wavelets. This leverages the interpretability of those layers, but the properties of the next layers are not yet fully understood. Several works suggest that they build a representation that can be transferred on different datasets, which implies that they build representation that captures generic properties of images and are not merely memorizing samples or patterns [117].

Often, their ability to generalize or approximate the objective class function is justified via the universal approximation theorem which states that a 2 layers deep network can approximate any regular function under weak conditions [116]. However this statement is not enough. For example, a minimal depth [5] larger than 2 is necessary to obtain good performances: in this chapter, we observe that progressive mechanisms permit the building of a good representation. The two mains contributions of this chapter are the following: first, we introduce a simplified architecture that leads to state-of-the-art results and secondly, we observe a progressive dimensionality reduction across layers, that we study in details. Let us describe the contributions linked respectively to the architecture, and representation aspects.

For this study, it is necessary to specify a precise class of deep networks that we will study numerically, in order to avoid side effects that are due to specific architectures hints or optimization. The cascade of a deep networks can use complex non-linear submodules, such as max-pooling [10], local contrast normalization [56], residual connections [46]... We show it is possible to obtain state-of-the-art results on CIFAR dataset, by simply using only few non-linearities, linear CNNs operators with identical size and regularization at each depth and few regularization. In particular, we show that a max-pooling operator is not necessary to obtain good performances, which means that the

invariance to translation can be learned from the data via the linear operator.

Simplifying a CNN architecture permits to varying hyper parameters. For example, increasing the width of the network leads to orders of accuracy improvement. This is surprising, as the number of parameters increase and the network should be prone to a large overfitting. Another aspect we study is the degree of the non-linearity: "more non-linear" does better is a common affirmation in deep learning talks. Contrarily to this claim, we show that a point-wise non-linearity applied to a layer can be actually linear on a fraction of the coefficients. We observe other properties concerning the non-linearity.

In a classification task, the intrinsic dimension of the classes is a quite low-dimensional structure in comparison with the original dimension of the signals, thus a classification task requires estimating a (often non-linear) projection onto a low-dimensional space. It implies that the space of representation is contracted. Up to renormalization, a linear operator is always a contractive operator, yet this is not necessary the case of a non-linear operator. In particular, we show it is possible to use point-wise non-linearity that are neither continuous nor contractive: it implies that the intermediary CNN layers are not contractive on the whole space.

Furthermore, we propose a taxonomy of the classification boundary. In particular, we show that the intra-class variance and distances of the CNNs representation at each layer are progressively reduced. It means that the latters are decreasing with depth, until the final layer for which the representations of different classes are linearly separable.

This leads us to the notion of *Local Support Vectors*, that consist in points that permits to define locally the complexity of the boundary of classification. We build a measure of the contraction and separation due to those vectors, that indicates a progressive contraction of the space. In particular, this suggests that the representation is progressively embedded in a lower dimensional space, which explains why a nearest-neighbor is progressively improves with depth.

This chapter is divided as follow. First, in Section 5.1, we define a simplified framework that leads to state-of-the-art results on CIFAR10. Then, we numerically study it in Section 5.2.

5.1 Simplifying a state-of-the-art CNN architecture

We introduce a state-of-the-art pipeline for CIFAR10 classification that uses a minimal number of ad-hoc engineered tricks, which depends on two hyper-parameters: its width and a non-linearity. We demonstrate that this framework is flexible and simple, as we can vary those hyper parameters and relate them to the final accuracy of the network.

In particular, adjusting the width of each CNN operators, and thus the size of the representation, permits adjusting a trade-off between performance accuracies and speed of computations. In all our practical cases, we report that

increasing the width systematically improves the final test performances.

5.1.1 Architecture

We describe the architecture that we used during all our experiment, with the datasets CIFAR10 and CIFAR100. It will depend only on $K \in \mathbb{N}$, which is the width of our network, and ρ a non-linear function. Our deep network consists of the cascade of 13 convolutional layers W_j with non-linearity ρ. A convolutional layer is by defined as an operator of l^2 which specified by an index of layers Λ_j, such that $\forall p, \forall \lambda \leq \Lambda_j$:

$$W_j x_j[p, \lambda] = \sum_{\tilde{\lambda}} x_j \star k_{j,\lambda,\tilde{\lambda}}[2^{d_j} p] \tag{5.1.1}$$

where p is a discrete spatial index and $k_{j,\lambda,\tilde{\lambda}}$ is convolutional kernel, and $d_j \in \{0, 1\}$, in order to apply a downsampling. The spatial support of the kernel is 3×3, and except for the first layer, the number of input and output layers is fixed equal to $\Lambda_j = K, \forall 1 < j < 13$, which is typically a power of 2. We did not learn any biases in the convolutional layers, however we subtract the empirical mean $Ex_j \in \mathbb{R}^{\Lambda_j}$ from our feature maps, which is estimated on all the dataset via the batch normalization technique [49].

For computational speed-up, we apply a spatial down-sampling of 2 at the output of the layers 6 and 10. Figure 5.1.1 describes our network, which can be formally summarized for an input x, via $x_0 = x$, and a cascade of blocks:

$$x_{j+1} = \rho W_j (x_j - E x_j) \tag{5.1.2}$$

In this case we are in a similar setting as [72], which proves that if W_j is unitary then for any depths $j \leq J$ the network preserves the energy of the input signal and is non-expansive. The output x_J of the final convolutional layer is linearly and globally spatially averaged by A:

$$A x_J[\lambda_J] = \sum_p x_J[p, \lambda_J] \tag{5.1.3}$$

, and then reduced to the number of classes C of the problem by a projection $L : \mathbb{R}^K \to \mathbb{R}^C$.

Again, no extra-max pooling was involved, which permits to remove extra-instabilities that could have been introduced by this module and should have been removed by the linear operators. Furthermore and again, all the trainable affine biases have been removed, which means the biases are computed via an estimator of the expected values of each layer.

CIFAR datasets are preprocessed using a standard procedure of whitening. The number of parameters used by our network with CIFAR10 is $9 \times (3K + 12K^2) + 10K$. To get our best accuracy, we used $K = 512$ which corresponds roughly to 28M parameters, that lead to 95.4% and 79.6% accuracies on CIFAR10 and CIFAR100 respectively, which is a competitive performance

Methods	Depth	#params	CIFAR10	CIFAR100
Ours	13	28M	95.4	79.6
SGDR	28	150M	96.2	82.3
RoR	58	13M	96.2	80.3
WRN	28	37M	95.8	80.0
All-CNN	9	1.3M	92.8	66.3

Table 5.1: Accuracy on CIFAR10 and CIFAR100 for state-of-the-arts supervised deep networks. Depth and number of parameters are reported to perform a fair comparison.

according to Table 5.1. Thus, we are in a state-of-the-art setting to perform an analysis of the features learned.

Surprisingly, increasing the number of parameters significantly increase the performances instead of reducing them, which is consistent with the facts that neural networks are not prone to overfitting [123]. We show that increasing K increases the classification accuracy of the network. In particular, it permits to work on a network with a small number of parameters, before scaling up the network with an expecting accuracy for a given size. [120] reports also this observation, which is not obvious since increasing K increases by K^2 the number of parameters and could lead to a severe overfitting. Besides, since a final dimensionality reduction must occur at the last layer, one could expect that the intermediate layers might have a small number of feature maps. Figure 5.1.2 reports the numerical accuracy respectively on CIFAR10 and CIFAR100, with respect to K. Setting $K = 512$ leads to 95.4% and 79.6% accuracy respectively on CIFAR10 and CIFAR100, while $K = 16$ leads to 79.8% and 30.6% accuracy on CIFAR10 and CIFAR100 respectively. It is not clear if the reason of this improvement is the optimization or if it is a structural reason. Nevertheless, it indicates that increasing the number of feature maps is a simple way to improve the network accuracy.

Training procedure

We trained our network via a SGD with momentum 0.9 to minimize the standard negative cross-entropy. We used a batch size of 128, and the training lasts $1.2\,10^5$ iterations. We used an initial learning rate of 0.25, that we divided by two every $1\,10^4$ batches. To avoid overfitting, we apply 4 regularizations. First, a weight decay of $2\,10^{-4}$ that corresponds to a l^2 regularization. Then, we used dropout every two layers, starting at the second layer, that randomly sets 40% of the activation layers to 0: this is our main trick to achieve good performances. Thirdly, we used spatial batch normalization regularization that is supposed to remove instabilities during the training, as developed in [49]. Finally, we applied standard random flipping and cropping techniques as data augmentation. Observe that we did not use any bias, simply removing the mean and did not use any non-linear pooling. Our architecture is thus kept as simple

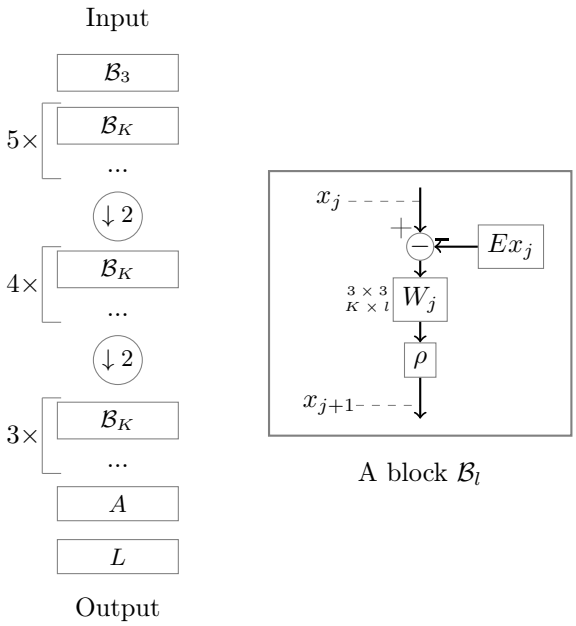


Figure 5.1.1: Schematic representation of our architecture. Our network is a cascade of block $\mathcal{B}_l$, l being the input size of the convolutional operator, followed by an averaging A and a projection L.

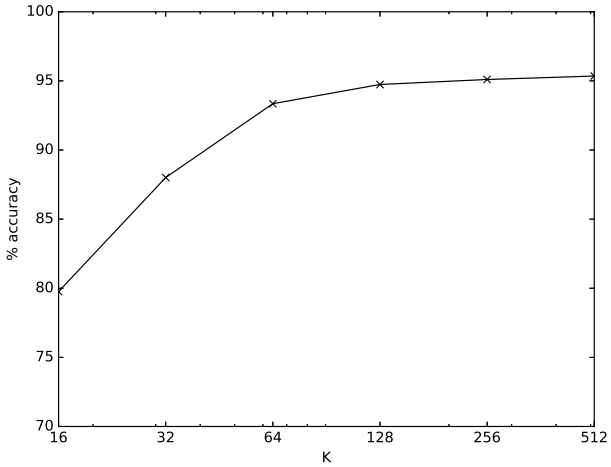
as possible, as in [106] but it only depends only on a few hyper parameters: its width and the non-linearity. Without any contrary mentions, we used $\rho = \text{ReLU}$ since it has heuristically been shown to achieve better performances. The first layer will always have a ReLU non-linearity.

5.1.2 The role of the non-linearity

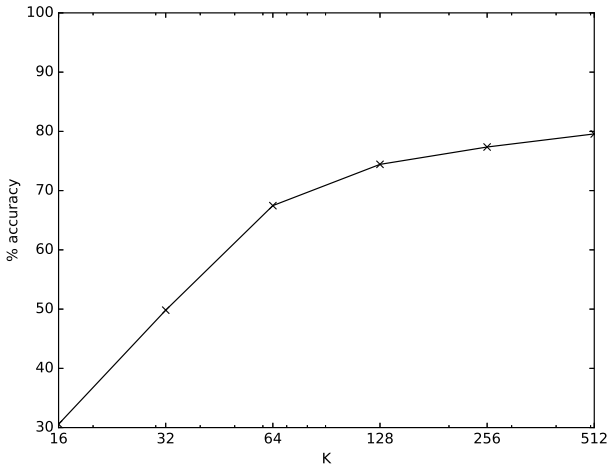
Contraction phenomenon is a necessary step to explain the tremendous dimensionality reduction of the space that occurs. A network cannot be purely linear, since most of the image classification problems are not linearly separated: indeed a linear operator can only contract along straight lines. Should ρ also be a contracting operator, as suggested in [71]? We study specifically the point-wise non linearity ρ in a CNN and its necessary conditions to reach good classification accuracy.

5.1.2.1 Unneccesity to contract via ρ

In this subsection, we review several non-linearities and their properties. We discuss the contraction which is built by a non-expansive non-linearity, and the properties of the modulus combined with an analytical filter. Our conclusion is



(a) Accuracy when varying K on CIFAR10 dataset, the axis of K is in log scale.



(b) Accuracy when varying K on CIFAR100 dataset, the axis of K is in log scale.

Figure 5.1.2: Effect of K on the classification accuracy

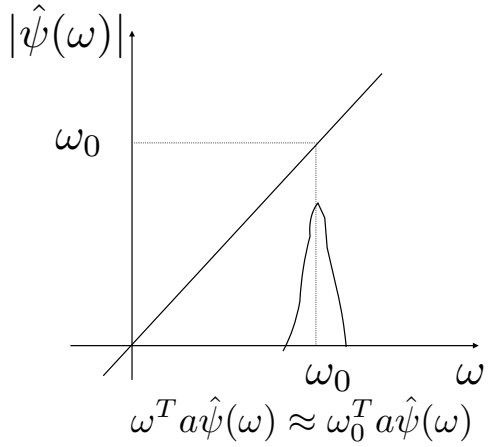


Figure 5.1.3: Localization in Fourier implies that a translations results in a phase multiplication, up to approximation terms

that none of these properties are necessary to obtain good classification accuracies.

First, we explain the contraction property. Since the AlexNet [56] , non-linearity is often chosen to be a $\mathrm{ReLU}(x) = \max(0, x)$. This is a non-expansive function, e.g.:

$$|\mathrm{ReLU}(x) - \mathrm{ReLU}(y)| \leq |x - y| \tag{5.1.4}$$

, which implies it is also continuous. Adding the negative part [7, 121], it is possible to avoid any loss of information. Consequently, a cascade of linear operators of norm less than 1 and this non-linearity is non-expansive which is a convenient property to reduce or maintain the volume of the data. For example, in the framework of [71], the non-linear operator is simply viewed as a way to contract the space.

Now, we explain how the modulus non-linearities in complex networks have been suggested to remove the phase of a signal, which is in several frameworks variability due to translation[70, 71, 15, 13]. For example, in Fourier, for any translations a, one has:

$$\widehat{\mathcal{L}_a x}(\omega) = e^{i\omega^T a}\hat{x}(\omega) \tag{5.1.5}$$

For instance, if the linear operator consists in a wavelet transform with appropriate mother wavelet [68], then the spectrum of each convolution with a wavelet is localized in Fourier, and in this case, a modulus smoothens the envelop of the signal. Mathematically, we mean that the filter energy should be concentrated in Fourier around ω_0 with radius η_0, which is expressed by:

$$\exists \eta_0 > 0, \exists \omega_0, \epsilon \geq 0, \int_{\|\omega-\omega_0\| \geq \eta_0} |\hat{\psi}(\omega)|^2 d\omega \leq \epsilon \tag{5.1.6}$$

then, in this case:

$$\begin{aligned}
\int |(e^{i\omega^T a} - e^{i\omega_0^T a})\hat{\psi}(\omega)|^2 d\omega &\leq 2\epsilon + \int_{\|\omega-\omega_0\|<\eta_0} |(e^{i\omega^T a} - e^{i\omega_0^T a})\hat{\psi}(\omega)|^2 d\omega \\
&= 2\epsilon + \int_{\|\omega-\omega_0\|<\eta_0} |\sin(\frac{a^T(\omega-\omega_0)}{2})\hat{\psi}(\omega)|^2 d\omega \\
&\leq 2\epsilon + \frac{\|a\|^2\|\eta_0\|^2}{4} \qquad (5.1.7)
\end{aligned}$$

The remaining energy is thus bounded by the vanishing energy of the wavelet out of the η_0-ball, and the magnitude of a. This property is illustrated on Figure 5.1.3. It means as well that if $\|\omega_0\| > \eta_0$, then the wavelet is approximatively analytic. An informal way to understand this equation is to use the fact that the infinitesimal generator of the translation is the derivation operator, indeed, observe that if x is $\mathcal{C}^\infty$:

$$\mathcal{L}_a x(u) = x(u+a) = \sum_{n \geq 0} \frac{x^{(n)}(u)(a,...,a)}{n!} \tag{5.1.8}$$

where $x^{(n)}$ is the order n differential, which is equivalent in Fourier to:

$$\widehat{\mathcal{L}_a x}(\omega) = \sum_{n \geq 0} \frac{(i\omega^T a)^n}{n!} \hat{x}(\omega) \tag{5.1.9}$$

Thus, again, informally:

$$\begin{aligned}
\widehat{\mathcal{L}_a(x \star \psi)}(\omega) &= e^{i\omega^T a}\hat{\psi}(\omega)\hat{x}(\omega) && (5.1.10)\\
&= \sum_{n \geq 0} \frac{(i\omega^T a)^n}{n!} \hat{\psi}(\omega)\hat{x}(\omega) && (5.1.11)\\
&\approx \sum_{n \geq 0} \frac{(i\omega_0^T a)^n}{n!} \hat{\psi}(\omega)\hat{x}(\omega) \text{ by the Figure 5.1.3} && (5.1.12)\\
&= e^{i\omega_0^T a}\hat{\psi}(\omega)\hat{x}(\omega) && (5.1.13)\\
&= e^{i\omega_0^T a}\widehat{x \star \psi}(\omega) && (5.1.14)
\end{aligned}$$

It implies that in the real domain, if a is small enough:

$$\mathcal{L}_a x \star \psi \approx e^{i\omega_0^T a} x \star \psi \tag{5.1.15}$$

Applying a modulus removes this variability. As a classical result of signal theory, observe also that an averaged rectified signal is approximately equal

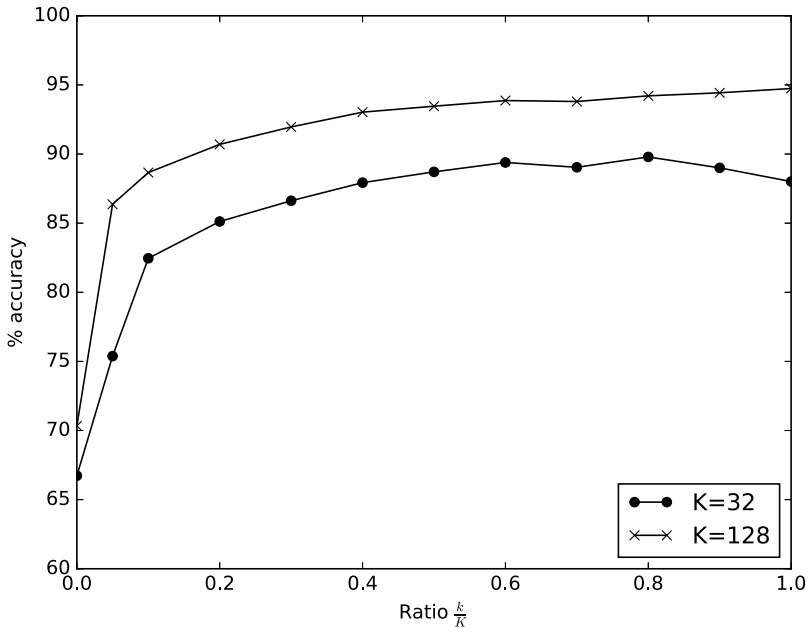


Figure 5.1.4: Accuracy when varying the degree of non-linearity $\frac{k}{K}$, reported with $K = 32$ and $K = 128$. When $k = K$, one obtains 88.0% and 94.7% respectively for $K = 32$ and $K = 128$. The maximum accuracies are then respectively 89.8% and 94.7%, which indicates that a point-wise non-linearity is not necessarily the optimal configuration.

to the average of its complex envelope [68]. Consequently, cascaded with an average pooling and an analytic wavelet, a ReLU and a modulus might have the same use.

Our conclusion is the following: experimentally, it is possible to build a deep network that leads to 89.0% accuracy on CIFAR10, with $K = 256$, with the non-linearity chosen as:

$$\rho(x) = \text{sign}(x)(\sqrt{|x|} + 0.1) \tag{5.1.16}$$

In the neighborhood of 0, this non-linearity is not continuous in 0, has an arbitrary large derivative, and preserves the sign of the signal. This shows that those three properties are not necessary to obtain good classification accuracy, and that the linear operator can handle the necessary contraction process.

5.1.2.2 Degree of non-linearity

In this subsection, we try to weaken the traditional property of point-wise non-linearity. Indeed, being non-linear is essential to ensure that the different classes can be separated, however the recent work on ResNet [46], that we introduced

in the previous chapter, suggests that it is not necessary to apply a point-wise non-linearity, thanks to identity mapping that can be interpreted as the the concatenation of a linear block (the identity) and a non-linear block. In this case, a non-linearity is applied only on a half of the feature maps. We investigate the question to understand if this property generalizes to our architecture by introducing a ReLU with a degree $\frac{k}{K}$ of non-linearity that we apply to a feature map $x(u,l)$, defined by:

$$\text{ReLU}_k^K(x)(u,l) \triangleq \begin{cases} \text{ReLU}(x(u,l)), & \text{if } l \leq k \\ x(u,l), & \text{otherwise} \end{cases} \tag{5.1.17}$$

A counter example is given by the scattering network for example, which has quite less effective non-linearity than usual deep network: $\frac{\#non-linearity}{\#kernels} = \frac{JL+2\times\frac{J(J-1)}{2}L^2}{J(1+JL+\frac{J(J-1)}{2}L^2)} \approx \frac{2}{J}$ (a typical value of J is 3) when a Deep network has usually a ratio of 1. Let us count the number of non-linearity, to compute the ratio of the non-linearity.

In the case $k = 0$, we have an almost linear network (there is the ReLU non-linearity at the first layer), and when $k = K$, it is a standard deep network with point-wise non-linearity. Figure 5.1.4 reports the numerical accuracy when we vary k, fixing K equal to 32 or 128. A linear deep network performs poorly, leading to an accuracy of roughly 70% on CIFAR10. We see that there is a plateau when $\frac{k}{K} \geq 0.6 = \frac{k_0}{K}$, and that the maximum accuracy is not necessary obtained for $k = K$. Our networks could reach 89.8% and 94.4% classification accuracy respectively for $K = 32$ and $K = 128$, whereas an almost linear network achieves 66.7%$ and 70.3% accuracies respectively.

This is an opportunity to reinterpret the non-linearity. Let τ be a cyclic translation of $\{1, ..., K\}$, which is:

$$\tau([1, ..., K]) = [K, 1, ..., K-1]$$

such that we define:

$$\tau(x)(u,l) \triangleq x(u, \tau(l)). \tag{5.1.18}$$

In this case, τ is a linear operator that translates cyclically the channels of a feature map. Observe that:

$$\text{ReLU}_k^K x = \underbrace{\tau \circ \text{ReLU}_1^K \circ ... \tau \circ \text{ReLU}_1^K}_{k \text{ times}} \tag{5.1.19}$$

In this setting, one might interpret a CNN with depth J and that uses K feature maps as a CNN of depth JK, since it is also a cascade of JK ReLU_1^K. τ might be learned as well as being fixed. It means also that if $k < K$, by cascading enough layers, it is possible to write a solution of our classification problem that uses a ReLU as a network using ReLU_K^k. For $k < k_0$, we tried to increase the depth of the deep network to recover its original accuracy since there will be as much non-linearity as in the case $k = K$, and we know there

exists an optimal solution. However, our network was not able to perform as well, which implies that there is an issue with the optimization. Restricting the non-linearity application to only one feature map could help future analysis, since it gives explicitly the coefficients that exhibits non-linear effects. This framework is intensively used in the next section.

5.2 Progressive space contraction

We now describe necessary conditions to solve a classification task. Regularity of a representation with respect to the classes is necessary to classify high-dimensional samples. Regularity means here that a supervised classifier builds a covering of the full data space with training samples via ε-balls that is small in term of volume or number of balls, yet that it still generalizes well [113]. The issue is that it is hard to track this measure.

For example, assume the data lay on a 2D or 3D manifold and are smooth with respect to the classes, then it is possible to build a classifier which is locally affine using manifold learning techniques. In particular, this implies that the euclidean metric is locally meaningful. We take a second example: when a nearest neighbor obtains good generalization properties on a test set. In this case, our problem is regular since it means again that locally the euclidean metric is meaningful and that the representation of the training set is mostly covering the data space.

We show that supervised deep networks progressively build a representation where euclidean distance becomes more meaningful. Indeed, we numerically demonstrate that the performance of local classifiers, e.g. which assign a class by giving more important weights to points of the training set that are in a neighborhood of the testing sample, progressively improves with depth. We also show that a progressive variance reduction occurs with depth.

In this section and the following, in order to save computation time, we used the network previously introduced with $K = 32$ and $\rho = \text{ReLU}$. Numerically, we however tried several parameters as a sanity check that our conclusion generalizes to any values of K, to avoid a loss in generality. With $K = 32$, the accuracy of the network is 88.0%. Increasing K to 512 should approximatively increase all the reported results by 7 absolute percents of accuracy.

Translation is one of the symmetries of the image classification problem, thus it is necessary to remove this variability, even if the features are not extracted at the final layer. In the following, we perform our experiments using:

$$\bar{x} \triangleq Ax \in \mathbb{R}^{32} \tag{5.2.1}$$

We have the following ϵ separation property, thanks to non-expansivity of averaging operators:

$$\|x - y\| \geq \|\bar{x} - \bar{y}\| \geq \epsilon \tag{5.2.2}$$

The previous equation means that a separation by ϵ of the averaged signals implies a separation by at least ϵ of any translated versions of the original signals. We denote the features at depth j of the training set by:

$$\bar{X}^j_{\text{train}} = \{\bar{x}_j, x \in \text{training}\} \tag{5.2.3}$$

and the features of depth j of the testing set by

$$\bar{X}^j_{\text{test}} = \{\bar{x}_j, x \in \text{testing}\} \tag{5.2.4}$$

For a given x, x_j or $\bar{x}_j$, we write its class $y(x)$, $y(x_j)$ or $y(\bar{x}_j)$ since there is no confusion possible.

5.2.1 Intra-class variance and distance reduction

We want to quantify the contraction of the space performed by our CNN. In this subsection, we show that the samples of a same class define a structure that progressively becomes low-dimensional. We investigate the question to understand if there is a progressive variance reduction, layer per layer. First, we check wether a linear dimensionality reduction is implemented by our CNN. To this end, we apply a PCA on the features $\bar{X}^j_{\text{train}}$ belonging to the same class at each depth j. As a normalization, the features at each depth j were globally standardized. In other words, the data at each depth are in the l^2 balls of radius 32. Remember that in our case, $\bar{x}_j \in \mathbb{R}^{32}$.

Figure 5.2.1 represents the cumulated variances of the $K = 32$ principal component axis of a given class at different depth j. The obtained diagram and conclusions are not specific to this class. The accumulated variance indicates the proportion of energy that is explained by the first axis. The slope of a curve is an indicator of the dimension of the classes: as a plateau is reached, the last components are not useful to represent the class for classifiers based on l^2 distances.

The first observation is that the variance seems to be uniformly reduced with depth. However, certain plots of successive depths are almost indisguishable: it indicates that almost no variance reduction has been performed. Except for the last layer, the decay of the 20 last eigenvalues is slow: this is not surprising since nothing indicates that the dimension should be reduced and small variance coefficients might be important for the classification task. The very last layer exhibits a large variance reduction and is low-dimensional, yet this is logical since by construction, the final features should be linearly separable and be in space of dimension 10.

We then focus on the contraction of the intra-class distances. As a cascade of non-expansive operators, a deep network is also non-expansive up to a multiplicative constant. In particular, the intra-class distances should be smaller. We should observe a progressive reduction of the volume of the space, yet does it occur class per class as well? Under this hypothesis, in a similar fashion as [39], we study the average intra-class distances. As a normalization, the features $\bar{X}^j_{\text{train}}$ at depth j are standardized over the dataset. On CIFAR10 (where each

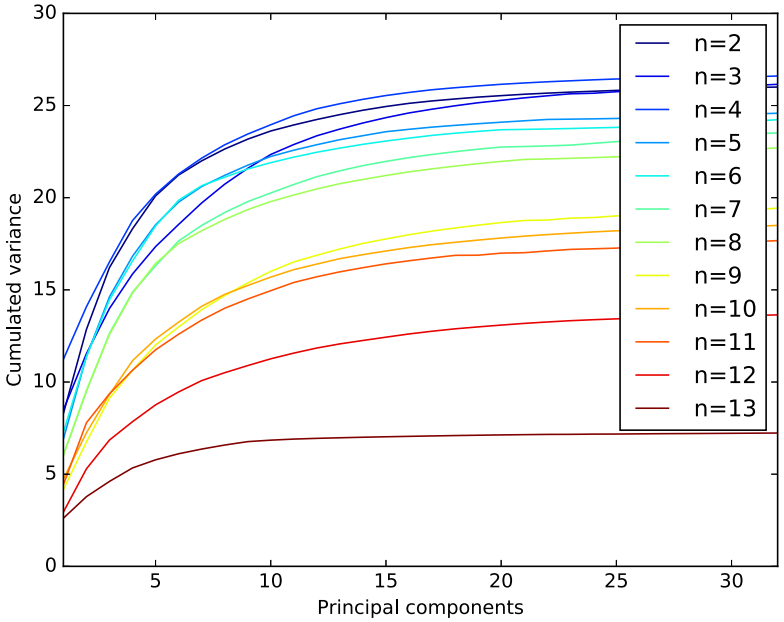


Figure 5.2.1: Cumulated variances of the principal component of a given class at different depths j, for a network trained on CIFAR10 with $K = 32$. In general, one observes a reduction of variance with depth. Best viewed in color.

of the 10 classes has 5000 samples) we compute an estimation of the average distances of the intra-class samples of the features $\bar{X}^j_{\text{train}}$ at depth j for the class c:

$$\frac{1}{5000^2} \sum_{\substack{\bar{x}_j \in \bar{X}^j_{\text{train}} \\ y(x_j)=c}} \sum_{\substack{\bar{x'}_j \in \bar{X}^j_{\text{train}} \\ y(x'_j)=c}} \|\bar{x}_j - \bar{x'}_j\| \tag{5.2.5}$$

Figure 5.2.2 reports this value for different classes c and different depths j. One sees that the intra-class distances do not strictly decrease with depth, except on the last layer, which must be low-dimensional since the features are, up to projection, in a space of size 10. This is due to two phenomena: the normalization procedure whose choice can drastically change the final results and the averaging. Indeed, let us assume here that $\|W_j\| \leq 1$, then if $x, \tilde{x}$ are in the same class,

$$\|x_{j+1} - \tilde{x}_{j+1}\| \leq \|x_j - \tilde{x}_j\|, \tag{5.2.6}$$

but this does not imply that

$$\|\bar{x}_{j+1} - \bar{\tilde{x}}_{j+1}\| \leq \|\bar{x}_j - \bar{\tilde{x}}_j\|, \tag{5.2.7}$$

since the averaging is a projection and could break the distance inequality.

Those two experiments indicate we need to refine our measurement of contraction to explain the progressive and constant improvement of a 1-NN, that we will demonstrate in the next Section 5.2. Specifically, one should estimate the local intrinsic dimension: this is not possible since we do not have enough available samples in each neighborhood [28].

5.2.2 Progressive separation

Brutal contraction via a linear projection of the space would not preserve the distances between different classes. Yet, with a cascade of linear and non linear operators, it would be possible to progressively contract the space without losing in discriminability [72]. Several works [122, 79] reported that linear separability of deep features increases with depth. It might (but this is not the only solution) indicate that intra-class variabilities are progressively linearized [71], until the last layer, such that a final linear projector can build invariants. Following this approach, we start by applying a Gaussian SVM at each depth j, which is a discriminative locally linear classifier, with a fixed bandwidth. Indeed, observe here that the case of an infinite bandwidth corresponds to a linear SVM. We train it on the standardized features $\bar{X}^j_{\text{train}}$ corresponding to the training set at depth j and test it on $\bar{X}^j_{\text{test}}$, via a Gaussian SVM with bandwidth equal to the average l^2-norm of the points of the training set. We only cross-validate once the regularization parameter at one layer and then kept the parameters of the SVM constant. Figure 5.2.3 reports that the accuracy of this classifier increases at regular step with depth, which confirms the features become more separable.

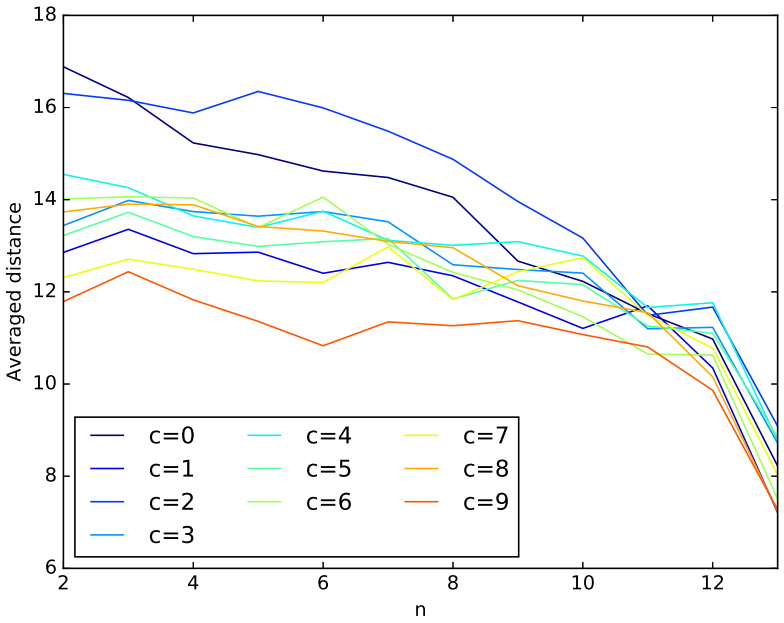


Figure 5.2.2: Averaged intra-class distances on CIFAR10, $K = 32$, at different depths j. Different colors correspond to different classes. The intra-class distances are globally decreasing with depth. Best viewed in color.

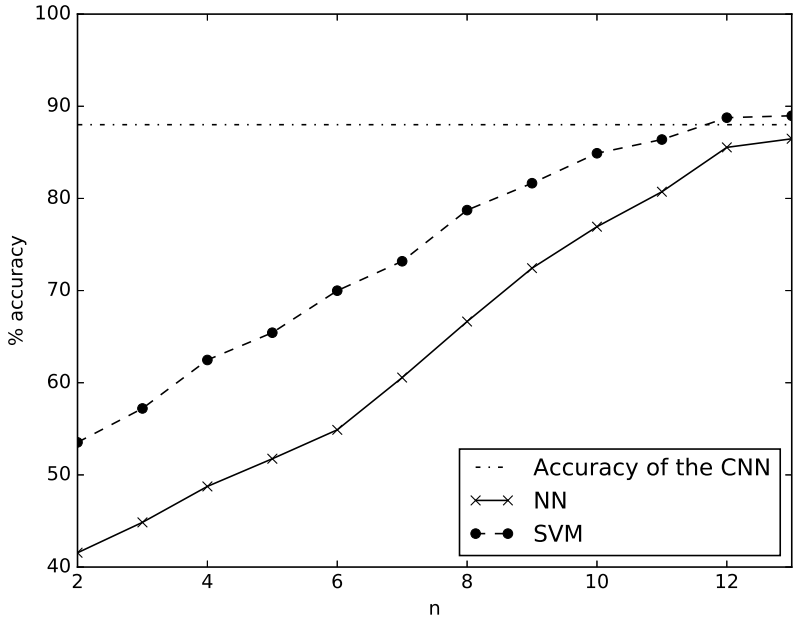


Figure 5.2.3: Accuracy on CIFAR10 at depth j via a Gaussian SVM and 1-NN. The size of the network is $K = 32$ and its accuracy on the testing set is 88.0%.

In fact, we prove this Gaussian SVM acts as a local classifier. A 1 nearest neighbor (1-NN) classifier is a naive and simple non-parametric classifier for high-dimensional signals, that simply assigns to a point the class of its closest neighbor. It can be interpreted as a local classifier with adaptive bandwidth [18]. It is unbiased, yet it bears a lot of variance. Besides, resampling the data will affect a lot the classification results. We train a 1-NN on $\bar{X}^j_{\text{train}}$ and test it on $\bar{X}^j_{\text{test}}$. We denote by $x^{(k)}$ the result of the k-th closest neighbor of a point, that is distinct of itself. We observe in Figure 5.2.3 that a 1-NN trained on $\bar{X}^j_{\text{train}}$ and tested on $\bar{X}^j_{\text{test}}$ performs nearly as well as a Gaussian SVM. Besides, the progression of this classifier is almost linear with respect to the depth. It means that the representation built by a deep network is in fact progressively more regular, which explains why the Gaussian SVM accuracy progressively improves.

5.2.3 Local Support Vectors

Our objective is to quantify at each depth j, the regularity of the representation constructed by a deep net in order to understand the progressive contraction of the space. In other words, we need to build a measure of this regularity. The contraction of the space is global, but we know from below that neighbors are meaningful: we would like to explain how they separate the different classes. We

thus introduce a notion of *local support vectors*. In the case of a SVM, a support vector corresponds to samples of the training set that delimit different classes, by interpolating a hyperplane between them [27]. It means that a support vectors permit to avoid the collapsing of the boundary classification [71]. But in our case, we do not have enough samples to estimate the exact boundaries. *Local support vectors* corresponds to support vectors defined by a local neighborhood. In other words, at depth j, the set of support vectors is defined as

$$\Gamma_j = \{\bar{x}_j, y(\bar{x}_j^{(1)}) \neq y(\bar{x}_j)\} \subset \bar{X}^j_{\text{train}} \tag{5.2.8}$$

which is the set of nearest neighbors that have a different class. In this section for a finite set X, we denote its cardinality by $|X|$.

5.2.3.1 Margin separation

In this subsection, we numerically observe a margin between support vectors. In [105], bounds on the margin are obtained with hypothesis of low dimensional structures, and this might be restrictive according to the analysis above. A margin at depth j is defined as:

$$\gamma^j = \inf_{y(\bar{x}_j^{(1)}) \neq y(\bar{x}_j)} \|\bar{x}_j^{(1)} - \bar{x}_j\| \geq 0 \tag{5.2.9}$$

Since our data are in finite number this quantity is always different from 0, but we need to measure if it is significant. We thus compare the distributions of distances of nearest neighbors belonging to the same class:

$$A_j = \{\|\bar{x}_j^{(1)} - \bar{x}_j\|, \bar{x}_j \notin \Gamma_j\}$$

and the distributions of the distances between support vectors

$$B_j = \{\|\bar{x}_j^{(1)} - \bar{x}_j\|, \bar{x}_j \in \Gamma_j\}. \tag{5.2.10}$$

The features have been normalized by a standardization. Figure 5.2.4 represents the cumulative distributions of A_j and B_j for different depths j. We recall that a cumulative distribution of a finite set $A \subset \mathbb{R}$ is defined as:

$$\mathcal{A}(t) = \frac{|\{x \leq t, x \in A\}|}{|A|} \tag{5.2.11}$$

. One observes that in a neighborhood of 0, $\mathcal{A}_j(t)$ is roughly the translation of $\mathcal{B}_j(t)$ by 0.5. It indicates there is a significant difference, showing γ_j is actually meaningful. Consequently there exists a margin between the spatially averaged samples of different classes, which means by Equation 5.2.2 that this margin exists between the samples themselves and their orbits by the action of translations.

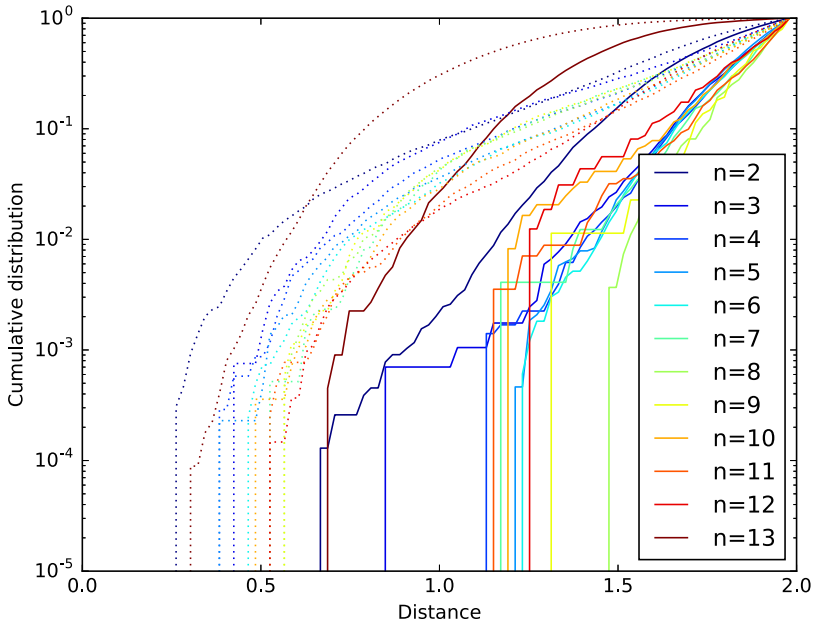


Figure 5.2.4: Cumulative distributions of distances: between a support vector and its nearest neighbors, e.g. $\mathcal{B}_j(t)$ (continuous line), and a point that is not a support vector and its nearest neighbor, e.g. $\mathcal{A}_j(t)$ (dashed line). Different colors correspond to different depths. The axis of the magnitude of the distance is in log scale. At a given depth, one sees there is a significative difference between the cumulative distribution, which indicates the existence of a margin. Best viewed in color.

5.2.3.2 Complexity of the classification boundary

Estimating a local intrinsic dimension is difficult when few samples per neighborhood are available, but the classes of the neighbors of each layers of the samples of $\bar{X}_{\text{train}}$ are known. In this subsection, we build a measure of the complexity of the classification boundary based on neighbors. This permits evaluating both separation and contraction properties. It can be viewed as a weak estimation of the intrinsic dimension [12], even if the manifold hypothesis might not hold. We compute an estimate of the efficiency of a k-NN to correctly find the label of a local support vector. To this end, we define by recurrence at depth j and for a given $k \in \mathbb{N}$, which is a number of neighbors, Γ_j^k via $\Gamma_j^1 = \Gamma_j$ and:

$$\Gamma_j^{k+1} = \left\{ \bar{x}_j \in \Gamma_j^k, |\{y(\bar{x}_j^{(l)}) \neq y(\bar{x}_j), l \leq k+1\}| > \frac{k}{2} \right\} \tag{5.2.12}$$

In other words, Γ_j^k is the set of points at depth j that are not well-classified by l-NN using majority vote, for $l \leq k$. By construction, $\Gamma_j^{k+1} \subset \Gamma_j^k$ which implies that

$$|\Gamma_j^{k+1}| \leq |\Gamma_j^k|. \tag{5.2.13}$$

Since the number of samples is finite, this sequence converges to the number of samples of the training set that can not be identified by their nearest neighbors. The decay and the amplitude of $|\Gamma_j^k|$ is an indicator of the regularity of the classification boundary. Recall that for a deep network, the 1-NN classifier has better generalization properties with deeper features. A small value of $|\Gamma_j^k|$ indicates that a few samples are necessary to build the classification boundary (contraction), and at a given depth j, if $|\Gamma_j^k|$ decreases quickly to its constant value, it means a few neighbors are required to build the decision boundary (separation).

Figure 5.2.5 indicates that the classification boundary is uniformly more regular with depth, in term of number of local support vectors and number of neighbors required to estimate the correct class. This measure has the advantage of being simple to compute, yet this analysis must be refined in a future work.

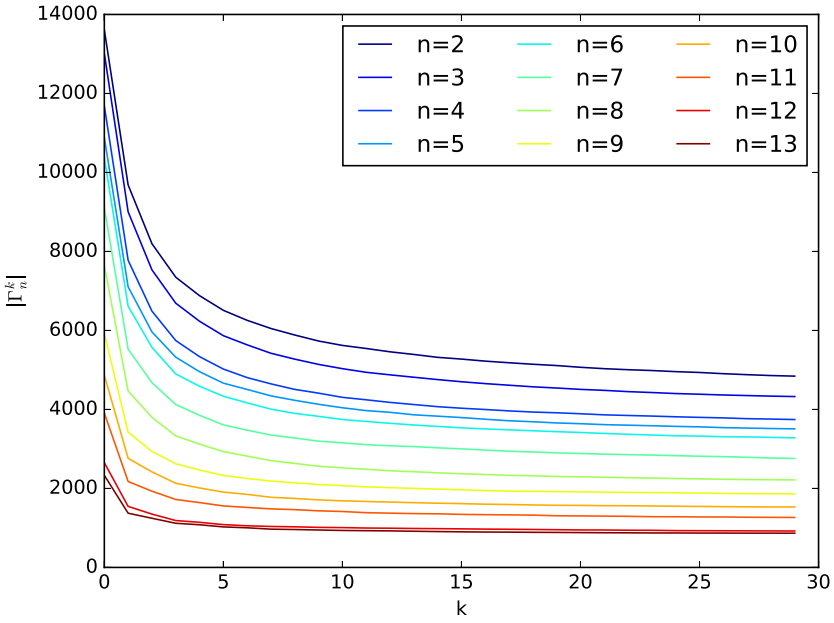


Figure 5.2.5: $|\Gamma_j^k|$ for different depth j. Different depths are represented by different colors. The limit value of $|\Gamma_j^k|$ is reached faster by deeper layers, and the value $|\Gamma_j^k|$ globally decrease with depth: the boundary classification is progressively more regular. Best viewed in color.

Chapter 6

Hierarchical Attribute CNNs

This chapter introduces some classes of CNNs that organizes the channel axes of each layer. Indeed, their architectures consist in high-dimensional convolutions that are performed jointly in space and along multi-dimensional channel axis. It can be viewed as an incremental work over the roto-translation scattering presented in Chapter 3, in the following sense: a roto-translation scattering performs a convolution along the index of the angular and spatial variables [101], that were created by the first wavelet transform. Following this idea, CNNs [71, 16, 25] that perform convolutions along each newly created indexes are built. We refer to these indexes as attributes because they aim to hold the discriminative information of the representation, and thus non-informative attributes should be progressively reduced as we explain below.

The variabilities which must be eliminated are mathematically defined as the group of symmetries of the classification function [71, 38]. It is the group of transformations (not necessarily linear) which preserves the labels of the classification problem. Translations are one example of such a symmetry group. In a vanilla CNN invariants to translations are computed with spatial convolutions, followed by a final averaging. Besides memorizing symmetries, a network must as well memorizing properties of the samples of a dataset: can those two mechanisms be unified? Memory means here the ability to encode and restore properties that are important for classification.

Fully understanding a deep neural network classifier requires specifying its symmetry group and invariants beyond translations, especially of non-geometrical nature and without any specific semantic meaning. The Multiscale Hierarchical CNNs introduced in [71] are a new class of CNNs. A typical Multiscale Hierarchical CNNs [71] is a cascade of general convolutions along the symmetry group. Contrarily to common CNNs introduced by [59] that perform a unspecified sum of weighted spatial kernels, the spatial kernels are combined through convolutions along the channel indexes. They give explicit information on invariants by disentangling progressively more signal attributes as the depth increases. It permits to study the invariances w.r.t. those symmetries, and addresses several questions: is it possible to organize the memory of the network with high dimen-

sional translations? If so, what is the nature of those translations? However, its implementation is complicated because the number of parameters and the size of the layers grows exponentially with the depth.

To tackle this issue, we introduce and implement a novel class of CNNs, called Hierarchical Attribute Convolutional Neural Networks (HCNNs), which is a subset of the generic class Multiscale Hierarchical CNNs described in [71]. We suggest that images variabilities can be then learned and smoothly mapped into the group of translations of $\mathbb{R}^d, d \in \mathbb{N}$. Smoothly means that the linear metric preserves locally the property of an image, and that it is almost always possible to obtain translated representations along the attributes, which should not be the case in vanilla CNNs.

Our contributions via an experimental study of the trained HCNNs are the following. First, we discuss CNNs and necessary conditions on their structure in Subsection 6.1.1, and we relate them to Multiscale Hierarchical CNNs in Subsection 6.1.2. Then, we introduce the class of HCNNs in Subsection 6.1.3. Subsection 6.2.1 shows that they require a reduced number of parameters compared to vanilla CNNs to achieve similar performances. Secondly, we give evidence of an effective memory organization at several layers in Subsection 6.2.2.1. We also explain in Subsection 6.2.2.2 why this method is not likely to have structured correctly the memory of the network.

6.1 Architectures descriptions

6.1.1 Deep Convolutional Networks and Group Invariants

A classification problem associates a class $y = f(x)$ to any vector $x \in \mathbb{R}^N$ of N parameters. Deep convolutional networks transforms x into multiple layers x_j of coefficients at depths j, whose dimensions are progressively reduced after a certain depth [60]. We review general properties of CNNs and motivate the notion of group of symmetries, which is a key concept of this work.

We adopt different notations from the previous sections as we interpret differently the layers of a neuron.We shall numerically concentrate on color images $x(u, v)$ where $u = (u_1, u_2)$ are the spatial coordinates and $1 \leq v \leq 3$ is the index of a color channel. The input $x(u, v)$ may, however, correspond to any other type of signals. For sounds, $u = u_1$ is time and v may be the index of audio channels recorded at different spatial locations.

Each layer is an array of signals $x_j(u, v)$ where u is the native spatial index of x, and v is a 1-dimensional channel parameter. A deep convolutional network iteratively computes:

$$x_{j+1} = \rho W_{j+1} x_j \tag{6.1.1}$$

with $x_0 = x$. Each W_{j+1} computes sums over v of convolutions along u, with filters of small support, here chosen to be 3. The resolution of $x_j(u, v)$ along u is progressively reduced by a subsampling as j increases until an averaging in the final output layer. The operator $\rho(z)$ is a point-wise non-linearity In this work,

we shall use exponential linear units ELU [23]. It transforms each coefficient z with bias b into:

$$\rho(z) = \begin{cases} z + b & \text{if } z \geq -b \\ e^c - 1 & \text{otherwise} \end{cases} \tag{6.1.2}$$

As the depth increases, the discriminative variations of the initial image x along u are progressively transferred to the channel index v of x_j. At the last layer x_J, v stands for the class index and u has disappeared. An estimation $\tilde{y}$ of the signal class $y = f(x)$ is computed by applying a soft-max to $x_J(v)$. It is difficult to understand the meaning of this channel index v whose size and properties changes with depth. It mixes multiple unknown signal representations with an arbitrary ordering. Hierarchical Attribute convolution networks will dress this issue by imposing a high-dimensional hierarchical structure on v, with an ordering specified by the translation group.

In standard CNN, each $x_j = \Phi_j x$ is computed with a cascade of convolutions and non-linearities

$$\Phi_j = \rho W_j ... \rho W_1, \tag{6.1.3}$$

whose supports along u increase with the depth j. These operators replace x by the variables x_j to estimate the class $y = f(x)$. To avoid errors, this change of variable must be discriminative, despite the dimensionality reduction, in the sense that

$$\forall (x, x') \in \mathbb{R}^{2N} \Phi_j(x) = \Phi_j(x') \Rightarrow f(x) = f(x'). \tag{6.1.4}$$

This is necessary and sufficient to guarantee that there exists a classification function f_j such that

$$f = f_j \circ \Phi_j \tag{6.1.5}$$

and hence

$$\forall x \in \mathbb{R}^N, f_j(x_j) = f(x). \tag{6.1.6}$$

The function $f(x)$ can be characterized by its groups of symmetries. A group of symmetries of f is by definition a group of operators g that are invertible and stable w.r.t. composition, which transforms any x into $x' = g.x$ which belong to the same class: $f(x) = f(g.x)$. The discriminative property (6.1.4) implies that:

$$\Phi_j(x) = \Phi_j(g.x) \Rightarrow f(x) = f(g.x). \tag{6.1.7}$$

The property (6.1.4) is thus equivalent to impose that groups of symmetries of Φ_j are groups of symmetries of f. Learning appropriate changes of variables can thus be interpreted as learning progressively more symmetries of f, from the perspective of [71]. The network must be sufficiently flexible to compute changes of variables Φ_j whose symmetries approximate the symmetries of f.

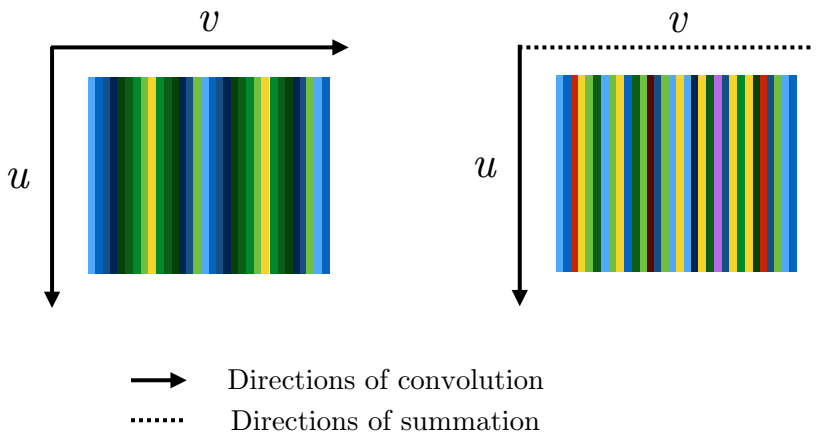


Figure 6.1.1: We illustrate the difference between linear operators of vanilla CNNs (Right) and the convolution of a HCNN (Left)

Deep convolutional networks are cascading convolutions along the spatial variable u so that Φ_j is covariant to spatial translations. If x is translated along u then $x_j = \Phi_j x$ is also translated along u. This covariance implies that for all v, $\sum_u x_j(u, v)$ is invariant to translations of x. Next section explains how to extend this property to higher dimensional attributes with multidimensional convolutions.

6.1.2 Multiscale Hierarchical Convolutional Neural Networks

Multiscale Hierarchical networks are convolutional networks that explicitly structure the channel dimension via cascade of convolutions along symmetry groups [71, 16]. The one-dimensional unordered index v of a vanilla CNN is replaced by an ordered multidimensional vector of attributes $v = (v_1, ..., v_j)$ and all linear operators W_j are convolutions over (u, v). We explain their construction and a specific architecture adapted to an efficient learning procedure. Each layer $x_j(u, v)$ is indexed by a vector of multidimensional parameters $v = (v_1, ..., v_j)$ of dimension j. Each v_k is an "attribute" of x which is learned to discriminate classes $y = f(x)$. This word is appropriate as it refers to a strong discriminative index of the channels: this is in particular true for the final layer of the CNN, as an attribute corresponds to one class of the classification problem. The operators W_j are defined as multidimensional convolutions along the index space

(u, v). As previously explained, this covariance to translations implies that, for $0 \leq k \leq j$, the sum:

$$\sum_{v_k} x_j(u, v_0, ..., v_j) \tag{6.1.8}$$

is invariant to translations of previous layers along v_k. A convolution of $z(u, v)$ by a filter $w(u, v)$ of support S is written

$$z \star w(u, v) = \sum_{(u',v') \in S} z(u - u', v - v')\, w(u', v'). \tag{6.1.9}$$

Since $z(u, v)$ is defined in a finite domain of (u, v), boundary issues can be solved by extending z with zeros or as a periodic signal. We use zero-padding extensions for the next sections, except for the last section, where we use periodic convolutions. Both cases give similar accuracies. The Figure 6.1.1 highlights the differences between the convolutions performed in a vanilla CNN and in a Multiscale Hierarchical CNN.

The network takes as an input a color image $x(u, v_0)$, or any type of multichannel signal indexed by v_0. The first layer computes a sum of convolutions of $x(u, v_0)$ along u, with filters $w_{1,v_0,v_1}(u)$

$$x_1(u, v_1) = \rho\Big(\sum_{v_0} x(\cdot, v_0) \star w_{1,v_0,v_1}(u) \Big). \tag{6.1.10}$$

For any $j \geq 2$, W_j computes convolutions of $x_{j-1}(u, v)$ for $v = (v_1, ..., v_{j-1})$ with a family of filters $\{w_{v_j}\}_{v_j}$ indexed by the new attribute v_j:

$$x_j(u, v, v_j) = \rho\Big(x_{j-1} \star w_{v_j}(u, v) \Big). \tag{6.1.11}$$

As explained in [71], W_j has two roles. First, these convolutions indexed by v_j prepare the discriminability (6.1.4) of the next layer x_{j+1}, despite local or global summations along $(u, v_1, ..., v_{j-1})$ implemented at this next layer. It thus propagates discriminative variations of x_{j-1} from $(u, v_1, ..., v_{j-1})$ into v_j.

Second, each convolution with w_{v_j} computes local or global invariants by summations along $(u, v_1, ..., v_{j-2})$, in order to reduce dimensionality. This dimensionality reduction is implemented by a subsampling of (u, v) at the output (6.1.11), which we omitted here for simplicity. Since v_k is the index of multidimensional filters, a translation along v_k is a shift along an ordered set of multidimensional filters. Again, for any $k < j - 1$, $\sum_{v_k} x_{j-1}(u, v_1, ..., v_k, ..., v_{j-1})$ is invariant to any such shift.

The final operator W_J computes invariants over u and all attributes v_k but the last one:

$$x_J(v_{J-1}) = \sum_{u, v_1, ..., v_{J-1}} x_{J-1}(u, v_1, ..., v_{J-1}). \tag{6.1.12}$$

The last attribute v_{J-1} corresponds to the class index, and its size is the number of classes. The class $y = f(x)$ is estimated by applying a soft-max operator on $x_J(v_{J-1})$.

Proposition 5. *The last layer x_J is invariant to translations of $x_j(u, v_1, ..., v_j)$ along $(u, v_1, ..., v_j)$, for any $j < J - 1$.*

Proof. Observe that $x_J = W_J \rho W_{J-1} ... \rho W_j x_j$. Each W_k for $j < k < J$ is a convolution along $(u, v_0, ..., v_j, ..., v_k)$ and hence covariant to translations of $(u, v_0, ..., v_j)$. Since ρ is a point-wise operator, it is also covariant to translations. Translating x_j along $(u, v_1, ..., v_j)$ thus translates x_{J-1}. Since (6.1.12) computes a sum over these indices, it is invariant to these translations □

This proposition proves that the soft-max of x_J approximates the classification function:

$$f_J(x_J) = f(x) \tag{6.1.13}$$

by an operator which is invariant to translations along the high-dimensional index $(u, v) = (u, v_1, ..., v_j) \in \mathbb{R}^{j+2}$. Ideally, the change of variable x_j thus aims at mapping the symmetry group of f into a high-dimensional translation group, which is the translation group $\mathbb{R}^j$. The next Section 6.2 develops the outcomes of this approach.

However, this requires an important word of caution. A translation of $x_j(u, v_1, ..., v_j)$ along u corresponds to a translation of $x(u, v_0)$ along u. On the contrary, a translation along the attributes $(v_1, ..., v_j)$ usually does not correspond to transformations on x. Translations of x_j along $(v_1, ..., v_j)$ form a group of symmetries of f_j but do not define transformations of x and hence do not correspond to a symmetry group of f. Next sections analyze the properties of translations along attributes computed numerically.

Let us give examples over images or audio signals $x(u)$ having a single channel. The first layer (6.1.10) computes convolutions along u:

$$x_1(u, v_1) \quad = \quad \rho(x \star w_{v_1}(u))$$

For audio signals, u is time. This first layer usually computes a wavelet spectrogram, with wavelet filters w_{v_1} indexed by a log-frequency index v_1. A frequency transposition corresponds to a log-frequency translation $x_1(u, v_1 - \tau)$ along v_1. If x is a sinusoidal wave then this translation corresponds to a shift of its frequency and hence to a transformation of x.

However, for more general signals x, there exists no x' such that $\rho(x' \star w_{v_1}(u)) = x_1(u, v_1 - \tau)$. It is indeed well known that a frequency transposition does not define an exact signal transformation. Other audio attributes such as timber are not well-defined transformations on x either although important attributes to classify sounds.

For images, $u = (u_1, u_2)$ is a spatial index. If $w_{v_1} = w(r_{v_1}^{-1} u)$ is a rotation of a filter $w(u)$ by an angle v_1 then

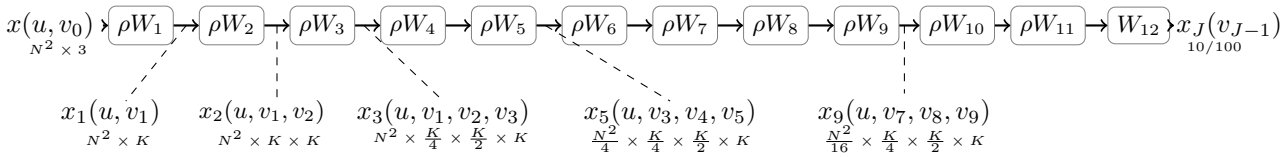


Figure 6.1.2: Implementation of a hierarchical attribute convolutional network as a cascade of 5D convolutions W_j . The figure gives the size of the intermediate layers stored in 5D arrays. Dash dots lines indicate the parametrization of a layer x_j and its dimension. We only represent dimensions when the output has a different size from the input.

$$x_1(u, v_1 - \tau) = \rho(x_\tau \star w_{v_1}(r_\tau u)) \quad \text{with} \quad x_\tau(u) = x(r_\tau^{-1} u). \tag{6.1.14}$$

However, there exists no x' such that:

$$\rho(x' \star w_{v_1}(u)) = x_1(u, v_1 - \tau)$$

because of the missing spatial rotation $r_\tau u$. These examples show that translations $x_j(u, v_1, .., v_j)$ along the attributes $(v_1, ..., v_j)$ usually do not correspond to a transformation of x.

6.1.3 Hierarchical Attribute CNNs

We now depict the Hierarchical Attribute Networks and specify its properties and implementation.

6.1.3.1 5-D Dimensional Architectures

Hierarchical Attribute Network are in a subset of the Multiscale Hierarchical Networks, with an efficient implementation. Indeed, the convolutions kernels size and layer size grow exponentially with depth, and thus it is necessary to incorporate intermediate projections to avoid this difficulty. The layers are indexed by two-dimensional spatial indices $u = (u_1, u_2)$ and progressively higher dimensional attributes $v = (v_1, ..., v_j)$. To avoid computing high-dimensional vectors and convolutions, we introduce an image classification architecture which eliminates the dependency relatively to all attributes but the last three (v_{j-2}, v_{j-1}, v_j), for $j > 2$. Since $u = (u_1, u_2)$, all layers are stored in five dimensional arrays.

The network takes as an input a color image $x(u, v_0)$, with three color channels $1 \leq v_0 \leq 3$ and $u = (u_1, u_2)$. Applying (6.1.10) and (6.1.11) up to $j = 3$ computes a five-dimensional layer $x_3(u, v_1, v_2, v_3)$. For $j > 3$, x_j is computed as a linear combination of marginal sums of x_{j-1} along v_{j-3}. Thus, it does not depend anymore on v_{j-3} and can be stored in a five-dimensional array indexed

by $(u, v_{j-2}, v_{j-1}, v_j)$. This is done by convolving x_{j-1} with a a filter w_{v_j} which does not depend upon v_{j-3}:

$$w_{v_j}(u, v_{j-3}, v_{j-2}, v_{j-1}) = w_{v_j}(u, v_{j-2}, v_{j-1}) \ . \tag{6.1.15}$$

We indeed verify that this convolution is a linear combination of sums over v_{j-3}, so x_j depends only upon $(u, v_{j-2}, v_{j-1}, v_j)$. The convolution is subsampled by 2^{s_j} with $s_j \in \{0, 1\}$ along u, and a factor 2 along v_{j-1} and v_j:

$$x_j(u, v_{j-2}, v_{j-1}, v_j) = x_{j-1} \star w_{v_j}(2^{s_j} u, 2v_{j-2}, 2v_{j-1}), \tag{6.1.16}$$

At depth j, the array of attributes $v = (v_{j-2}, v_{j-1}, v_j)$ is of size $K/4 \times K/2 \times K$. The parameters K and spatial subsampling factors s_j are adjusted with a trade-off between computations, memory and classification accuracy. As said below, the final layer is computed with a sum (6.1.12) over all parameters but the last one, which corresponds to the class index:

$$x_J(v_{J-1}) = \sum_{u, v_{J-3}, v_{J-2}} x_{J-1}(u, v_{J-3}, v_{J-2}, v_{J-1}). \tag{6.1.17}$$

6.1.3.2 Filter Factorization for Training

Due to its substantial number of parameters, the optimization of CNN is ill-conditioned [43]. The work of [49] suggests that a batch normalization permits to partially solve this problem and permits to remove other regularization techniques such as dropout [107].

Just as in any other CNN, the gradient descent is badly conditioned because of the large number of parameters [43]. We precondition and regularize the 4 dimensional filters w_{v_j}, by normalizing a factorization of these filters. We factorize $w_{v_j}(u, v_{j-3}, v_{j-2}, v_{j-1})$ into a sum of Q separable filters:

$$w_{v_j}(u, v_{j-3}, v_{j-2}, v_{j-1}) = \sum_{q=1}^{Q} h_{j,q}(u)\, g_{v_j,q}(v_{j-2}, v_{j-1}) \tag{6.1.18}$$

and introduce an intermediate normalization before the sum. For δ a Dirac, to highlight that multidimensional convolutions are performed, let us write:

$$h_{j,q}(u, v) = h_{j,q}(u)\, \delta(v) \tag{6.1.19}$$

and

$$g_{v_j,q}(u, v) = \delta(u)\, g_{v_j,q}(v). \tag{6.1.20}$$

The batch normalization is applied to $x_{j-1} \star h_{j,q}$ and subtracts a mean array $m_{j,q}$ while normalizing by the standard deviations $\sigma_{j,q}$:

$$\tilde{x}_{j,q}(u, v) = \frac{x_{j-1} \star h_{j,q} - m_{j,q}}{\sigma_{j,q}}. \tag{6.1.21}$$

This normalized output is retransformed according to (6.1.16) by a sum over q and a subsampling:

$$x_j(u,v) = \rho\Big(\sum_{q=1}^{Q} \tilde{x}_{j,q} \star g_{v_j,q}(2^{s_j}u, 2v)\Big) \tag{6.1.22}$$

The convolution operator W_j is thus subdivided into a first operator W_j^h which computes standardized 2D convolutions along u cascaded with W_j^g which sums Q 2D convolutions along v. Since the tensor rank of W_j cannot be larger than 9, using $Q \geq 9$ does not restrict the rank of the operators W_j. However, as reported in [51], increasing the value of Q introduces an overparametrization which regularizes the optimization. Increasing Q from 9 to 16 and then from 16 to 32 brings a substantial improvement.

We also report a modification of our network (denoted by $(+)$) which incorporates an intermediate non-linearity:

$$x_j(u,v) = \rho(W_j^g \rho(W_j^h x_{j-1})). \tag{6.1.23}$$

Observe that in this case, x_j is still covariant with the actions of the translations along (u,v), yet the factorization of w_{v_j} into $(h_{j,q}, g_{v_j,q})$ does not hold anymore. Figure 6.1.2 describes our two model architectures, with layer sizes that are discussed in the next subsection.

6.2 Expliciting the structuration

This section shows that Hierarchical Attribute Convolution Networks achieve similar classification accuracies on the CIFAR image dataset as state-of-the-art architectures, with much fewer parameters. We also investigate the properties of translations along the attributes v_j learned on CIFAR.

6.2.1 Hierarchical Attribute CNNs on CIFAR datasets

6.2.1.1 Performances and Parameters Reduction

Our newly introduced Hierarchical Attribute Convolution Networks (HCNN) have been tested on CIFAR10 and CIFAR100 image databases. CIFAR10 has 10 classes, while CIFAR100 has 100 classes, which makes it more challenging. The train and test sets have $50k$ and $10k$ colored images of 32×32 pixels. Images are preprocessed via a standardization along the RGB channels. No whitening is applied as we did not observe any improvement.

Our HCNN is trained in the same way as a classical CNN. We train it by minimizing a neg-log entropy loss, via SGD with Nesterov momentum 0.9 for 200 epochs. An initial learning rate of 0.25 is chosen while being reduced by a factor 2 every 40 epochs. Each mini-batch is of size 50. The learning is regularized by a weight decay of $2\,10^{-4}$ [56]. We incorporate a data augmentation with random translations of 4 pixels and flips [56].

Model	# Parameters	Accuracy
HCNN	0.098M	91.43
HCNN (+)	0.34M	92.50
All-CNN [106]	1.3M	92.75
ResNet-20 [46]	0.27M	91.25
NiN [63]	0.98M	91.20
WRN-student [119]	0.17M	91.23
FitNet [91]	2.5M	91.61

(a) Classification accuracy on CIFAR10 dataset.

Model	# Parameters	Accuracy
HCNN	0.25M	62.01
HCNN (+)	0.89M	63.19
All-CNN [106]	1.3M	66.29
NiN [63]	0.98M	64.32
FitNet [91]	2.5M	64.96

(b) Classification accuracy on CIFAR100 dataset.

Table 6.1: Classification accuracy on CIFAR datasets, with the number of parameters of a given architecture.

We evaluate our Hierarchical CNN on CIFAR datasets (Table 6.1) in the setting explained above. Our network achieves an error of 8.6% on CIFAR-10, which is comparable to recent architectures. On CIFAR-100 we achieve an error rate of 38%, which is about 4% worse than the closely related all-convolutional network baseline, but our architecture has an order of magnitude fewer parameters.

Classification algorithms using a priori defined representations or representations computed with unsupervised algorithms have an accuracy which barely goes above 80% on CIFAR-10, as shown in Chapter 3. On the contrary, supervised CNN have an accuracy above 90% as shown by Table 6.1a. This is also the case for our structured hierarchical network which has an accuracy above 91%. Improving these results may be done with larger K and Q which could be done with faster GPU implementation of multidimensional convolutions, although it is a technical challenge [17].

Our proposed architecture is based on "plain vanilla" CNN architectures to which we compare our results in Table 6.1. Even though the performance gap to the best models is substantial, we do perform in the same regime as state-of-the-art architectures, showing that our HCNN, despite its highly constrained architecture, manages to capture most important variations in the data. In the following, we study the properties resulting from the hierarchical structuration of our network, compared with classical CNN.

The structuration of a Deep neural network aims at reducing the number of parameters and making them easier to interpret in relation to signal models.

Reducing the number of parameters means characterizing better the structures which govern the classification. Let us then count the number of parameters of our implementation of the HCNN.

The number of free parameters of the original architecture is the number of parameters of the convolution kernels w_{v_j} for $1 \leq v_j \leq K$ and $2 < j < J$, although they are factorized into separable filters $h_{j,q}(u)$ and $g_{v_j,q}(v_{j-2}, v_{j-1})$ which involve more parameters. The filters w_{v_j} have less parameters for $j = 2, 3$ because they are lower-dimensional convolution kernels.

Parametrization of the HCNN

For CIFAR datasets, a spatial downsampling by 2 along u is applied at depth $j = 4, 8$. Figure 6.1.2 describes our two model architectures, omitting batch normalization and non-linearity for brevity. Each layer has $K = 16$ outputs and its total depth is $J = 12$. In CIFAR-10, for $3 < j < J$, each w_{v_j} has a spatial support of size 3^2 and a support of 7×11 along (v_{j-2}, v_{j-1}). If we add the 10 filters which output the last layer, the resulting total number of network parameters is approximately 0.098M. In CIFAR-100, the filters rather have a support of 11×11 along (v_{j-2}, v_{j-1}) but the last layer has a size 100 which considerable increases the number of parameters which is approximatively 0.25M.

The second implementation (+) introduces a non-linearity ρ between each separable filter, so the overall computations can not be reduced to equivalent filters w_{v_j}. There are $Q = 32$ spatial filters $h_{j,q}(u)$ of support 3×3 and QK filters $g_{v_j,q}(v_{j-2}, v_{j-1})$ of support 7×11. The total number of coefficients required to parametrize $h_{j,q}, g_{v_j,q}$ is approximatively 0.34M. In CIFAR-100, the number of parameters becomes 0.89M. The total number of parameters of the implementation (+) is thus much bigger than the original implementation which does not add intermediate non-linearities. Next section compares these numbers of parameters with architectures that have similar numerical performances.

6.2.1.2 Comparison with limited parameters architectures

This section compares our HCNN to other structured architectures and algorithms which reduce the number of parameters of a CNN during, and after training. We show that Hierarchical Attribute Convolutional Networks involve less parameters during and after training than other architectures in the literature.

We review various strategies to reduce the number of parameters of a CNN and compare them with our Hierarchical Attribute CNN. Several studies show that one can factorize CNN filters [32, 52] *a posteriori*. A reduction of parameters is obtained by computing low-rank factorized approximations of the filters calculated by a trained CNN. It leads to more efficient computations with operators defined by fewer parameters. Another strategy to reduce the number of network weights is to use teacher and student networks [119, 91], which

optimize a CNN defined by fewer parameters. The student network adapts a reduced number of parameters for data classification via the teacher network.

A parameter redundancy has also been observed in the final fully connected layers used by number of neural network architectures, which contain most of the CNN parameters [22, 66]. This last layer is replaced by a circulant matrix during the CNN training, with no loss in accuracy, which indicates that last layer can indeed be structured. Other approaches [51] represent the filters with few parameters in different bases, instead of imposing that they have a small spatial support. These filters are represented as linear combinations of a given family of filters, for example, computed with derivatives. This approach jointly structures the channel and spatial dimensions. Finally, HyperNetworks [44] permit to drastically reduce the number of parameters used during the training step, to 0.097M and obtain 91.98% accuracy. However, we do not report them as 0.97M corresponds to a non-linear number of parameters for the network, e.g. the parametrization they use is not linear. No other architecture trained from scratch with this number of parameters obtains a similar performance.

Table 6.1 gives the performance of different CNN architectures with their number of parameters, for the CIFAR10 and CIFAR100 datasets. For Hierarchical Attribute networks, the convolution filters are invariant to translations along u and v which reduces the number of parameters by an important factor compared to other architectures. All-CNN [106] is an architecture based only on sums of spatial convolutions and ReLU non-linearities, which has a total of 1.3M parameters, and a similar accuracy to ours. Its architecture is similar to our hierarchical architecture, but it has much more parameters because filters are not translation invariant along v. Interestingly, a ResNet [46] has more parameters and performs similarly whereas it is a more complex architecture, due to the shortcut connections. WRN-student is a student resnet [119] with 0.2M parameters trained via a teacher using 0.6M parameters and which gets an accuracy of 93.42% on CIFAR10. FitNet networks [91] also use compression methods but need at least 2.5M parameters, which is much larger than our network. Our architecture brings an important parameter reduction on CIFAR10 for accuracies around 90% There is also a drastic reduction of parameters on CIFAR100.

6.2.2 A potential organization of the representation indexes

The principal motivation and investigation of this work was to understand if the translation along the new attributes is effective. It means that the structure of the operators should be inherited by the representation. The attributes and their translation should ideally have a semantic meaning. For example, in the last layer, an attribute stands for the class.

This was partially achieved, and we suggest solutions to improve it. First, we explain a method to interpret translations along a new attribute, and finally we explain the issues of our result.

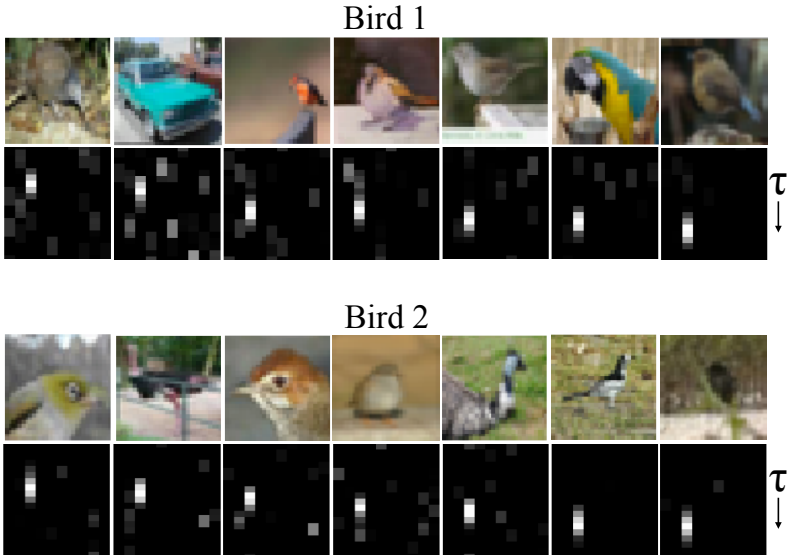


Figure 6.2.1: The first images of the first and third rows are the two input image x. Their invariant attribute array $\bar{x}_j(v_{j-1}, v_j)$ is shown below for $j = J-1$, with high amplitude coefficients appearing as white points. Vertical and horizontal axes correspond respectively to v_{j-1} and v_j, so translations of v_{j-1} by τ are vertical translations. An image x^τ in a column $\tau + 1$ has an invariant attribute $\bar{x}_j^\tau$ which is shown below. It is the closest to $\bar{x}_j(v_{j-1} - \tau, v_j)$ in the databases.

6.2.2.1 Interpreting the translation

In this subsection, we try to interpret semantically the notion of translation along an attribute. In other words, we try to derive if translations along attributes are related to properties of natural images.

The structure of Hierarchical Attribute CNN opens up the possibility of interpreting inner network coefficients, which is usually not possible for CNNs. A major mathematical challenge is to understand the type of invariants computed by deeper layers of a CNN. Hierarchical networks computes invariants to translations relatively to learned attributes v_j, which are indices of the filters w_{v_j}. One can try to relate the translations of these attributes to modifications of image properties.

As explained in Section 6.1.2, a translation of x_j along v_j usually does not correspond to a well-defined transformation of the input signal x but it produces a translation of the next layers. Translating x_j along v_j by τ translates $x_{j+1}(u, v_{j-1}, v_j, v_{j+1})$ along v_j by τ.

To analyze the effect of this translation, we eliminate the variability along v_{j-2} and define an invariant attribute array by choosing the central spatial position u_0:

Figure 6.2.2: The first columns give the input image x, from which we compute the invariant array $\bar{x}_j$ at a depth $3 \leq j \leq 11$ which increases with the row. The next images in the same row are the images x^τ whose invariant arrays $\bar{x}_j^\tau$ are the closest to $\bar{x}_j$ translated by $1 \leq \tau \leq 7$, among all other images in the databases. The value of τ is the column index minus 1.

$$\bar{x}_j(v_{j-1}, v_j) = \sum_{v_{j-2}} x_j(u_0, v_{j-2}, v_{j-1}, v_j).$$

We relate this translation to an image in the training dataset by finding the image x^τ in the dataset which minimizes:

$$\|\bar{x}_j(v_{j-1} - \tau, v_j) - \bar{x}_j^\tau(v_{j-1}, v_j)\|_2,$$

if this minimum Euclidean distance is sufficiently small. To compute accurately a translation by τ we eliminate the high frequency variations of x_j and x_j^τ along v_{j-1} with a filter which averages consecutive samples, before computing their translation. The network used in this experiment is implemented with circular convolutions to avoid border effects, which have nearly the same classification performance.

Figure 6.2.1 shows the sequence of x^τ obtained with a translation by τ of $\bar{x}_j$ at depth $j = J - 1$, for two images x in the "bird" class. Since we are close to the output, we expect that translated images belong to the same class, because by construction, two signals with a similar averaging will belong to the same class. This is not the case for the second image of the first "Bird 1". It is a "car" instead of a "bird". This corresponds to a classification error but observe that $\bar{x}_{J-1}^\tau$ is quite different from $\bar{x}_{J-1}$ translated. We otherwise observe that in these final layers, translations of $\bar{x}_{J-1}$ defines images in the same class.

Figure 6.2.2 gives sequences of translated attribute images x^τ, computed by translating $\bar{x}_j$ by τ at different depth j and for different input x. As expected, at small depth j, translating an attribute v_{j-1} does not define images in the same class. These attribute rather correspond to low-level image properties which depend upon fine scale image properties. However, these low-level properties can not be identified just by looking at these images. Indeed, the closer images x^τ identified in the databases are obtained with a distance over coefficients which are invariant relatively to all other attributes. These images are thus very different and involve variabilities relatively to all other attributes. To identify the nature of an attribute v_j, a possible approach is to correlate the images x^τ over a large set of images, while modifying known properties of x.

At deep layers j, translations of $\bar{x}_j$ define images x^r which have a progressively higher probability to belong to the same class as x. These attributes transformations correspond to large scale image pattern related to modifications of the filters $w_{v_{j-1}}$. In this case, the attribute indices could be interpreted as addresses in organized arrays. The translation group would then correspond to translations of addresses. Understanding better the properties of attributes at different depth is an issue that will be explored in the future.

6.2.2.2 Limitations

Ideally, HCNNs are an easier way to learn symmetries. Surprisingly, we could not relate those symmetries to geometrical properties of a signal such as rotation. It implies that a reverse engineering to understand the meaning of the translation is difficult.

Besides, none of the filters or the intermediary representation were smooth or with a localized support, w.r.t. the attributes v. It is surprising because it suggests that the network builds an irregular representation, which is inconsistent with the hypothesis of learning a Lie group of symmetries.

This suggests also that we do not know if there is a memorization phenomenon linked to translations along attributes, or if there is a generalization thanks to those translations. A large part of the problem is to make apparent the regularities that the network exploits.

There is an analogy with the work of Coifman [26] on databases. It suggests that, in the case of "questionnaires", finding regular representations aim at organizing the "question", yet as well the "answers" to the questionnaires, simultaneously. This is not a natural property. In our work, "questions" correspond to the architecture whereas "answers" are the set of representation at a given depth. We organized the "questions", and we observed it does not imply an organization of the "answers". This is surprising, because the "questions" (e.g. the model) are optimized through the "answers" (e.g. the training set), and this seems to indicate that we should more precisely incorporate an organization of the "answers". In particular, it requires to better understand the nature of the source of regularity of CNNs: are they indeed parallel transport along group of variabilities, as suggested by [71]?

Chapter 7

Conclusion

In this thesis, we have provided some various pipelines for classifications of images, that permit to elucidate some intriguous properties of deep neural networks. We now review the content of each chapter of this thesis.

We conclude that despite not being learned, the Scattering Transform can be applied to complex image classification with competitive performances. We have presented applications of the Separable Roto-translation Scattering Transform followed by SVM classifiers for complex image classification. We demonstrate that this representation is competitive with unsupervised algorithms. Yet, contrary to them, no specific adaptation to a dataset is necessary, and the Scattering Transform provides theoretically stability properties w.r.t. the roto-translation group and small deformations.

However, we could not integrate the full affine group. For example, the scaling transformations are missing. Several explanations can be found: first, contrary to the Euclidean group, applying a wavelet transform along the scale is non trivial because few scales are used in practice, and handling the border effect is difficult. Secondly, the work on the SLE, which is a very localized descriptor, in Chapter 4 indicates that scale variations might not be important variations. In [101], scales are linearized via a data augmentation using scales, which improves by about 5% relative percent the classification accuracies.

Besides, we do not capture color variation. The first layer of a deep network often incorporates a localized filter that performs a difference between for example the blue and its opposite color, the yellow. It has been suggested in [124] that this could possibly lead to an improved representation.

Finally, it is not clear if we want to enumerate each image variability since there can be a lot of them. Instead, similarly to CNNs, we would like to automatically discover them. Yet, contrary to CNNs, we try to find a way to explicit them (e.g. the work of Chapter 6).

We establish that predefined representations are of interest for deep learning applications and that learning all the layers from scratch is not necessary to

obtain a competitive deep network. We improve the Translation Scattering Transform implementation to scale it on ImageNet2012 via GPUs, and provide a wide range of results in settings where a limited amount of data is available. We also introduce the Share Local Encoder, which is a cascade of 1×1 convolutions that encodes non-overlapping patches of the Scattering Transform. While the localization is a strong constraint, we however were able to obtain competitive performances with the famous AlexNet [56].

Incorporating the Scattering permits to speed up the computations due to the learning procedure of CNNs since it acts as an efficient down-sampler. Most of the computation cost is due to the earlier layers, and it is possible to store the Scattering representation to obtain a significant speed-up of 20. The storage requires to buy some SSD to store the Scattering features. Efficient implementations could be also obtained with FGPA [20] chips.

Finally, the 1×1 layers brings an opportunity to interpret the layer of a CNN, because the representation is shallower and does not mix up different spatial locations. More structure should be incorporated in those layers or analyzed. Besides, this new image "descriptor" could find different applications than image classification, for example for detection of image matching as SIFT [65].

We have dissected several CNNs in Chapter 5 and show empirically that they build a progressively lower dimensional representation. We introduce a simplified class of CNNs which uses only convolutions and non-linearities and which involve no other extra-modules, while having a fixed depth and leading to the state-of-the-art on standard benchmarks. Variations of the hyper-parameters on the final classification accuracy have been studied, for example increasing the width of a CNN permits to increase its final accuracy to a similar extent to stacking more layers.

We however did not explain why the progressive dimensionality reduction has occurred. For example, investigating linearization phenomena is an important topic: several works [2, 71, 75] suggest that a deep network linearizes variabilities. The properties of the representation are a conundrum that will require to design new mathematical tools [123, 112].

Finally, we show that structure constraints can be incorporated in the convolutional operators of deep networks. This structure relies on a notion of symmetry of the classification task that has been introduced in [71, 16]. We define the Hierarchical Attribute CNNs, that are a subclass of Multiscale Hierarchical CNNs [71], which consist in a deep cascade of high-dimensional convolutions that we interpret as parallel transports along the symmetries of the classification task. This structure reduces drastically the number of parameters, yet, interpreting those symmetries remains a complex task.

There are at least two limitations in this work: the multi-dimensional convolution implementations are slow, and thus it is difficult to scale it on natural image tasks. Fortunately, more pipelines [17] are dedicated to fast tensor computations. And secondly, it is not clear if the symmetries that we captured are

related to attributes of image classification. Combining the SLE of the Chapter 4 with high-dimensional convolutions along the channel axis would be an interesting start.

To conclude, CNNs are efficient algorithms to solve complex high-dimensional tasks and their theoretical properties must be understood because their usage becomes more prominent in the industry: theoretical guarantees are necessary. For example, in settings like self-driving cars where the safety of the "driver" is critical, stability properties must be derived. Furthermore, this thesis suggests that pre-defined features are still of interest. Combined with appropriate learning methods, they could permit having more theoretical guarantees that are necessary to engineer better deep models and stable representations.

Bibliography

[1] J Andén, L Sifre, S Mallat, M Kapoko, V Lostanlen, and E Oyallon. Scatnet. *Research University Paris,[Online]. Available: http://www. di. ens. fr/data/software/scatnet/.[Accessed 13 July 2016]*, 2014.

[2] Mathieu Aubry and Bryan C Russell. Understanding deep features with computer-generated imagery. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2875–2883, 2015.

[3] Francis Bach. Breaking the curse of dimensionality with convex neural networks. *arXiv preprint arXiv:1412.8690*, 2014.

[4] Yoshua Bengio, Aaron C Courville, and Pascal Vincent. Unsupervised feature learning and deep learning: A review and new perspectives. *CoRR, abs/1206.5538*, 1, 2012.

[5] Yoshua Bengio and Olivier Delalleau. On the expressive power of deep architectures. In *International Conference on Algorithmic Learning Theory*, pages 18–36. Springer, 2011.

[6] 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*, pages 221–246. Springer, 2013.

[7] Michael Blot, Matthieu Cord, and Nicolas Thome. Max-min convolutional neural networks for image classification. In *Image Processing (ICIP), 2016 IEEE International Conference on*, pages 3678–3682. IEEE, 2016.

[8] Thomas Blumensath and Mike E Davies. On the difference between orthogonal matching pursuit and orthogonal least squares. 2007.

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

[10] Y-Lan Boureau, Nicolas Le Roux, Francis Bach, Jean Ponce, and Yann 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.

[11] Olivier Bousquet, Stéphane Boucheron, and Gábor Lugosi. Introduction to statistical learning theory. In *Advanced lectures on machine learning*, pages 169–207. Springer, 2004.

[12] MR Brito, AJ Quiroz, and Joseph E Yukich. Intrinsic dimension identification via graph-theoretic methods. *Journal of Multivariate Analysis*, 116:263–277, 2013.

[13] Joan Bruna and Stéphane Mallat. Classification with scattering operators. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 1561–1566. IEEE, 2011.

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

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

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

[17] David Budden, Alexander Matveev, Shibani Santurkar, Shraman Ray Chaudhuri, and Nir Shavit. Deep tensor convolution on multicores. *arXiv preprint arXiv:1611.06565*, 2016.

[18] Prabir Burman and Deborah Nolan. Location-adaptive density estimation and nearest-neighbor distance. *Journal of multivariate analysis*, 40(1):132–157, 1992.

[19] Micael Carvalho, Matthieu Cord, Sandra Avila, Nicolas Thome, and Eduardo Valle. Deep neural networks under stress. In *Image Processing (ICIP), 2016 IEEE International Conference on*, pages 4443–4447. IEEE, 2016.

[20] Jeff Chase, Brent Nelson, John Bodily, Zhaoyi Wei, and Dah-Jye Lee. Real-time optical flow calculations on fpga and gpu architectures: a comparison study. In *Field-Programmable Custom Computing Machines, 2008. FCCM'08. 16th International Symposium on*, pages 173–182. IEEE, 2008.

[21] S Chen and J Wigger. Fast orthogonal least squares algorithm for efficient subset model selection. *IEEE Transactions on Signal Processing*, 43(7):1713–1715, 1995.

[22] Yu Cheng, Felix X Yu, Rogerio S Feris, Sanjiv Kumar, Alok Choudhary, and Shi-Fu Chang. An exploration of parameter redundancy in deep networks with circulant projections. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2857–2865, 2015.

[23] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast and accurate deep network learning by exponential linear units (elus). *arXiv preprint arXiv:1511.07289*, 2015.

[24] Adam Coates and Andrew Y Ng. The importance of encoding versus training with sparse coding and vector quantization. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 921–928, 2011.

[25] Taco Cohen and Max Welling. Group equivariant convolutional networks. In *International Conference on Machine Learning*, pages 2990–2999, 2016.

[26] Ronald R Coifman and Matan Gavish. Harmonic analysis of digital data bases. In *Wavelets and Multiscale analysis*, pages 161–197. Springer, 2011.

[27] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[28] Jose A Costa, Abhishek Girotra, and AO Hero. Estimating local intrinsic dimension with k-nearest neighbor graphs. In *Statistical Signal Processing, 2005 IEEE/SP 13th Workshop on*, pages 417–422. IEEE, 2005.

[29] George Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals, and Systems (MCSS)*, 2(4):303–314, 1989.

[30] Navneet Dalal and Bill 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.

[31] Ingrid Daubechies. *Ten lectures on wavelets*. SIAM, 1992.

[32] Emily L Denton, Wojciech Zaremba, Joan Bruna, Yann LeCun, and Rob Fergus. Exploiting linear structure within convolutional networks for efficient evaluation. In *Advances in Neural Information Processing Systems*, pages 1269–1277, 2014.

[33] Ivan Dokmanić, Joan Bruna, Stéphane Mallat, and Maarten de Hoop. Inverse problems with invariant multiscale statistics. *arXiv preprint arXiv:1609.05502*, 2016.

[34] 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*, pages 766–774, 2014.

[35] Thibaut Durand, Taylor Mordan, Nicolas Thome, and Matthieu Cord. Wildcat: Weakly supervised learning of deep convnets for image classification, pointwise localization and segmentation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[36] Michael Eickenberg, Alexandre Gramfort, Gaël Varoquaux, and Bertrand Thirion. Seeing it all: Convolutional network layers map the function of the human visual system. *NeuroImage*, 152:184–194, 2017.

[37] Leon A Gatys, Alexander S Ecker, and Matthias Bethge. Image style transfer using convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2414–2423, 2016.

[38] Robert Gens and Pedro M Domingos. Deep symmetry networks. In *Advances in neural information processing systems*, pages 2537–2545, 2014.

[39] Raja Giryes, Guillermo Sapiro, and Alex M Bronstein. Deep neural networks with random gaussian weights: A universal classification strategy. *CoRR, abs/1504.08291*, 2015.

[40] Ian Goodfellow, Honglak Lee, Quoc V Le, Andrew Saxe, and Andrew Y Ng. Measuring invariances in deep networks. In *Advances in neural information processing systems*, pages 646–654, 2009.

[41] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

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

[43] Ian J Goodfellow, Oriol Vinyals, and Andrew M Saxe. Qualitatively characterizing neural network optimization problems. *arXiv preprint arXiv:1412.6544*, 2014.

[44] David Ha, Andrew Dai, and Quoc V Le. Hypernetworks. arxiv preprint. *arXiv preprint arXiv:1609.09106*, 2016.

[45] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. In *European Conference on Computer Vision*, pages 346–361. Springer, 2014.

[46] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.

[47] Elad Hoffer, Itay Hubara, and Nir Ailon. Deep unsupervised learning through spatial contrasting. *arXiv preprint arXiv:1610.00243*, 2016.

[48] Gilbert Agnew Hunt. Semi-groups of measures on lie groups. *Transactions of the American Mathematical Society*, 81(2):264–293, 1956.

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

[50] Jörn-Henrik Jacobsen, Edouard Oyallon, Stéphane Mallat, and Arnold WM Smeulders. Multiscale hierarchical convolutional networks. *arXiv preprint arXiv:1703.04140*, 2017.

[51] Jorn-Henrik Jacobsen, Jan van Gemert, Zhongyu Lou, and Arnold WM Smeulders. Structured receptive fields in cnns. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2610–2619, 2016.

[52] Max Jaderberg, Andrea Vedaldi, and Andrew Zisserman. Speeding up convolutional neural networks with low rank expansions. *arXiv preprint arXiv:1405.3866*, 2014.

[53] Yangqing Jia, Chang Huang, and Trevor Darrell. Beyond spatial pyramids: Receptive field learning for pooled image features. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 3370–3377. IEEE, 2012.

[54] Andreas Klöckner, Nicolas Pinto, Yunsup Lee, Bryan Catanzaro, Paul Ivanov, and Ahmed Fasih. Pycuda and pyopencl: A scripting-based approach to gpu run-time code generation. *Parallel Computing*, 38(3):157–174, 2012.

[55] Jan J Koenderink and Andrea J Van Doorn. The structure of locally orderless images. *International Journal of Computer Vision*, 31(2):159–168, 1999.

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

[57] Svetlana Lazebnik, Cordelia Schmid, and Jean 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.

[58] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

[59] Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989.

[60] Yann LeCun, Koray Kavukcuoglu, and Clément Farabet. Convolutional networks and applications in vision. In *Circuits and Systems (ISCAS), Proceedings of 2010 IEEE International Symposium on*, pages 253–256. IEEE, 2010.

[61] Honglak Lee, Roger Grosse, Rajesh Ranganath, and Andrew Y Ng. Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In *Proceedings of the 26th annual international conference on machine learning*, pages 609–616. ACM, 2009.

[62] Jerome Y Lettvin, Humberto R Maturana, Warren S McCulloch, and Walter H Pitts. What the frog's eye tells the frog's brain. *Proceedings of the IRE*, 47(11):1940–1951, 1959.

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

[64] Tsung-Han Lin and HT Kung. Stable and efficient representation learning with nonnegativity constraints. In *ICML*, pages 1323–1331, 2014.

[65] David 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.

[66] Zhiyun Lu, Vikas Sindhwani, and Tara N Sainath. Learning compact recurrent neural networks. In *Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on*, pages 5960–5964. IEEE, 2016.

[67] Mateusz Malinowski and Mario Fritz. Learning smooth pooling regions for visual recognition. In *24th British Machine Vision Conference*, pages 1–11. BMVA Press, 2013.

[68] Stéphane Mallat. *A wavelet tour of signal processing*. Academic press, 1999.

[69] Stéphane Mallat. Recursive interferometric representation. In *Proc. of EUSICO conference, Danemark*, 2010.

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

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

[72] Stéphane Mallat and Irene Waldspurger. Deep learning by scattering. *arXiv preprint arXiv:1306.5532*, 2013.

[73] Sancho McCann and David G Lowe. Spatially local coding for object recognition. In *Asian Conference on Computer Vision*, pages 204–217. Springer, 2012.

[74] Yves Meyer and L Ondelettes. Algorithms and applications. *SIAM, philadelphia*, 1993.

[75] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

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

[77] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, Omar Fawzi, and Pascal Frossard. Universal adversarial perturbations. *arXiv preprint arXiv:1610.08401*, 2016.

[78] José E Moyal. Quantum mechanics as a statistical theory. In *Mathematical Proceedings of the Cambridge Philosophical Society*, volume 45, pages 99–124. Cambridge Univ Press, 1949.

[79] Tasha Nagamine, Michael L Seltzer, and Nima Mesgarani. Exploring how deep neural networks form phonemic categories. In *INTERSPEECH*, pages 1912–1916, 2015.

[80] Maxime Oquab, Léon Bottou, Ivan Laptev, and Josef Sivic. Is object localization for free?-weakly-supervised learning with convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 685–694, 2015.

[81] Wanli Ouyang and Xiaogang Wang. Joint deep learning for pedestrian detection. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2056–2063, 2013.

[82] Edouard Oyallon. A hybrid network: Scattering and convnet. 2016.

[83] Edouard Oyallon. Building a regular decision boundary with deep networks. *arXiv preprint arXiv:1703.01775*, 2017.

[84] Edouard Oyallon, Eugene Belilovsky, and Sergey Zagoruyko. Scaling the scattering transform: Deep hybrid networks. *arXiv preprint arXiv:1703.08961*, 2017.

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

[86] Edouard Oyallon, Stéphane Mallat, and Laurent Sifre. Generic deep networks with wavelet scattering. *arXiv preprint arXiv:1312.5940*, 2013.

[87] Xingchao Peng, Baochen Sun, Karim Ali, and Kate Saenko. Exploring invariances in deep convolutional neural networks using synthetic images. *CoRR, abs/1412.7122*, 2(4), 2014.

[88] 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*, pages 3743–3752, 2015.

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

[90] Mirco Ravanelli, Benjamin Elizalde, Karl Ni, and Gerald Friedland. Audio concept classification with hierarchical deep neural networks. In *Signal Processing Conference (EUSIPCO), 2014 Proceedings of the 22nd European*, pages 606–610. IEEE, 2014.

[91] Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, Carlo Gatta, and Yoshua Bengio. Fitnets: Hints for thin deep nets. *arXiv preprint arXiv:1412.6550*, 2014.

[92] Frank Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386, 1958.

[93] Frank Rosenblatt. Principles of neurodynamics. perceptrons and the theory of brain mechanisms. Technical report, DTIC Document, 1961.

[94] Jorge Sánchez and Florent 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.

[95] Jorge Sánchez, Florent Perronnin, Thomas Mensink, and Jakob Verbeek. Image classification with the fisher vector: Theory and practice. *International journal of computer vision*, 105(3):222–245, 2013.

[96] Andrew Saxe, Pang W Koh, Zhenghao Chen, Maneesh Bhand, Bipin Suresh, and Andrew Y Ng. On random weights and unsupervised feature learning. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pages 1089–1096, 2011.

[97] Ivan W Selesnick, Richard G Baraniuk, and Nick C Kingsbury. The dual-tree complex wavelet transform. *IEEE signal processing magazine*, 22(6):123–151, 2005.

[98] Thomas Serre and Maximilian 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.

[99] Thomas Serre, Lior Wolf, and Tomaso Poggio. Object recognition with features inspired by visual cortex. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 2, pages 994–1000. Ieee, 2005.

[100] Laurent Sifre and Stéphane Mallat. Combined scattering for rotation invariant texture analysis. In *ESANN*, volume 44, pages 68–81, 2012.

[101] 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*, pages 1233–1240, 2013.

[102] Laurent Sifre and Stéphane Mallat. Ecole polytechnique, cmap phd thesis rigid-motion scattering for image classification author. 2014.

[103] 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.

[104] Kihyuk Sohn and Honglak Lee. Learning invariant representations with local transformations. *arXiv preprint arXiv:1206.6418*, 2012.

[105] Jure Sokolic, Raja Giryes, Guillermo Sapiro, and Miguel RD Rodrigues. Robust large margin deep neural networks. *arXiv preprint arXiv:1605.08254*, 2016.

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

[107] Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[108] Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. Highway networks. *arXiv preprint arXiv:1505.00387*, 2015.

[109] Mitsuo Sugiura. *Unitary representations and harmonic analysis: an introduction*, volume 44. Elsevier, 1990.

[110] Ilya Sutskever, James Martens, and Geoffrey E Hinton. Generating text with recurrent neural networks. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 1017–1024, 2011.

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

[112] 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.

[113] Vladimir N Vapnik. Methods of pattern recognition. In *The nature of statistical learning theory*, pages 123–180. Springer, 2000.

[114] Subhashini Venugopalan, Huijuan Xu, Jeff Donahue, Marcus Rohrbach, Raymond Mooney, and Kate Saenko. Translating videos to natural language using deep recurrent neural networks. *arXiv preprint arXiv:1412.4729*, 2014.

[115] Irène Waldspurger. Wavelet transform modulus: phase retrieval and scattering. 2015.

[116] Li Wan, Matthew Zeiler, Sixin Zhang, Yann L Cun, and Rob Fergus. Regularization of neural networks using dropconnect. In *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, pages 1058–1066, 2013.

[117] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? In *Advances in neural information processing systems*, pages 3320–3328, 2014.

[118] Jason Yosinski, Jeff Clune, Anh Nguyen, Thomas Fuchs, and Hod Lipson. Understanding neural networks through deep visualization. *arXiv preprint arXiv:1506.06579*, 2015.

[119] Sergey Zagoruyko and Nikos Komodakis. Paying more attention to attention: Improving the performance of convolutional neural networks via attention transfer. *arXiv preprint arXiv:1612.03928*, 2016.

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

[121] Sergey Zagoruyko and Nikos Komodakis. Diracnets: Training very deep neural networks without skip-connections. *arXiv preprint arXiv:1706.00388*, 2017.

[122] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *European conference on computer vision*, pages 818–833. Springer, 2014.

[123] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. *arXiv preprint arXiv:1611.03530*, 2016.

[124] Jun Zhang, Youssef Barhomi, and Thomas Serre. A new biologically inspired color image descriptor. *Computer vision–ECCV 2012*, pages 312–324, 2012.

[125] Liang Zheng, Yali Zhao, Shengjin Wang, Jingdong Wang, and Qi Tian. Good practice in cnn feature transfer. *arXiv preprint arXiv:1604.00133*, 2016.

[126] Stephan Zheng, Yang Song, Thomas Leung, and Ian Goodfellow. Improving the robustness of deep neural networks via stability training. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4480–4488, 2016.

Résumé

Cette thèse étudie des propriétés empiriques des réseaux de neurones convolutifs profonds, et en particulier de la transformée en Scattering. En effet, l'analyse théorique de ces derniers est difficile et représente jusqu'à ce jour un défi: les couches successives de neurones ont la capacité de réaliser des opérations complexes, dont la nature est encore inconnue, via des algorithmes d'apprentissages dont les garanties de convergences ne sont pas bien comprises. Pourtant, ces réseaux de neurones sont de formidables outils pour s'attaquer à une grande variété de tâches difficiles telles la classification d'images, ou plus simplement effectuer des prédictions. La transformée de Scattering est un opérateur mathématique, non-linéaire dont les spécifications sont inspirées par les réseaux convolutifs. Dans ce travail, elle est appliquée sur des images naturelles et obtient des résultats compétitifs avec les architectures non-supervisées. En plaçant un réseau de neurones convolutifs supervisés à la suite du Scattering, on obtient des performances compétitives sur ImageNet2012, qui est le plus grand jeux de donnée d'images étiquetées accessibles aux chercheurs. Cela nécessite d'implémenter un algorithme efficace sur carte graphique. Dans un second temps, cette thèse s'intéresse aux propriétés des couches à différentes profondeurs. On montre qu'un phénomène de réduction de dimensionnalité progressif à lieu et on s'intéresse aux propriétés de classifications supervisées lorsqu'on varie des hyper paramètres de ces réseaux. Finalement, on introduit une nouvelle classe de réseaux convolutifs, dont les opérateurs sont structurés par des groupes de symétries du problème de classification.

Mots Clés

Réseaux de neurones profonds, traitement du signal, apprentissage, vision par ordinateur

Abstract

This thesis studies empirical properties of deep convolutional neural networks, and in particular the Scattering Transform. Indeed, the theoretical analysis of the latter is hard and until now remains a challenge: successive layers of neurons have the ability to produce complex computations, whose nature is still unknown, thanks to learning algorithms whose convergence guarantees are not well understood. However, those neural networks are outstanding tools to tackle a wide variety of difficult tasks, like image classification or more formally statistical prediction. The Scattering Transform is a non-linear mathematical operator whose properties are inspired by convolutional networks. In this work, we apply it to natural images, and obtain competitive accuracies with unsupervised architectures. Cascading a supervised neural networks after the Scattering permits to compete on ImageNet2012, which is the largest dataset of labeled images available. An efficient GPU implementation is provided. Then, this thesis focuses on the properties of layers of neurons at various depths. We show that a progressive dimensionality reduction occurs and we study the numerical properties of the supervised classification when we vary the hyper parameters of the network. Finally, we introduce a new class of convolutional networks, whose linear operators are structured by the symmetry groups of the classification task.

Key Words

Deep learning, signal processing, machine learning, computer vision